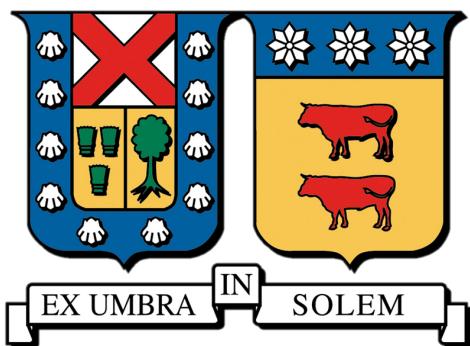


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INGENIERÍA MECÁNICA
VALPARAÍSO - CHILE



**APRENDIZAJE PROFUNDO PARA LA
ELECTROSTÁTICA DE MACROMOLÉCULAS:
RESOLVIENDO LA ECUACIÓN DE
POISSON-BOLTZMANN A PARTIR DE REDES
NEURONALES INFORMADAS POR LA FÍSICA**

MARTÍN ANDRÉS ACHONDO MERCADO

TESIS DE GRADO PARA OPTAR AL GRADO DE:
MAGÍSTER EN CIENCIAS DE INGENIERÍA MECÁNICA
Y AL TÍTULO DE:
INGENIERO CIVIL MECÁNICO

Profesor Guía: PhD. Christopher Cooper V.

Profesor Correferente Interno: PhD. Joaquín Mura M.

Profesor Correferente Externo: PhD. Ignacio Muga U.

Agosto - 2024

Agradecimientos

Quiero partir agradeciendo a mi mamá Verónica y a mi papá Matías, por todas sus enseñanzas y por siempre creer en mí. Se que ellos dieron todo lo que tenían para que yo pudiera crecer y estudiar, primero en el colegio y luego en la universidad. Sin ellos, no podría haber llegado a este punto de mi vida académica y personal.

También quiero agradecer a mis hermanos, Pablo y Joaquín, mis amigos de la vida, con quienes crecí y siempre los tuve a mi lado incondicionalmente, y a Dokiano, que siempre estuvo acompañándome mientras escribía esta Tesis.

Me gustaría agradecer a mi polola, Bernardita, mi compañera de las últimas aventuras, por ser quien siempre lograba levantarme el ánimo en los momentos más difíciles y darme la motivación para llegar a este punto.

De igual manera, quiero agradecer a mis amigos de siempre, Nicolás y José Tomás, y a mis amigos que conocí en esta etapa: Klaus, Enrique, Eduardo, Matías y Catalina, con quienes lograba desconectarme, por todas las risas y anécdotas que vivimos.

Por último, me gustaría agradecer a mi profesor Christopher, por su confianza, el conocimiento que me traspasó y por la motivación que me entregó a lo largo de esta investigación.

Gracias.

Resumen

Las Redes Neuronales Informadas por la Física (PINNs) han sido aplicadas en diversas áreas para resolver Ecuaciones Diferenciales Parciales (PDEs), generando un nuevo método de resolución en el campo de la computación científica. El uso de redes neuronales permite incluir información adicional que no está presente en la PDE, como datos experimentales o resultados de otras simulaciones. Esto es particularmente útil en la electrostática molecular, donde comúnmente se resuelve la Ecuación de Poisson-Boltzmann (PB) basada en el Modelo de Solvente Implícito, sin considerar toda la dinámica molecular.

En este trabajo, se desarrolla una metodología para utilizar PINNs en la resolución de la Ecuación de Poisson-Boltzmann, aplicándola a diversas moléculas de distintos tamaños. Se revisaron las formulaciones de la ecuación de PB que mejor se adaptan a PINNs, las arquitecturas que permiten representar adecuadamente la solución y los factores que afectan el proceso de minimización. Para todas las moléculas, se obtuvieron buenos resultados en la energía de solvatación, que resultó ser un indicador bastante robusto, y en el potencial de reacción, logrando errores del orden de 10^{-3} en ambos indicadores.

Para una implementación efectiva de PINNs, se encontró que es necesario regularizar la ecuación de PB y utilizar dos redes neuronales, una por subdominio (sólido y solvente). Además, se observó que la correcta representación de la geometría molecular es crucial para la convergencia a la solución real del problema. En este caso, se propone generar una malla fina de la superficie y obtener los puntos de colocación aleatoriamente desde sus elementos.

Sin embargo, se descubrió que falta investigación para incluir términos adicionales basados en integrales en la función de pérdida (como datos experimentales), ya que interrumpen el proceso de minimización de los residuales. No obstante, este trabajo sirve como punto de partida para alcanzar esos objetivos.

Finalmente, todo el código fue implementado como una librería llamada XPPBE, para facilitar su uso por la comunidad científica.

Palabras claves: Ecuación de Poisson-Boltzmann, PINNs.

Abstract

Physics-Informed Neural Networks (PINNs) have been applied in various fields to solve Partial Differential Equations (PDEs), creating a new method for solving problems in the field of scientific computing. The use of neural networks allows for the inclusion of additional information not present in the PDE, such as experimental data or results from other simulations. This is particularly useful in molecular electrostatics, where the Poisson-Boltzmann Equation (PB) is commonly solved based on the Implicit Solvent Model, without considering the full molecular dynamics.

In this work, a methodology is developed to use PINNs to solve the Poisson-Boltzmann Equation, applying it to various molecules of different sizes. The formulations of the PB equation that best suit PINNs, the architectures that adequately represent the solution, and the factors that affect the minimization process were reviewed. For all molecules, good results were obtained in solvation energy, which proved to be a quite robust indicator, and in reaction potential, achieving errors of the order of 10^{-3} in both indicators.

For an effective implementation of PINNs, it was found necessary to regularize the PB equation and use two neural networks, one for each subdomain (solute and solvent). Additionally, it was observed that the correct representation of molecular geometry is crucial for convergence to the real solution of the problem. In this case, it is proposed to generate a fine mesh of the surface and obtain the collocation points randomly from its elements.

However, it was discovered that more research is needed to include additional terms based on integrals in the loss function (such as experimental data), as they disrupt the residual minimization process. Nonetheless, this work serves as a starting point to achieve these goals.

Finally, all the code was implemented as a library called XPPBE to facilitate its use by the scientific community

Keywords: Poisson-Boltzmann Equation, PINNs.

Índice General

Agradecimientos	I
Resumen	II
Abstract	III
1. Introducción	1
1.1. Objetivo General	3
1.2. Objetivos Específicos	3
2. Marco Teórico	4
2.1. Modelo de Solvente Implícito	4
2.1.1. Electrostática en un Medio Dieléctrico Continuo	6
2.1.2. Ecuación de Poisson-Boltzmann	8
2.1.2.1. Ecuación de Poisson Boltzmann Linealizada	10
2.1.3. Interfaz Soluto-Solvente	11
2.1.4. Esquemas de Regularización de la Ecuación de Poisson-Boltzmann	12
2.1.5. Formulación Integral en la Frontera para la Ecuación de Poisson-Boltzmann	13
2.1.6. Energías Libres	14
2.1.6.1. Energía Libre No Polar	15
2.1.6.2. Energía Libre Electrostática	15
2.1.6.3. Energía Libre de Coulomb	17
2.1.6.4. Energía de Solvatación	17
2.1.7. Mediciones Experimentales	18
2.1.8. Soluciones Analíticas	19
2.1.8.1. Ion de Born	20
2.1.8.2. Multipolo Esférico	21

2.1.8.3. Cargas Puntuales Inmersas en Solvente	22
2.2. Redes Neuronales Artificiales	24
2.2.1. Arquitecturas	25
2.2.1.1. Multi Layer Perceptron	25
2.2.1.2. Residual Neural Network	27
2.2.1.3. Convolutional Neural Network	28
2.2.1.4. Recurrent Neural Network	28
2.2.1.5. Funciones de Activación	29
2.2.2. Función de Pérdida	29
2.2.3. Métodos de Optimización	30
2.2.3.1. Descenso de Gradiente	30
2.2.3.2. ADAM (Adaptive Moment Estimation)	31
2.2.3.3. BFGS (Broyden–Fletcher–Goldfarb–Shanno)	32
2.2.4. Diferenciación Automática y Back Propagation	33
2.3. Physics Informed Neural Networks	36
2.3.1. Métodos Basados en Puntos de Colocación	38
2.3.1.1. Descomposición de Dominios	39
2.3.2. Métodos Basados en la Formulación Variacional	40
2.3.3. Métodos Basados en la Formulación Integral en la Frontera	40
2.3.4. Métodos Basados en el Aprendizaje de Operadores	41
3. Metodología	43
3.1. PINNs Aplicado a PDEs Elípticas con Interfaz	43
3.1.1. Aplicación a Poisson-Boltzmann	47
3.2. Implementación de PINNs Aplicado a Poisson-Boltzmann	48
3.2.1. Ecuación de Poisson-Boltzmann con Formulación Directa	49
3.2.2. Ecuación de Poisson-Boltzmann con Esquema de Regularización 1	51
3.2.3. Ecuación de Poisson-Boltzmann con Esquema de Regularización 2	52
3.2.4. Términos Adicionales de la Función de Pérdida	53
3.3. Cálculo de Operadores Diferenciales con Diferenciación Automática	55
3.4. Puntos de Colocación y Mallas	58
3.4.1. Muestreo Aleatorio	59
3.4.2. Muestreo Fijo	61
3.5. Implementaciones Adicionales	62
3.5.1. Arquitecturas	62

3.5.1.1. Factorización Aleatoria de los Pesos de la Red	63
3.5.1.2. Capas de Escalamiento	63
3.5.1.2.1. Estimación de las Cotas del Potencial	64
3.5.1.3. Capa de Fourier	65
3.5.1.4. Función de Activación Entrenable	65
3.5.2. Algoritmo de Ponderación para la Función de Pérdida	65
3.5.3. Adimensionalización	66
3.6. Validación de Resultados	67
3.6.1. Función de Pérdida de Entrenamiento y de Validación	67
3.6.2. Energía de Solvatación	68
3.6.3. Potencial de Reacción	69
3.7. Código Implementado	70
3.7.1. Softwares y Librerías Utilizadas	72
4. Resultados y Análisis	74
4.1. Pruebas Realizadas	74
4.2. Ion de Born	77
4.2.1. Estudio de Esquemas y Descomposición de Dominios	78
4.2.1.1. Uso de la Descomposición de Dominios	78
4.2.1.2. Sin Descomposición de Dominios	80
4.2.2. Estudio de Puntos de Colocación	83
4.2.3. Estudio de Arquitectura	86
4.2.4. Estudio de Funciones de Pérdida	93
4.2.5. Estudio del Método de Optimización	99
4.3. Multipolo Esférico	102
4.3.1. Carga Descentralizada	103
4.3.2. 2 Cargas Positivas	106
4.3.3. 2 Cargas de Signos Opuestos	109
4.3.4. Colección de Cargas	112
4.4. Metanol	115
4.5. Arginina	124
4.5.1. Estudio de Puntos de Colocación	130
4.5.2. Uso de Datos Experimentales	135
4.6. Ubicuitina	137
4.7. ADN	142

5. Conclusiones	147
Bibliografía	149
Anexos	157
A. Información Estructural de Moléculas	157
A.1. Archivos PQR	157
B. Librería XPPBE	159
B.1. Ejemplo de Uso de XPPBE	159
B.2. Ejemplo del Archivo Requerido por XPPBE	160
B.3. Instalación de XPPBE	162

Índice de Figuras

2.1.	Modelos para el solvente	4
2.2.	Aproximación del modelo de Solvente Implícito	5
2.3.	Esquema del modelo de Solvente Implícito	8
2.4.	Definiciones de interfaz	11
2.5.	Superficie SES de referencia para la Lisozima	11
2.6.	Ciclo termodinámico para las energías libres	14
2.7.	Ion de Born	20
2.8.	Esquema del perceptrón	25
2.9.	Esquema de MLP	26
2.10.	Esquema de ResNet para un bloque oculto B	27
2.11.	Esquema de métodos PINN	37
3.1.	Representación del dominio con interfaz Γ	43
3.2.	Esquema de método PINN con 2 dominios y 1 optimizador	49
3.3.	Ejemplos de mallas para la arginina	58
3.4.	Ejemplo de puntos de colocación para la arginina	61
3.5.	Esquema de la superposición de las distintas cargas puntuales	64
3.6.	Repositorio de GitHub	71
3.7.	Estructura de archivos para el código implementado	72
4.1.	Esquema y malla para el Ion de Born	77
4.2.	Potencial de reacción para los distintos esquemas de la ecuación de PB	79
4.3.	Energía de solvatación para los distintos esquemas de la ecuación de PB	80
4.4.	Potencial de reacción para los distintos esquemas de la ecuación de PB	81
4.5.	Energía de solvatación para los distintos esquemas de la ecuación de PB	82
4.6.	Evolución de la energía de solvatación para estudio de puntos de colocación	84
4.7.	Evolución del loss de entrenamiento para estudio de puntos de colocación	84

4.8. Potencial de reacción para estudio de puntos de colocación	85
4.9. Potencial de reacción para las distintas arquitecturas (usando capa de Fourier) .	88
4.10. Potencial de reacción para las distintas arquitecturas (usando factorización de parámetros y capa Fourier)	89
4.11. Potencial de reacción para las distintas arquitecturas (sin capa de Fourier) . .	90
4.12. Potencial de reacción (usos C. Escal. y Func. entr.)	90
4.13. Energía de solvatación y loss de entrenamiento para revisión de número de capas y neuronas	91
4.14. Energía de solvatación y loss de entrenamiento para las configuraciones a mantener para el multipolo esférico	92
4.15. Energía de solvatación y loss de entrenamiento para estudio de función de pérdida	94
4.16. Potencial de reacción para distintos casos, estudio de función de pérdida . .	95
4.17. Términos de la función de pérdida para los casos probados	96
4.18. Potencial de reacción para casos con distintos términos en la función de pérdida	97
4.19. Términos de la función de pérdida para los casos probados	98
4.20. Loss de entrenamiento y de validación para estudio del método de optimización	100
4.21. Potencial de reacción para estudio del método de optimización	101
4.22. Energía de solvatación y loss de entrenamiento para carga descentrada	103
4.23. Potencial de reacción para carga descentrada, dirección $\theta = 0, \phi = \pi/2$	104
4.24. Potencial de reacción para carga descentrada, dirección $\theta = \pi/2, \phi = \pi$	105
4.25. Energía de solvatación y loss de entrenamiento para 2 cargas positivas	106
4.26. Potencial de reacción para 2 cargas positivas, dirección $\theta = 0, \phi = \pi/2$	107
4.27. Potencial de reacción para 2 cargas positivas, dirección $\theta = \pi/2, \phi = \pi$	108
4.28. Energía de solvatación y loss de entrenamiento para 2 cargas de signos opuestos	109
4.29. Potencial de reacción para 2 cargas opuestas, dirección $\theta = 0, \phi = \pi/2$	110
4.30. Potencial de reacción para 2 cargas opuestas, dirección $\theta = \pi/2, \phi = \pi$	111
4.31. Energía de solvatación y loss de entrenamiento para colección de cargas	112
4.32. Potencial de reacción para colección de cargas, dirección $\theta = 0, \phi = \pi/2$	113
4.33. Potencial de reacción para colección de cargas, dirección $\theta = \pi/2, \phi = \pi/2$. .	114
4.34. Estructura y malla para el metanol	115
4.35. Energía de solvatación y loss de entrenamiento para los casos de prueba del metanol	116
4.36. Potencial de reacción para el metanol en 3 dirección diferentes	117
4.37. Potencial de reacción en la interfaz del metanol	118
4.38. Error absoluto y relativo en la interfaz del metanol	119

4.39. Energía de solvatación y loss de entrenamiento al no usar datos puntuales para el metanol	120
4.40. Potencial de reacción para el metanol en 3 dirección diferentes	121
4.41. Potencial de reacción en la interfaz del metanol	122
4.42. Error absoluto y relativo en la interfaz del metanol	123
4.43. Estructura y malla para la arginina	124
4.44. Energía de solvatación y loss de entrenamiento para los casos de prueba de la arginina	125
4.45. Potencial de reacción para la arginina en 3 direcciones diferentes	126
4.46. Potencial de reacción en la interfaz de la arginina	127
4.47. Error absoluto y relativo en la interfaz de la arginina	128
4.48. Energía de solvatación y loss de entrenamiento para los casos de prueba de la arginina	129
4.49. Potencial de reacción para la arginina usando distintas mallas, dirección $\theta = \pi/2$, $\phi = \pi/2$	131
4.50. Evolución de la energía de solvatación de la arginina para las distintas mallas . .	132
4.51. Potencial de reacción en la interfaz de la arginina para las distintas mallas . .	133
4.52. Potencial de reacción para la arginina, dirección $\theta = \pi/2$, $\phi = \pi/2$	134
4.53. Energía de solvatación y loss de entrenamiento para uso de datos experimentales	135
4.54. Potencial de reacción para la arginina, uso de datos experimentales	136
4.55. Estructura y malla para la ubicuitina	137
4.56. Energía de solvatación y loss de entrenamiento para los casos de la ubicuitina .	138
4.57. Potencial de reacción para la ubicuitina, dirección $\theta = 0$, $\phi = \pi/2$	139
4.58. Potencial de reacción en la interfaz de la ubicuitina	140
4.59. Error absoluto y relativo en la interfaz de la ubicuitina, caso no escalado	141
4.60. Error absoluto y relativo en la interfaz de la ubicuitina, caso escalado	141
4.61. Estructura y malla para el ADN	142
4.62. Energía de solvatación (componente lineal) del ADN para casos lineal y no lineal	143
4.63. Potencial de reacción para el ADN, caso lineal	144
4.64. Potencial de reacción para el ADN, caso no lineal	144
4.65. Potencial de reacción en la interfaz del ADN, caso lineal	145
4.66. Potencial de reacción en la interfaz del ADN, caso no lineal	146
4.67. Error absoluto y relativo en la interfaz del ADN, caso lineal	146
4.68. Error absoluto y relativo en la interfaz del ADN, caso no lineal	146

Índice de Tablas

3.1. Adimensionalización Poisson-Boltzmann	67
4.1. Puntos de colocación para el estudio de esquemas	78
4.2. Resultados para el estudio de esquemas utilizando descomposición de dominios .	78
4.3. Resultados para el estudio de esquemas sin descomposición de dominios	80
4.4. Cantidad de puntos de colocación para ambos métodos de muestreo	83
4.5. Resultados para el estudio de puntos de colocación	83
4.6. Casos de prueba para estudio de arquitectura	86
4.7. Resultados para estudio de arquitectura	87
4.8. Configuraciones a mantener para multipolo esférico	92
4.9. Puntos de colocación para estudio de función de pérdida	93
4.10. Casos de prueba para estudio de función de pérdida	93
4.11. Resultados para estudio de función de pérdida	94
4.12. Casos de prueba para estudio del método de optimización	99
4.13. Resultados para estudio del método de optimización	100
4.14. Casos de prueba para el multipolo esférico	102
4.15. Puntos de colocación para el multipolo esférico	103
4.16. Resultados para la carga descentrada	103
4.17. Resultados para 2 cargas positivas	106
4.18. Resultados para 2 cargas de signos opuestos	109
4.19. Resultados para colección de cargas	112
4.20. Casos de prueba para el metanol	115
4.21. Puntos de colocación para el metanol	116
4.22. Resultados para los casos de prueba del metanol	116
4.23. Resultados al probar mismos casos del metanol sin datos puntuales	120
4.24. Puntos de colocación para la arginina	124
4.25. Resultados para la arginina, uso de datos puntuales	125

Índice de Tablas

4.26. Resultados para la arginina, uso de la capa de escalamiento del output	128
4.27. Puntos de colocación para las distintas mallas de la arginina	130
4.28. Resultados para estudio de malla de la arginina	130
4.29. Puntos de colocación utilizados para las nuevas mallas de la arginina	133
4.30. Resultados para estudio de malla de la arginina	134
4.31. Puntos de colocación para la arginina, uso de datos experimentales	135
4.32. Resultados para el uso de datos experimentales ϕ_{ENS} para la arginina	135
4.33. Puntos de colocación utilizados para la ubicuitina	138
4.34. Resultados para el uso de la capa del escalamiento del output para la ubicuitina	138
4.35. Puntos de colocación utilizados para el ADN	143
4.36. Resultados de ambos casos para el ADN	143
 A.1. Archivo .pqr para el Ion de Born	157
A.2. Archivo .pqr para la esfera con carga descentrada	157
A.3. Archivo .pqr para la esfera con 2 cargas positivas	158
A.4. Archivo .pqr para la esfera con 2 cargas de signos opuestos	158
A.5. Archivo .pqr para la esfera con colección de 5 cargas	158

Índice de Algoritmos

3.1.	Algoritmo para obtener puntos de colocación aleatorios \mathcal{S}	60
3.2.	Algoritmo para ponderación de pesos	66
3.3.	Algoritmo del código implementado con PINNs: XPPBE	71

Índice de Códigos

3.1. Ejemplo de uso básico de la librería XPPBE	70
B.1. Ejemplo de uso de la librería XPPBE más post-procesamiento	159
B.2. Archivo .yaml de ejemplo	160
B.3. Ejemplo de instalación de XPPBE	162

Capítulo 1

Introducción

La electrostática molecular es de suma relevancia en el campo de la biofísica, ya que detalla las interacciones entre macromoléculas, proteínas, ácidos nucleicos y otras biomoléculas, incluso con superficies cargadas. Este campo tiene aplicación directa, como por ejemplo el transporte de fármacos en la sangre, o el uso de biosensores. Comúnmente, estos sistemas se modelan mediante la ecuación de Poisson-Boltzmann (PBE), basada en el modelo de Solvente Implícito [1]. Al resolver esta ecuación, se puede obtener el potencial electrostático en todo el espacio, y con esto calcular variables de interés, como la energía de solvatación, las fuerzas eléctricas y los cálculos de pKa [2, 3, 4]. Para resolver esta ecuación, normalmente se utilizan técnicas como el Método de Elementos de Frontera (BEM), el Método de Elementos Finitos (FEM) o el Método de Diferencias Finitas (FDM).

Por otra parte, el avance en las tecnologías computacionales ha potenciado el uso de redes neuronales en diversas áreas de la ciencia para resolver problemas complejos [5]. En este contexto, las Redes Neuronales Informadas por la Física (PINNs) han emergido como una herramienta para resolver Ecuaciones Diferenciales Parciales (PDEs), combinando principios de la física con el poder del aprendizaje profundo [6, 7, 8]. Estas redes neuronales logran replicar la solución de la ecuación diferencial a resolver minimizando una función de pérdida, ofreciendo una alternativa innovadora a los métodos tradicionales.

Utilizar PINNs para resolver la ecuación de Poisson-Boltzmann representa un gran avance en la modelación de la electrostática molecular mediante el aprendizaje profundo (DL). Las PINNs permiten incluir mediciones experimentales o restricciones físicas en la función de pérdida a minimizar, lo que puede conducir a nuevas soluciones que los métodos tradicionales no pueden alcanzar. Además, la capacidad de agregar términos adicionales a la función de pérdida permite

1. Introducción

incorporar información sobre modelaciones de dinámica molecular, información sobre la estructura de la sal y el solvente, o de la interfaz entre la molécula y el medio. Aspectos que se pierden al considerar el solvente como un continuo, según lo modela la PBE. Asimismo, las PINNs pueden resolver problemas inversos, siendo relevantes para encontrar constantes dieléctricas locales a partir de mediciones experimentales.

Sin embargo, aunque los métodos basados en PINNs no alcanzan aún la misma precisión que los métodos numéricos tradicionales, se espera que en el futuro se logren avances significativos. El aprendizaje profundo se utiliza en diversas áreas, por lo que existe una gran parte de la comunidad científica investigando el uso de redes neuronales. En esta línea, cualquier innovación en DL podría contribuir al método de PINNs. Además, se espera que en un futuro se avance en enfoques híbridos, combinando PINNs con los métodos tradicionales de resolución. Esto podría potenciar aún más la precisión y eficiencia en la resolución de problemas complejos, como en la electrostática molecular.

En este trabajo se presenta un análisis detallado de cómo el método de Redes Neuronales Informadas por la Física (PINNs) puede aplicarse para resolver la ecuación de Poisson-Boltzmann. Este método se prueba con distintas moléculas, partiendo desde el Ion de Born hasta moléculas con más de 1000 cargas. Además, se llevan a cabo estudios sobre diferentes esquemas de implementación, puntos de colocación y configuraciones de la función de pérdida. De esta forma, se buscará determinar la mejor configuración para resolver este tipo de problemas y evaluar la aplicabilidad de las PINNs en diversos escenarios reales en la electrostática molecular.

Todo el código implementado se ha publicado como una librería, llamada XPPBE (PINNs for PBE) [9], facilitando el uso de PINNs para usuarios externos, generando una herramienta totalmente customizable. La librería de XPPBE contribuye al desarrollo del uso de redes neuronales para modelar la electrostática molecular.

1.1. Objetivo General

En este estudio se busca, a partir del desarrollo de un modelo de PINNs (*Physics Informed Neural Networks*), resolver la ecuación de Poisson-Boltzmann aplicado a macromoléculas reales en medios polarizables. A partir de PINNs se espera obtener el potencial electrostático en la superficie de distintas macromoléculas, junto con sus energías de solvatación.

1.2. Objetivos Específicos

- Evaluar el uso de las distintas formulaciones y esquemas de regularización de la ecuación de Poisson-Boltzmann para su resolución con PINNs.
- Programar distintas arquitecturas de redes neuronales artificiales (ANN) y revisar la sensibilidad respecto a sus hiperparámetros al resolver la PBE.
- Revisar la convergencia y la precisión del método de PINNs al variar el conjunto de puntos de colocación, y al incluir distintos términos a la función de pérdida, considerando mediciones experimentales y/o restricciones físicas adicionales, entre otros.
- Revisar la convergencia y la precisión del método de PINNs al considerar distintas distribuciones de cargas y geometrías moleculares reales, comparando con la solución obtenida por métodos numéricos tradicionales, como BEM y/o FDM.
- Crear un código amigable y totalmente personalizable para el usuario que permita resolver la ecuación de Poisson-Boltzmann utilizando PINNs.

Capítulo 2

Marco Teórico

2.1. Modelo de Solvente Implícito

En la modelación de la electrostática para la solvatación de macromoléculas, se emplean diversos modelos para representar tanto a la macromolécula en cuestión (llamada soluto) como al medio polarizable (solvente). Entre los más representativos se encuentran: el modelo de Solvente Explícito y el modelo de Solvente Implícito [1]. La principal diferencia entre ambos modelos radica en el nivel de detalle con el que se describe el solvente.

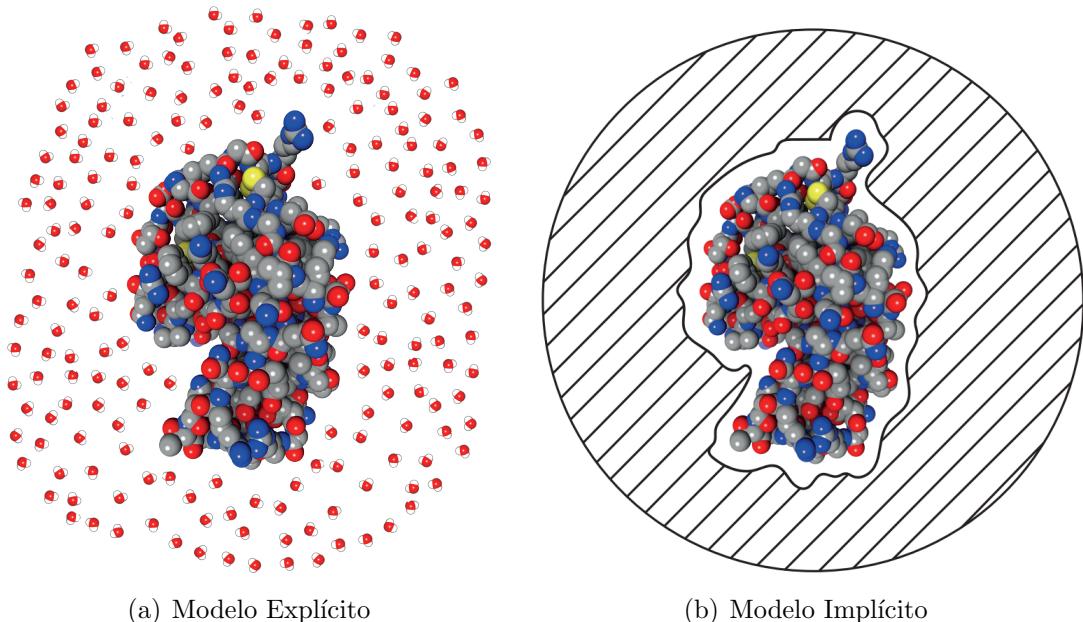


Figura 2.1: Modelos para el solvente

El modelo de Solvente Explícito proporciona una descripción detallada del solvente, abarcando todas las interacciones entre las moléculas que lo conforman. Estas interacciones consideran tanto las relaciones entre las moléculas constituyentes, como las interacciones con la macromolécula inmersa en el solvente. Sin embargo, debido a la necesidad de representar cada molécula con todos sus grados de libertad, este modelo se vuelve altamente costoso en términos computacionales. Un esquema de esta representación se visualiza en la figura 2.1a.

Por otra parte, el modelo de Solvente Implícito representa todo el solvente (incluidas todas las moléculas que lo componen) como un medio continuo polarizable [10], tal como lo muestra la figura 2.1b. A pesar que sea una gran aproximación debido a la naturaleza discreta de la materia, se justifica dado que busca el comportamiento promedio, utilizando las propiedades macroscópicas por sobre las microscópicas. Para este modelo, el solvente se modela con una constante dieléctrica fija, y los iones móviles como una distribución continua de carga. Al utilizar una representación promedio del solvente, los costos computacionales disminuyen bastante comparados con el modelo de Solvente Explícito.

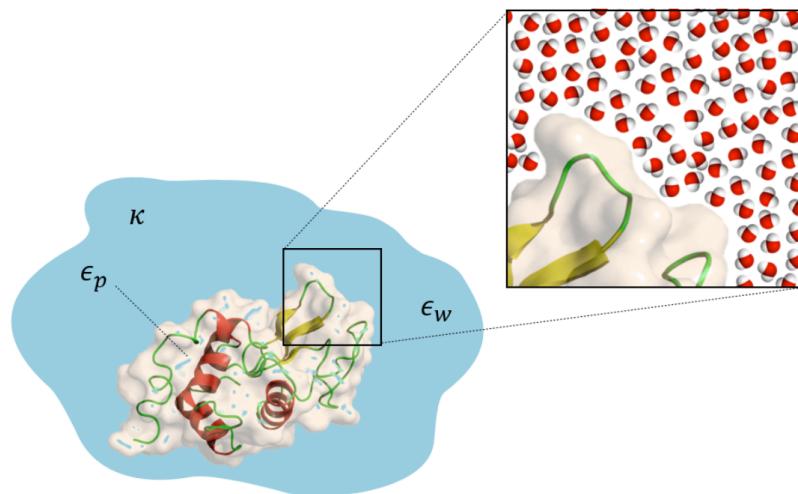


Figura 2.2: Aproximación del modelo de Solvente Implícito

En las próximas secciones se presentarán las ecuaciones que se obtienen al utilizar el modelo de Solvente Implícito, y como estas logran modelar la electrostática molecular.

2.1.1. Electrostática en un Medio Dieléctrico Continuo

Dentro del campo de la electrostática [11], se puede representar la interacción de cargas eléctricas en el vacío con la Ley de Gauss en su forma integral, como se ve a continuación.

$$\oint_{\Gamma} \mathbf{E} \cdot d\mathbf{S} = \frac{Q_{enc}}{\epsilon_0} \quad (2.1)$$

Esta integral relaciona el flujo eléctrico dado por un campo eléctrico \mathbf{E} en una superficie cerrada Γ con carga eléctrica neta encerrada en su interior Q_{enc} , siendo ϵ_0 la constante de permitividad del vacío. Adicionalmente, la ecuación 2.1 se puede escribir en forma diferencial de la siguiente forma.

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \quad (2.2)$$

Donde ρ corresponde a la densidad volumétrica de carga. En ausencias de cambios en campos magnéticos \mathbf{B} , el rotacional del campo eléctrico debe ser cero según las Leyes de Maxwell, es decir:

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} = 0 \quad (2.3)$$

A partir de lo anterior, se puede definir un potencial electrostático ϕ que se relaciona con el campo eléctrico como $\mathbf{E} = -\nabla\phi$. De esta manera, la ecuación 2.2 queda como:

$$\nabla^2 \phi = -\frac{\rho}{\epsilon_0} \quad (2.4)$$

La ecuación 2.4 obtenida es llamada ecuación de Poisson. Al resolver esta ecuación, se puede obtener el potencial electrostático ϕ , y entonces el campo eléctrico \mathbf{E} , para distribuciones de cargas en el vacío.

La solución de la ecuación de Poisson para ϕ en un punto \mathbf{r} puede ser expresada en forma integral como:

$$\phi(\mathbf{r}) = \frac{1}{4\pi\epsilon_0} \int_{\Omega} \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dV' \quad (2.5)$$

Por otra parte, a diferencia de la electrostática en el vacío, al aplicar un campo eléctrico sobre un material dieléctrico, el material se polarizará. Este efecto es representado mediante la polarización eléctrica \mathbf{P} , la cual corresponde a la densidad de momentos dipolares por unidad de volumen en un material dieléctrico.

De esta forma, el potencial eléctrico producto de una polarización eléctrica se puede calcular

como:

$$\tilde{\phi}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0} \int_{\Omega} \frac{\mathbf{P}(\mathbf{r}') \cdot (\mathbf{r} - \mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|^3} dV' \quad (2.6)$$

Integrando por partes, y asumiendo que la única condición de contorno es que la polarización \mathbf{P} está acotada cuando $\mathbf{r}' \rightarrow \infty$ (dieléctrico finito), la integral anterior puede ser escrita de la siguiente forma:

$$\tilde{\phi}(\mathbf{r}) = \frac{1}{4\pi\epsilon_0} \int_{\Omega} \frac{-\nabla' \cdot \mathbf{P}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dV' \quad (2.7)$$

Notar la similitud con la ecuación 2.5. Se nota que la divergencia de la polarización actúa como una densidad de carga. Juntando ambas integrales (potencial producto de distribución de carga y por una polarización) se puede obtener una expresión para el potencial total.

$$\phi(\mathbf{r}) = \frac{1}{4\pi\epsilon_0} \int_{\Omega} \frac{1}{|\mathbf{r} - \mathbf{r}'|} [\rho(\mathbf{r}') - \nabla' \cdot \mathbf{P}(\mathbf{r}')] dV' \quad (2.8)$$

Debido a esto, la divergencia del campo eléctrico \mathbf{E} puede ser escrita como:

$$\nabla \cdot \mathbf{E} = \frac{1}{\epsilon_0} (\rho - \nabla \cdot \mathbf{P}) \quad (2.9)$$

A partir de lo anterior, se puede definir el vector de desplazamiento eléctrico \mathbf{D} como:

$$\mathbf{D} = \epsilon_0 \mathbf{E} + \mathbf{P} \quad (2.10)$$

De esta manera, la ecuación 2.9 queda como:

$$\nabla \cdot \mathbf{D} = \rho \quad (2.11)$$

Esta ecuación relaciona directamente el desplazamiento eléctrico \mathbf{D} producto de una densidad de carga ρ y de una polarización del medio \mathbf{P} . Para un material dieléctrico isotrópico, la polarización eléctrica \mathbf{P} está relacionada de manera lineal con el campo eléctrico \mathbf{E} , la permitividad del vacío ϵ_0 y la susceptibilidad eléctrica del medio χ_e . De esta manera, la polarización se puede escribir como:

$$\mathbf{P} = \epsilon_0 \chi_e \mathbf{E} \quad (2.12)$$

Con esta relación, el desplazamiento eléctrico se puede escribir en función del campo eléctrico.

$$\mathbf{D} = \epsilon_0 (1 + \chi_e) \mathbf{E} = \epsilon \mathbf{E} \quad (2.13)$$

En donde ϵ es la permitividad del medio, y puede calcularse como $\epsilon = \epsilon_0 (1 + \chi_e)$.

Considerando todo lo anterior, se puede modificar la ecuación de Poisson 2.4 para incluir la polarización del medio dieléctrico, incluida en la permitividad del medio ϵ .

$$\nabla^2 \phi = -\frac{\rho}{\epsilon} \quad (2.14)$$

2.1.2. Ecuación de Poisson-Boltzmann

La ecuación de Poisson-Boltzmann es una expresión que permite obtener el potencial electrostático para macromoléculas sumergidas en un solvente polarizable [2, 3, 12]. Para su obtención, se utiliza la ecuación de Poisson para un medio dieléctrico (ecuación 2.14) y el modelo de Solvente Implícito detallado anteriormente en la sección 2.1. El modelo a utilizar detalla una densidad de carga que se descompone en 2 términos: el primero asociado a las cargas fijas (cargas dentro de la macromolécula), y el segundo asociado a la distribución de cargas móviles presentes en el solvente.

$$\rho(\mathbf{r}) = \rho_f(\mathbf{r}) + \rho_m(\mathbf{r}) \quad (2.15)$$

En donde ρ_f y ρ_m hacen referencia a las densidades de cargas fijas y cargas móviles respectivamente. Entonces, en este modelo existirán 2 regiones, soluto y solvente (Ω_1 y Ω_2 respectivamente). Cada región tendrá su propia distribución de carga y permitividad eléctrica ϵ , tal como se muestra en la figura 2.3.

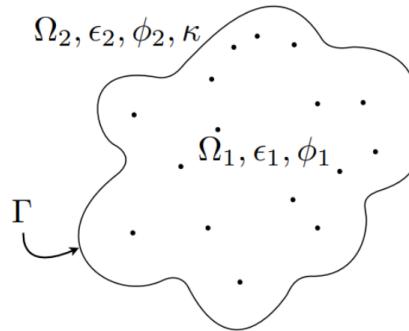


Figura 2.3: Esquema modelo de Solvente Implícito [4]

Con lo anterior, se puede generalizar la ecuación 2.14 que relaciona el potencial electrostático con la densidad de carga para ambos dominios, la cual es llamada ecuación de Poisson Boltzmann.

$$-\nabla \cdot (\epsilon(\mathbf{r}) \nabla \phi) = \rho_f(\mathbf{r}) + \rho_m(\mathbf{r}) \quad (2.16)$$

En esta ecuación se considera ϵ como una función dependiente de la posición (sólido o solvente).

$$\epsilon(\mathbf{r}) = \begin{cases} \epsilon_1 & \mathbf{r} \in \Omega_1 \\ \epsilon_2 & \mathbf{r} \in \Omega_2 \end{cases} \quad (2.17)$$

Para la densidad de carga fija, la cual corresponde a la carga dentro de la macromolécula, se considerarán cargas puntuales. Estas cargas puntuales coinciden con los átomos constituyentes de la macromolécula. De esta manera se obtiene:

$$\rho_f(\mathbf{r}) = \sum_k q_k \delta(\mathbf{r} - \mathbf{r}_k) \quad (2.18)$$

En donde q_k corresponde a la carga del átomo k y $\delta(\mathbf{r})$ corresponde a la función delta de Dirac.

Para la densidad de carga móvil, la cual corresponde a la distribución de cargas del solvente, se utilizará la siguiente expresión:

$$\rho_m(\mathbf{r}) = \sum_j q_j c_j \quad (2.19)$$

En donde q_j corresponde a la carga de la especie j , y c_j su respectiva concentración. En equilibrio, la concentración de las especies sigue una distribución de Boltzmann [1, 13] de la forma:

$$c_j(\mathbf{r}) = c_j^\infty e^{-\phi q_j / k_b T} \quad (2.20)$$

En donde c_j^∞ corresponde a la concentración de iones con carga q_j en el infinito, k_b la constante de Boltzmann, T la temperatura y ϕ el potencial electrostático. La justificación de la utilización de esta distribución se detalla en la sección 2.1.6.

Si el solvente se encuentra disuelto con cloruro de sodio (NaCl), se tienen dos especies de iones (Na^+ y Cl^-) en similar concentración y con cargas de valencias opuestas. De esta manera la densidad de carga en el solvente se puede reducir a lo siguiente:

$$\begin{aligned} \rho_m(\mathbf{r}) &= \sum_j q_j c_j^\infty e^{-\phi q_j / k_b T} \\ \rho_m(\mathbf{r}) &= q_e c^\infty e^{-\phi q_e / k_b T} - q_e c^\infty e^{\phi q_e / k_b T} \\ \rho_m(\mathbf{r}) &= -2q_e c^\infty \sinh(\phi q_e / k_b T) \end{aligned} \quad (2.21)$$

En donde q_e corresponde a la carga eléctrica del electrón. Teniendo ambas densidades de cargas, se puede obtener la ecuación de Poisson-Boltzmann. Considerando ϵ_1 y ϵ_2 a las constantes dieléctricas de cada medio, se obtiene:

$$\begin{aligned}\nabla^2 \phi_1 &= -\frac{1}{\epsilon_1} \sum_k q_k \delta(\mathbf{r} - \mathbf{r}_k) \quad \mathbf{r} \in \Omega_1 \\ \nabla^2 \phi_2 &= \frac{2q_e c^\infty}{\epsilon_2} \sinh(\phi_2 q_e / k_b T) \quad \mathbf{r} \in \Omega_2\end{aligned}\tag{2.22}$$

A estas ecuaciones hay que añadirle la continuidad del potencial y del desplazamiento eléctrico en la interfaz Γ de la macromolécula. La definición de la interfaz en este contexto será definida en la sección 2.1.3.

$$\begin{aligned}\phi_1 &= \phi_2 \quad \mathbf{r} \in \Gamma \\ \epsilon_1 \frac{\partial \phi_1}{\partial \mathbf{n}} &= \epsilon_2 \frac{\partial \phi_2}{\partial \mathbf{n}} \quad \mathbf{r} \in \Gamma\end{aligned}\tag{2.23}$$

2.1.2.1. Ecuación de Poisson Boltzmann Linealizada

La ecuación de Poisson-Boltzmann 2.22 detallada en la sección anterior puede ser linealizada bajo el siguiente supuesto. En caso de que el potencial eléctrico sea débil en el solvente, de tal forma que $\phi_2 q_e / k_b T \ll 1$ (normalmente cuando $|\phi| < 25$ [mV] [14]), la expresión en el solvente puede ser aproximada considerando el primer término de la expansión de la serie de Taylor asociada (teoría de Debye-Hückel).

$$\frac{2q_e c^\infty}{\epsilon_2} \sinh(\phi_2 q_e / k_b T) \approx \frac{2q_e c^\infty \phi q_e}{\epsilon_2 k_b T}\tag{2.24}$$

A partir de esto, se puede definir la constante κ correspondiente al inverso de la longitud de Debye.

$$\kappa^2 = \frac{2c^\infty q_e^2}{\epsilon_2 k_b T}\tag{2.25}$$

De esta manera, la ecuación de Poisson-Boltzmann linealizada queda de la siguiente forma:

$$\begin{aligned}\nabla^2 \phi_1 &= -\frac{1}{\epsilon_1} \sum_k q_k \delta(\mathbf{r} - \mathbf{r}_k) \quad \mathbf{r} \in \Omega_1 \\ \nabla^2 \phi_2 &= \kappa^2 \phi_2 \quad \mathbf{r} \in \Omega_2\end{aligned}\tag{2.26}$$

Esta ecuación linealizada resulta bastante útil en una gran cantidad de escenarios donde se tienen macromoléculas medianamente cargadas (como proteínas), ya que el potencial es débil en el solvente. Además, para estas macromoléculas las no linealidades son relevantes en zonas acotadas del dominio, que dependiendo de la aplicación pueden ser despreciadas. Dicho esto, la forma lineal entrega una buena aproximación permitiendo la utilización de diversos métodos numéricos, facilitando su resolución. En los casos en que se trabajen con macromoléculas altamente cargadas (como ácidos nucleicos), las no linealidades toman mayor importancia.

2.1.3. Interfaz Soluto-Solvente

Para la clasificación de las 2 regiones de interés, soluto y solvente, existen 3 definiciones de superficies. La superficie Gaussiana o de Van der Waals, la superficie accesible por el solvente (SAS), y la superficie excluida por el solvente (SES).

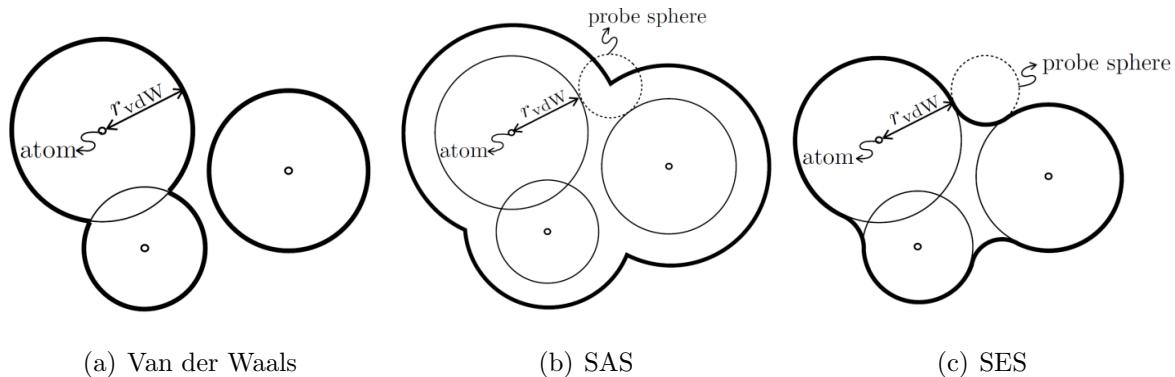


Figura 2.4: Definiciones de interfaz [4]

La superficie de Van der Waals considera como soluto todo lo contenido dentro del radio atómico de los átomos constituyentes, figura 2.4a. Por otra parte, la definición SAS, figura 2.4b, se genera al seguir la trayectoria que tiene el centroide de una probeta esférica del tamaño de una molécula de agua ($\sim 1.4\text{\AA}$) la cual rueda alrededor de la macromolécula. Finalmente, la superficie SES considera como superficie de la macromolécula a la sección más cercana a la que puede llegar la probeta esférica definida anteriormente. En las aplicaciones a detallar en este escrito, se prefiere la utilización de la superficie SES.

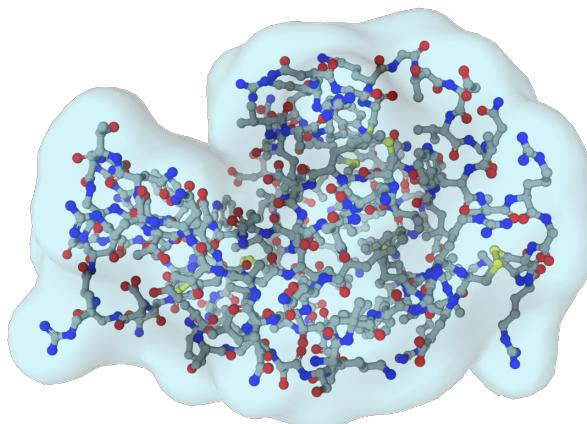


Figura 2.5: Superficie SES de referencia para la lisozima, renderizado con [15]

2.1.4. Esquemas de Regularización de la Ecuación de Poisson-Boltzmann

Debido a las singularidades existentes producto de las cargas puntuales en la macromolécula dadas en la ecuación 2.22 o en la ecuación linealizada 2.26, aparecen bastantes dificultades en la resolución producto de los métodos numéricos utilizados, comúnmente FEM o FDM. Es por esto que en ocasiones se realiza la siguiente transformación para obtener el set de ecuaciones regularizadas. Se detallarán las más simples, pero existen diversos esquemas adicionales [16, 17]. Los esquemas a detallar se basan en descomponer el potencial electrostático en su componente singular y regular (llamado potencial de reacción).

$$\phi = \mathcal{G} + \psi \quad (2.27)$$

En donde ϕ corresponde al potencial total, ψ al potencial de reacción, y \mathcal{G} al potencial de Coulomb (combinación lineal de la función de Green de la ecuación de Poisson). El potencial \mathcal{G} se calcula de la siguiente forma:

$$\mathcal{G}(\mathbf{r}) = \frac{1}{4\pi\epsilon_1} \sum_k \frac{q_k}{|\mathbf{r} - \mathbf{r}_k|} \quad (2.28)$$

Y cumple la siguiente relación:

$$\nabla^2 \mathcal{G}(\mathbf{r}) = -\frac{1}{\epsilon_1} \sum_k q_k \delta(\mathbf{r} - \mathbf{r}_k) \quad (2.29)$$

Se presentan a continuación 2 esquemas de regularización de 2 términos para la ecuación de Poisson-Boltzmann linealizada. Los esquemas son equivalentes para el caso no lineal.

- Esquema 1: Se basa en descomponer el potencial electrostático ϕ en su componente singular y regular en todo el dominio $\Omega = \Omega_1 \cup \Omega_2$.

$$\begin{aligned} \nabla^2 \psi_1 &= 0 & \mathbf{r} \in \Omega_1 \\ \nabla^2 \psi_2 &= \kappa^2 (\psi_2 + \mathcal{G}) & \mathbf{r} \in \Omega_2 \\ \psi_1 &= \psi_2 & \mathbf{r} \in \Gamma \\ \epsilon_1 \left(\frac{\partial \psi_1}{\partial \mathbf{n}} + \frac{\partial \mathcal{G}}{\partial \mathbf{n}} \right) &= \epsilon_2 \left(\frac{\partial \psi_2}{\partial \mathbf{n}} + \frac{\partial \mathcal{G}}{\partial \mathbf{n}} \right) & \mathbf{r} \in \Gamma \end{aligned} \quad (2.30)$$

El sistema de ecuaciones queda en términos del componente regular. Para obtener el potencial total, basta utilizar la ecuación 2.27.

- Esquema 2: Se basa en descomponer el potencial electrostático ϕ en su componente singular y regular solo en la zona del soluto Ω_1 .

$$\begin{aligned}
 \nabla^2 \psi_1 &= 0 & \mathbf{r} \in \Omega_1 \\
 \nabla^2 \phi_2 &= \kappa^2 \phi_2 & \mathbf{r} \in \Omega_2 \\
 \psi_1 + \mathcal{G} &= \phi_2 & \mathbf{r} \in \Gamma \\
 \epsilon_1 \left(\frac{\partial \psi_1}{\partial \mathbf{n}} + \frac{\partial \mathcal{G}}{\partial \mathbf{n}} \right) &= \epsilon_2 \frac{\partial \phi_2}{\partial \mathbf{n}} & \mathbf{r} \in \Gamma
 \end{aligned} \tag{2.31}$$

Notar que el sistema de ecuaciones queda en términos del componente regular en el soluto, y el potencial total en el solvente.

Bajo estos esquemas se evitan las singularidades, aumentando la precisión y la convergencia de los métodos de resolución.

2.1.5. Formulación Integral en la Frontera para la Ecuación de Poisson-Boltzmann

Pese a que se puede resolver directamente la ecuación 2.26 para obtener el potencial electrostático, existen métodos que resuelven su formulación integral en la interfaz. Lo anterior permite resolver la ecuación solo en la superficie de la macromolécula, sin tener que trabajar con todo el dominio, comúnmente utilizando el método de BEM [18, 19].

La formulación integral de la ecuación 2.26 en la interfaz se puede obtener haciendo uso de la Segunda y Tercera Identidad de Green. Considerando que $\mathbf{r}^s \in \Gamma$, se presenta el set de ecuaciones integrales en la interfaz Γ .

$$\begin{aligned}
 \phi_1(\mathbf{r}^s) &= \frac{1}{2\pi\epsilon_1} \sum_k \frac{q_k}{|\mathbf{r}^s - \mathbf{r}_k|} + \frac{1}{2\pi} \oint_{\Gamma} \frac{\partial \phi_1}{\partial \mathbf{n}} \frac{1}{|\mathbf{r}^s - \mathbf{r}'|} dS(\mathbf{r}') \\
 &\quad - \frac{1}{2\pi} \oint_{\Gamma} \phi_1 \frac{\partial}{\partial \mathbf{n}} \left(\frac{1}{|\mathbf{r}^s - \mathbf{r}'|} \right) dS(\mathbf{r}') \\
 \phi_2(\mathbf{r}^s) &= -\frac{1}{2\pi} \oint_{\Gamma} \frac{\partial \phi_2}{\partial \mathbf{n}} \frac{\exp(-\kappa|\mathbf{r}^s - \mathbf{r}'|)}{|\mathbf{r}^s - \mathbf{r}'|} dS(\mathbf{r}') \\
 &\quad + \frac{1}{2\pi} \oint_{\Gamma} \phi_2 \frac{\partial}{\partial \mathbf{n}} \left(\frac{\exp(-\kappa|\mathbf{r}^s - \mathbf{r}'|)}{|\mathbf{r}^s - \mathbf{r}'|} \right) dS(\mathbf{r}')
 \end{aligned} \tag{2.32}$$

Utilizando ambas integrales y las condiciones en la interfaz, se puede formar un sistema de ecuaciones de 2 variables (ϕ y $\partial_n \phi$) para obtener el potencial electrostático en la interfaz. Al

haber resuelto este sistema de ecuaciones en la interfaz, se puede obtener el potencial en todo el dominio con una modificación de estas integrales. Por ejemplo, si $\mathbf{r} \in \Omega_1$, se tiene:

$$\begin{aligned}\phi_1(\mathbf{r}) = & \frac{1}{4\pi\epsilon_1} \sum_k \frac{q_k}{|\mathbf{r} - \mathbf{r}_k|} + \frac{1}{4\pi} \oint_{\Gamma} \frac{\partial \phi_1}{\partial \mathbf{n}} \frac{1}{|\mathbf{r} - \mathbf{r}'|} dS(\mathbf{r}') \\ & - \frac{1}{4\pi} \oint_{\Gamma} \phi_1 \frac{\partial}{\partial \mathbf{n}} \left(\frac{1}{|\mathbf{r} - \mathbf{r}'|} \right) dS(\mathbf{r}')\end{aligned}\quad (2.33)$$

Asimismo, si $\mathbf{r} \in \Omega_2$, se tiene:

$$\begin{aligned}\phi_2(\mathbf{r}) = & -\frac{1}{4\pi} \oint_{\Gamma} \frac{\partial \phi_2}{\partial \mathbf{n}} \frac{\exp(-\kappa|\mathbf{r} - \mathbf{r}'|)}{|\mathbf{r} - \mathbf{r}'|} dS(\mathbf{r}') \\ & + \frac{1}{4\pi} \oint_{\Gamma} \phi_2 \frac{\partial}{\partial \mathbf{n}} \left(\frac{\exp(-\kappa|\mathbf{r} - \mathbf{r}'|)}{|\mathbf{r} - \mathbf{r}'|} \right) dS(\mathbf{r}')\end{aligned}\quad (2.34)$$

2.1.6. Energías Libres

La energía libre total de una macromolécula sumergida en un medio polarizable tiene componentes polares (o electrostáticos) y no polares. La relación entre las distintas energías libres se puede revisar en el siguiente ciclo termodinámico.

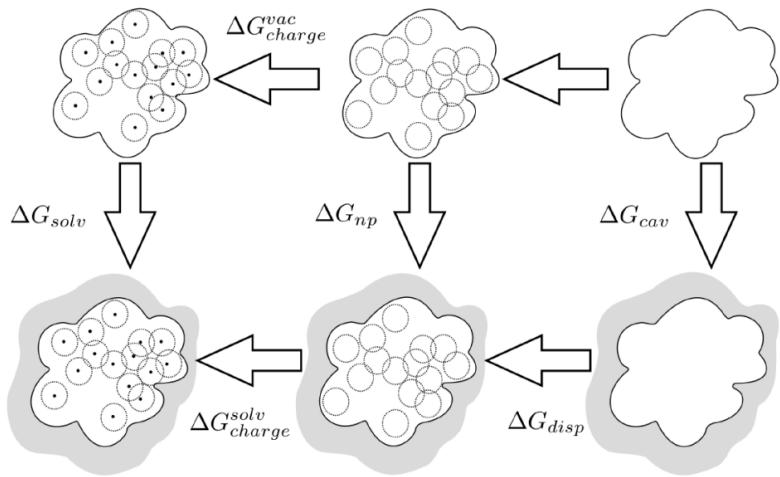


Figura 2.6: Ciclo termodinámico para las energías libres [20]

En las siguientes secciones se desarrollará la energía libre no polar, la energía libre electrostática y la energía de solvatación.

2.1.6.1. Energía Libre No Polar

La energía libre no polar corresponde a la energía necesaria para situar un soluto no cargado en el solvente. Debido a lo anterior, esta energía considera tanto la energía para generar la cavidad ΔG_{cavity} , como la energía producto de la interacción de dispersión entre el soluto y el solvente ΔG_{disp} .

$$\Delta G_{np} = \Delta G_{cavity} + \Delta G_{disp} \quad (2.35)$$

Siendo ΔG_{np} la energía libre no polar. Los modelos actuales consideran la energía no polar como una combinación lineal entre el área de la macromolécula accesible por el solvente A_{SASA} (superficie SAS) y otros parámetros geométricos [20].

$$\Delta G_{np} = \gamma A_{SASA} + b \quad (2.36)$$

En donde γ se interpreta como la tensión superficial de líquido-vapor, y b como una constante de escalamiento (usualmente presentada como pV , presión por volumen).

2.1.6.2. Energía Libre Electrostática

En el contexto del solvente implícito, se tiene el siguiente funcional para la energía libre de Gibbs G de un soluto disuelto en un solvente. Este funcional solo depende de las concentraciones de los iones c_i del solvente [21].

$$G[c_1, c_2, \dots, c_M] = \int_{\Omega} \frac{1}{2} \rho \phi + \beta^{-1} \sum_j^M c_j^\infty + \beta^{-1} \sum_j^M c_j [\ln(\Lambda^3 c_j) - 1] - \sum_j^M \mu_j c_j \, dV \quad (2.37)$$

En donde ϕ es el potencial electrostático, ρ la densidad de carga, β el inverso de la energía térmica (constante de Boltzmann por la temperatura), Λ la longitud de onda térmica de De Broglie, μ_j el potencial químico de la especie j y c_j^∞ la concentración de la especie j al infinito.

Para obtener la energía libre, se debe encontrar una distribución de concentraciones c_j que minimize el funcional $G[c_1, c_2, \dots, c_M]$. Para esto, se calcula el variacional de la ecuación 2.37 con respecto a la concentración c_j de cada ion.

$$(\delta G[c])_j = q_j \phi + \beta^{-1} \ln(\Lambda^3 c_j) - \mu_j = 0 \quad (2.38)$$

Despejando la variable de interés, se obtiene que las concentraciones deben seguir una distribución de Boltzmann para que el funcional G sea mínimo.

$$c_j = c_j^\infty e^{-\beta q_j \phi} \quad (2.39)$$

En donde c_j^∞ se puede expresar como:

$$c_j^\infty = \Lambda^{-3} e^{\beta \mu_j} \quad (2.40)$$

De esta forma, reemplazando la distribución de Boltzmann en el funcional 2.37 se obtiene la energía libre electrostática.

$$G = \int_{\Omega} \rho_f \phi - \frac{1}{2} \epsilon |\nabla \phi|^2 - \beta^{-1} \sum_j^M c_j^\infty (e^{-\beta q_j \phi} - 1) dV \quad (2.41)$$

La energía libre anterior se puede simplificar para el caso de una solución de NaCl, obteniéndose lo siguiente:

$$G = \int_{\Omega} \rho_f \phi - \frac{1}{2} \epsilon |\nabla \phi|^2 - 2\beta^{-1} c^\infty (\cosh(\phi \beta q_e) - 1) dV \quad (2.42)$$

Notar que al reemplazar la energía libre de Gibbs por unidad de volumen g en las ecuaciones de Euler Lagrange, se debería obtener una ecuación que gobierne la situación.

$$\frac{\partial g}{\partial \phi} - \nabla \cdot \left(\frac{\partial g}{\partial (\nabla \phi)} \right) = 0 \quad (2.43)$$

De donde se obtiene claramente la ecuación de Poisson-Boltzmann.

$$\nabla \cdot (\epsilon \nabla \phi) = -\rho_f + 2q_e c^\infty \sinh(\phi \beta q_e) \quad (2.44)$$

En la expresión 2.42 se puede reemplazar la ecuación de Poisson Boltzmann e integrar por partes para obtener una versión simplificada de la energía libre electrostática.

$$G = \int_{\Omega_1} \frac{1}{2} \rho_f \phi dV + \int_{\Omega_2} \phi q_e c^\infty \sinh(\phi \beta q_e) - 2\beta^{-1} c^\infty (\cosh(\phi \beta q_e) - 1) dV \quad (2.45)$$

Si se está trabajando con la ecuación de Poisson-Boltzmann lineal (ver sección 2.1.2.1), se puede simplificar la integral en el solvente al linealizar su integrando (útil cuando $\phi_2 q_e / k_b T \ll 1$), para obtener la energía libre del sistema.

$$G = \frac{1}{2} \int_{\Omega_1} \rho_f \phi dV \quad (2.46)$$

2.1.6.3. Energía Libre de Coulomb

La energía libre de Coulomb tiene relación a la distribución de cargas del soluto en el vacío (en este caso, las cargas puntuales de la macromolécula). Esta energía se puede obtener al reemplazar el potencial de Coulomb \mathcal{G} (el cual es solución de la ecuación de Poisson), en la ecuación 2.46 para la energía libre electrostática.

$$\mathcal{G}(\mathbf{r}) = \frac{1}{4\pi\epsilon_1} \sum_k \frac{q_k}{|\mathbf{r} - \mathbf{r}_k|} \quad (2.47)$$

De esta forma, la energía libre de Coulomb $G_{coulomb}$ se expresa como:

$$G_{coulomb} = \frac{1}{8\pi\epsilon_1} \sum_i \sum_j \frac{q_i q_j}{|\mathbf{r}_i - \mathbf{r}_j|} \quad (2.48)$$

2.1.6.4. Energía de Solvatación

La energía de solvatación tiene relación con la interacción entre el soluto y el solvente, en donde se estabilizan las partículas del medio acuoso. Esta se define como el cambio en la energía libre de Gibbs cuando una molécula es transferida desde el vacío al solvente. De manera equivalente, la energía de solvatación se puede entender como la energía que se requiere para disolver una molécula.

Revisando el ciclo termodinámico dado en la figura 2.6, se nota que la energía de solvatación tiene 2 componentes:

$$\Delta G_{solv} = \Delta G_{elec} + \Delta G_{np} \quad (2.49)$$

Donde ΔG_{solv} corresponde a la energía de solvatación, ΔG_{elec} al componente electrostático y ΔG_{np} al componente no polar. Una aproximación tolerable es despreciar el término no polar, de tal manera que la energía de solvatación depende exclusivamente del componente electrostático. Notar que debido al ciclo de la figura 2.6, el componente electrostático corresponde a la diferencia entre la energía electrostática total y la energía de Coulomb.

$$\Delta G_{elec} = G_{charge}^{solv} - G_{coulomb} \quad (2.50)$$

Siendo G_{charge}^{solv} la energía electrostática total y $G_{coulomb}$ la energía de Coulomb. Bajo esta línea, la energía de solvatación ΔG_{solv} puede ser calculada integrando el potencial de reacción ψ y la distribución de carga ρ_f en la región del soluto, más una expresión no lineal dependiente del

potencial total ϕ en el solvente (ecuación 2.45).

$$\Delta G_{solv} = \Delta G_{elec} + \cancel{\Delta G_{np}}^0 \\ \Delta G_{solv} = \frac{1}{2} \int_{\Omega_1} \rho_f \psi \, dV + \int_{\Omega_2} \phi q_e c^\infty \sinh(\phi \beta q_e) - 2\beta^{-1} c^\infty (\cosh(\phi \beta q_e) - 1) \, dV \quad (2.51)$$

En donde el potencial de reacción se calcula como la diferencia entre el potencial total y el potencial de Coulomb:

$$\psi = \phi - \mathcal{G} \quad (2.52)$$

Debido a que la densidad de cargas fijas ρ_f se compone por cargas puntuales q_k , la integral en el soluto puede ser simplificada a la siguiente expresión:

$$\Delta G_{solv} = \frac{1}{2} \sum_k q_k \psi(\mathbf{r}_k) + \int_{\Omega_2} \phi q_e c^\infty \sinh(\phi \beta q_e) - 2\beta^{-1} c^\infty (\cosh(\phi \beta q_e) - 1) \, dV \quad (2.53)$$

En los casos que se trabaje con la forma lineal de la ecuación de Poisson-Boltzmann, la energía de solvatación se reduce a lo siguiente:

$$\Delta G_{solv} = \frac{1}{2} \sum_k q_k \psi(\mathbf{r}_k) \quad (2.54)$$

2.1.7. Mediciones Experimentales

Con la tecnología que existe actualmente, no es posible medir puntualmente el potencial eléctrico en todo el dominio. Sin embargo, con resonancia magnética nuclear, se puede medir un potencial “promedio” alrededor de los átomos de hidrógeno que estén cerca de la superficie de la macromolécula. Este potencial promedio es llamado *effective near surface potential*, ϕ_{ENS} [22].

El método para obtener este potencial consiste en utilizar derivados del compuesto PROXYL: específicamente, amino-metil-PROXYL y carboxi-PROXYL. Estos compuestos paramagnéticos son análogos pero difieren en carga a pH neutro: el amino-metil-PROXYL es catiónico ($+1q_e$), mientras que el carboxi-PROXYL es aniónico ($-1q_e$).

Para la relajación por interacción electrónica paramagnética (PRE) que surge de moléculas paramagnéticas de cosoluto, la tasa de PRE para una magnetización transversal de 1H (Γ_2), a partir de un potencial ϕ , se expresa de la siguiente manera.

$$\Gamma_2 = 4\pi\zeta c_p \tau_c \int_0^\infty r^{-4} e^{-\frac{q_e \phi(r)}{k_b T}} dr \quad (2.55)$$

Donde ζ es un parámetro dependiente de los núcleos de hidrógeno respectivos de una macromolécula y los ratios giromagnéticos, c_p es la concentración paramagnética de la sonda (PROXYL), τ_c es la correlación efectiva de tiempo para la interacción dipolo-dipolo entre el electrón de la sonda y el núcleo de hidrógeno, r es la distancia entre el núcleo de hidrógeno y el punto de evaluación, ϕ el potencial electrostático, k_b la constante de Boltzmann y T la temperatura. Esta integral puede ser transformada a una integral de volumen de la siguiente manera.

$$\Gamma_2 = 4\pi\zeta c_p \tau_c \int_{\Omega^*} r^{-6} e^{-\frac{q_e \phi(\mathbf{r})}{k_b T}} dV \quad (2.56)$$

Con esta definición, el *effective near surface potential* puede calcularse como:

$$\phi_{ENS}(\mathbf{r}_H) = \frac{-k_b T}{2q_e} \ln(\Gamma_2 + / \Gamma_2 -) \quad (2.57)$$

Donde el signo \pm hace referencia a las 2 sondas a utilizar. Se considera \mathbf{r}_H como la posición del átomo de hidrógeno para el cual se calcula el potencial promedio. Reemplazando, se obtiene finalmente que ϕ_{ENS} se puede calcular de la siguiente manera:

$$\phi_{ENS}(\mathbf{r}_H) = \frac{-k_b T}{2q_e} \ln \left(\frac{\int_{\Omega^*} |\mathbf{r} - \mathbf{r}_h|^{-6} e^{-\frac{q_e \phi(\mathbf{r})}{k_b T}} dV}{\int_{\Omega^*} |\mathbf{r} - \mathbf{r}_h|^{-6} e^{\frac{q_e \phi(\mathbf{r})}{k_b T}} dV} \right) \quad (2.58)$$

La teoría producto de la ecuación de Poisson-Boltzmann ha predicho valores de ϕ_{ENS} a partir de la ecuación 2.58 bastante similares a los medidos experimentalmente. Un ejemplo para la ubicuitina se encuentra en [22].

2.1.8. Soluciones Analíticas

En esta sección se presentarán algunas soluciones analíticas de la ecuación de Poisson-Boltzmann bajo ciertas suposiciones y escenarios [12]. En todos los casos se trabajará con la ecuación de Poisson-Boltzmann linealizada, ecuación 2.26.

2.1.8.1. Ion de Born

Considera solo una carga puntual q en el centro de una macromolécula esférica de radio R (llamada *Ion de Born*), como lo muestra la siguiente imagen.

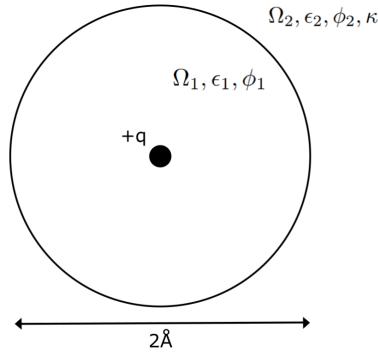


Figura 2.7: Ion de Born

Debido a la simetría esférica existente en este escenario, se puede asumir que $\phi = \phi(r)$, es decir, el potencial solo depende de la coordenada radial. De esta manera, la ecuación de Poisson-Boltzmann se simplifica a las siguientes expresiones:

$$\begin{aligned} \frac{1}{r^2} \frac{d}{dr} \left(r^2 \frac{d\phi_1}{dr} \right) &= -\frac{1}{\epsilon_1} q \delta(r) & \mathbf{r} \in \Omega_1 \\ \frac{1}{r^2} \frac{d}{dr} \left(r^2 \frac{d\phi_2}{dr} \right) &= \kappa^2 \phi_2 & \mathbf{r} \in \Omega_2 \\ \phi_1 &= \phi_2 & \mathbf{r} \in \Gamma \\ \epsilon_1 \frac{d\phi_1}{dr} &= \epsilon_2 \frac{d\phi_2}{dr} & \mathbf{r} \in \Gamma \end{aligned} \quad (2.59)$$

Resolviendo este sistema de ecuaciones, se obtiene la siguiente solución para el potencial electrostático.

$$\begin{aligned} \phi_1(r) &= \frac{q}{4\pi} \left(\frac{1}{\epsilon_1 r} - \frac{1}{\epsilon_1 R} + \frac{1}{\epsilon_2 (1 + \kappa R) R} \right) \\ \phi_2(r) &= \frac{q}{4\pi} \frac{e^{-\kappa(r-R)}}{\epsilon_2 (1 + \kappa R) r} \end{aligned} \quad (2.60)$$

Además, se puede obtener fácilmente una expresión para su energía de solvatación:

$$\Delta G_{solv} = \frac{q^2}{8\pi R \epsilon_1} \left(\frac{\epsilon_1}{\epsilon_2 (1 + \kappa R)} - 1 \right) \quad (2.61)$$

Estas expresiones son de utilidad para validar los métodos para resolver la ecuación en cuestión.

2.1.8.2. Multipolo Esférico

Esta solución analítica busca una solución para moléculas con una geometría esférica. Pese a que en la realidad no existan moléculas de este estilo, esta solución puede entregar órdenes de magnitud útiles de acuerdo a la aplicación.

Para esto, se considera una molécula esférica de radio R con N cargas puntuales. Para obtener esta solución, se puede aproximar el potencial electrostático como una combinación lineal de sus armónicos esféricos [23]. De esta manera, la solución de la ecuación de Poisson-Boltzmann linealizada queda de la siguiente forma:

$$\begin{aligned}\phi_1(\mathbf{r}) &= \frac{1}{4\pi\epsilon_1} \sum_k \frac{q_k}{|\mathbf{r} - \mathbf{r}_k|} + \sum_{n=0}^{\infty} \sum_{m=-n}^n B_{nm} r^n Y_n^m(\theta, \varphi) \\ \phi_2(\mathbf{r}) &= \sum_{n=0}^{\infty} \sum_{m=-n}^n A_{nm} \frac{1}{r^{n+1}} e^{-\kappa r} K_n(\kappa r) Y_n^m(\theta, \varphi)\end{aligned}\quad (2.62)$$

Siendo B_{nm} y A_{nm} los coeficientes asociados. Además, cada armónico esférico Y_n^m se representa de la siguiente manera:

$$Y_n^m(\theta, \varphi) = \sqrt{\frac{(2n+1)(n-|m|)!}{4\pi(n+|m|)!}} P_n^m(\cos(\varphi)) e^{im\theta} \quad (2.63)$$

Siendo P_n^m las funciones de Legendre asociadas. Además, las funciones $K_n(x)$ corresponden a los siguientes polinomios.

$$K_n(x) = \sum_{s=0}^n \frac{2^s n! (2n-s)!}{s!(2n)!(n-s)!} x^s \quad (2.64)$$

Para encontrar los coeficientes B_{nm} y A_{nm} , necesarios para estimar el potencial electrostático en todo el dominio, se puede realizar lo siguiente:

Se escribirá el potencial de Coulomb como una combinación lineal de armónicos esféricos.

$$\frac{1}{4\pi\epsilon_1} \sum_k \frac{q_k}{|\mathbf{r} - \mathbf{r}_k|} = \frac{1}{4\pi\epsilon_1} \sum_{n=0}^{\infty} \sum_{m=-n}^n E_{nm} \frac{1}{r^{n+1}} Y_n^m(\theta, \varphi) \quad (2.65)$$

Donde el coeficiente E_{nm} es conocido y se calcula como:

$$E_{nm} = \frac{4\pi}{2n+1} \sum_{k=1}^N q_k r_k^n Y_n^m(-\theta_k, \varphi_k) \quad (2.66)$$

Ahora, se puede plantear un sistema de ecuaciones en la interfaz de la molécula para poder obtener B_{nm} y A_{nm} utilizando las relaciones dadas en la ecuación 2.23.

$$\begin{aligned} \frac{E_{nm}}{4\pi\epsilon_1} + R^{2n+1}B_{nm} &= A_{nm}e^{-\kappa R}K_n(\kappa R) \\ \frac{n+1}{4\pi}E_{nm} - n\epsilon_1 R^{2n+1}B_{nm} &= \epsilon_2 A_{nm}e^{-\kappa R}[(n+1+\kappa R)K_n(\kappa R) - \kappa R K'_n(\kappa R)] \end{aligned} \quad (2.67)$$

Resolviendo este sistema se puede encontrar B_{nm} y A_{nm} y con esto una expresión para el potencial electrostático en todo el dominio. Por simplicidad, se expresarán ambos coeficientes en función de E_{nm} ya que es conocido.

$$\begin{aligned} A_{nm} &= E_{nm} \frac{1}{4\pi} \frac{(2n+1)e^{\kappa R}}{(\epsilon_1 - \epsilon_2)nK_n(\kappa R) + \epsilon_2(2n+1)K_{n+1}(\kappa R)} \\ B_{nm} &= \frac{1}{R^{2n+1}} \left(e^{-\kappa R} K_n(\kappa R) A_{nm} - \frac{1}{4\pi\epsilon_1} E_{nm} \right) \end{aligned} \quad (2.68)$$

Con esta solución, la energía de solvatación puede calcularse fácilmente como:

$$\Delta G_{solv} = \frac{1}{2} \sum_k \sum_{n=0}^{\infty} \sum_{m=-n}^n B_{nm} q_k r_k^n Y_n^m(\theta_k, \varphi_k) \quad (2.69)$$

Al igual que el caso del Ion de Born, esta solución analítica es útil para validar los métodos de resolución.

2.1.8.3. Cargas Puntuales Inmersas en Solvente

Una solución analítica útil para la ecuación de Poisson-Boltzmann es asumir que las cargas puntuales están inmersas en el solvente. De esta forma, no existe una región para el soluto, todo el dominio es solvente con una constante dieléctrica ϵ_2 . De esta forma, la ecuación 2.26 se simplifica a lo siguiente:

$$\nabla^2 \phi - \kappa^2 \phi = -\frac{1}{\epsilon_2} \sum_k q_k \delta(\mathbf{r} - \mathbf{r}_k) \quad (2.70)$$

Aprovechando esta simplificación, se puede encontrar la siguiente solución analítica, la cual corresponde a una combinación lineal de la función de Green de Yukawa¹:

$$\phi(\mathbf{r}) = \frac{1}{4\pi\epsilon_2} \sum_k \frac{q_k e^{-\kappa|\mathbf{r}-\mathbf{r}_k|}}{|\mathbf{r}-\mathbf{r}_k|} \quad (2.71)$$

Pese a que esta aproximación no considera la existencia de la geometría molecular (solo sus cargas puntuales), predice bastante bien la solución de la ecuación de Poisson-Boltzmann cuando se está en el solvente, lejos de la molécula. Incluso en algunos casos esta solución es utilizada como condición de contorno en el solvente [24].

¹Por simplicidad, a esta solución se le llamará función de Green de Yukawa

2.2. Redes Neuronales Artificiales

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 $\boldsymbol{\theta}$. La forma de estas funciones y la cantidad de parámetros en el conjunto $\boldsymbol{\theta}$ dependen exclusivamente de la arquitectura de la red neuronal [5, 25, 26].

Como ejemplo, se asumirá que \mathbf{u}_θ es la aproximación mediante una red neuronal \mathcal{N} con parámetros $\boldsymbol{\theta}$, de la función \mathbf{u} .

$$\mathbf{u} \approx \mathbf{u}_\theta = \mathcal{N}(\mathbf{x}; \boldsymbol{\theta}) \quad (2.72)$$

De esta forma, la red \mathcal{N} consistirá en una composición de funciones (o capas) f_θ^i que operan sobre el input \mathbf{x} de la siguiente forma:

$$\mathbf{u}_\theta = f_\theta^{L+1} \circ f_\theta^L \circ \dots \circ f_\theta^1(\mathbf{x}) \quad (2.73)$$

La arquitectura de la red neuronal determinará la forma de las capas f_θ^i .

Para que una ANN reproduzca el resultado esperado, es necesario entrenarla bajo un proceso de minimización. Para esto, se plantea una función de pérdida \mathcal{L} (comúnmente como error cuadrático medio) que depende de los inputs y los parámetros $\boldsymbol{\theta}$ 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 $\boldsymbol{\theta}^*$ óptimos:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}) \quad (2.74)$$

Normalmente, este problema de minimización es llevado a cabo con métodos de optimización de descenso de gradiente. La forma de la función de pérdida \mathcal{L} va a depender del contexto del problema y su aplicación. Es importante destacar 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 [27], implica que puede ser utilizado en un gran abanico de contextos.

2.2.1. Arquitecturas

En esta sección se detallarán 4 de las arquitecturas más utilizadas en el aprendizaje profundo.

2.2.1.1. Multi Layer Perceptron

El perceptrón consiste en una neurona artificial en donde se genera una operación no lineal. De esta forma, se procesa una señal de entrada \mathbf{x} para obtener una señal de salida y .

$$y = \sigma \left(\sum_{j=1}^m w_j x_j + b \right) \quad (2.75)$$

En donde σ corresponde a una función de activación no lineal, w_j a los pesos de las conexiones a la entrada y b un bias. Un esquema se puede revisar en la figura 2.8.

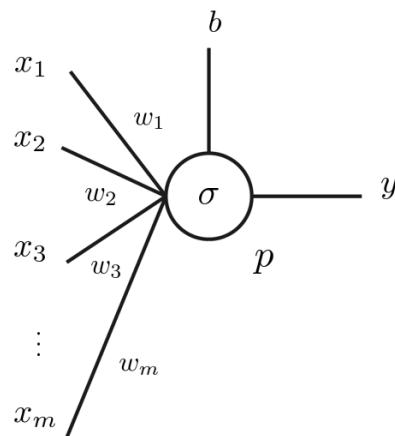


Figura 2.8: Esquema del perceptrón

Dicho lo anterior, la arquitectura MLP, *Multi Layer Perceptron*, o *Fully Connected Neural Network*, se basa en conectar $L + 1$ capas de m_i perceptrones cada una, de manera que todos los perceptrones de cada capa quedan conectados con todos los de la capa anterior. De esta forma, en cada perceptrón p de la capa i , ocurre la siguiente transformación:

$$h_p^i = \sigma_p^i \left(\sum_{j=1}^{m_i} w_{pj}^i h_j^{i-1} + b_p^i \right) \quad (2.76)$$

En donde w_{pj}^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 σ_p^i una función de activación no lineal asociada al perceptrón.

Para simplificar la notación anterior, se trabajará con la matriz de pesos \mathbf{W}^i y el vector de bias \mathbf{b}^i de cada capa i . De esta forma, para las $L + 1$ capas de perceptrones, se podrá formar el conjunto $\boldsymbol{\theta}$ de parámetros como:

$$\boldsymbol{\theta} = \{\mathbf{W}^1, \mathbf{b}^1, \dots, \mathbf{W}^{L+1}, \mathbf{b}^{L+1}\} \quad (2.77)$$

Un esquema de la arquitectura MLP se puede revisar en la figura 2.9.

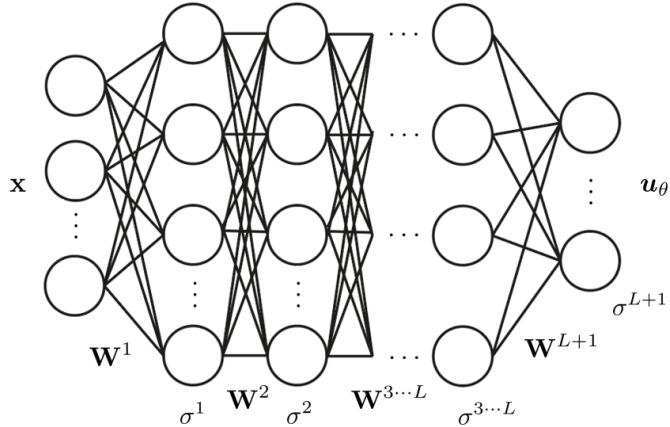


Figura 2.9: Esquema de MLP

En la imagen anterior, la red tiene como input el vector \mathbf{x} y como output el vector \mathbf{u}_θ .

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

$$\begin{aligned} \mathbf{h}^1 &= \sigma^1(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) & i = 1 \text{ capa entrada} \\ \mathbf{h}^i &= \sigma^i(\mathbf{W}^i \mathbf{h}^{i-1} + \mathbf{b}^i) & 2 \leq i \leq L \text{ capas ocultas} \\ \mathbf{u}_\theta &= \sigma^{L+1}(\mathbf{W}^{L+1} \mathbf{h}^L + \mathbf{b}^{L+1}) & i = L + 1 \text{ capa salida} \end{aligned} \quad (2.78)$$

Esta arquitectura al ser bastante simple, es utilizada en la mayoría de las aplicaciones. Además, en algunas aplicaciones se utilizan arquitecturas más complejas incluyendo capas de perceptrones conectados entre si. Estas capas normalmente se llaman *Fully Connected Layers*.

2.2.1.2. Residual Neural Network

La arquitectura ResNet (*Residual Neural Network*) [28] nace dado que trabajar con un número elevado de capas ocultas genera problemas en el entrenamiento. 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. 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 evitar este fenómeno, esta arquitectura añade conexiones residuales, lo que permite traspasar la información directamente entre capa y capa. Un esquema de lo anterior se visualiza en la figura 2.10.

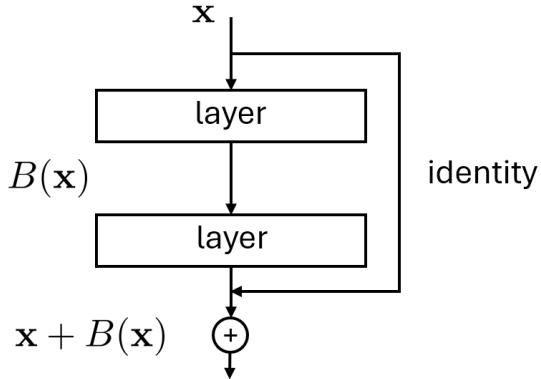


Figura 2.10: Esquema de ResNet para un bloque oculto B

A modo de ejemplo, se detallará como aplicar estas conexiones residuales a la arquitectura MLP definida en la sección anterior. En este caso, la ResNet considerará 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 de la siguiente forma.

$$\begin{aligned}
 \mathbf{h}^0 &= \sigma^0(\mathbf{W}^0 \mathbf{x} + \mathbf{b}^0) & i = 0 & \text{capa entrada} \\
 \mathbf{g}^i &= \sigma_1^i(\mathbf{W}_1^i \mathbf{h}^{i-1} + \mathbf{b}_1^i) & 1 \leq i \leq L & \text{bloques ocultos} \\
 \mathbf{h}^i &= \mathbf{h}^{i-1} + \sigma_2^i(\mathbf{W}_2^i \mathbf{g}^i + \mathbf{b}_2^i) \\
 \mathbf{u}_\theta &= \sigma^{L+1}(\mathbf{W}^{L+1} \mathbf{h}^L + \mathbf{b}^{L+1}) & i = L + 1 & \text{capa salida}
 \end{aligned} \tag{2.79}$$

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

en la solución.

2.2.1.3. Convolutional Neural Network

Las redes neuronales convolucionales (CNN) operan sobre datos estructurados en mallas, como imágenes. Para operar sobre estos datos, se utilizan capas de convolución y *pooling*. Las capas de convolución están compuestas por filtros (*kernels*) a los cuales se les aplica una convolución con los datos de entrada. Luego, las capas de *pooling* generan un muestreo del output para reducir su dimensión. Un ejemplo de las operaciones que ocurren en la capa i -ésima de una CNN se visualiza a continuación:

$$\mathbf{H}^i = \Phi^i(\sigma^i(\mathbf{K}^i * \mathbf{H}^{i-1})) \quad (2.80)$$

En esta ecuación, \mathbf{K}^i corresponde al filtro de la capa (que contiene parámetros entrenables pertenecientes al conjunto θ), σ^i a la función de activación no lineal, Φ^i al operador de *pooling* y $*$ el operador de convolución. De esta manera, se procesa un input \mathbf{H}^{i-1} para obtener un output \mathbf{H}^i . Adicional a las capas de convolución, es común encontrar capas de perceptrones a la salida de la red neuronal.

2.2.1.4. Recurrent Neural Network

Las redes neuronales recurrentes (RNN) están diseñadas para datos secuenciales como texto, audio o series temporales. Su ventaja es que logran guardar un estado interno que depende de todos los pasos anteriores. De esta manera, esa información es utilizada para el siguiente paso temporal, sin pérdida de información. Para lograr esto, las RNN tienen conexiones recurrentes que les permiten mantener y actualizar un estado interno a medida que procesan la secuencia. Un ejemplo de la capa i -ésima de una red recurrente se visualiza a continuación:

$$\mathbf{h}_t^i = \sigma^i(\mathbf{W}^i \mathbf{h}_t^{i-1} + \mathbf{b}^i + \mathbf{U} \mathbf{h}_{t-1}^i) \quad (2.81)$$

De esta forma, se obtiene un output \mathbf{h}_t^i a partir de un input \mathbf{h}_t^{i-1} para un paso temporal t , siendo \mathbf{W}^i la matriz de pesos de la conexión, \mathbf{b}^i al vector de bias y σ^i la función de activación no lineal. Notar además que el output para el paso temporal t depende del output del paso temporal anterior \mathbf{h}_{t-1}^i , el cual es transformado con la matriz de pesos \mathbf{U} . En este caso, las matrices \mathbf{W}^i y \mathbf{U} , junto al vector \mathbf{b}^i , pertenecerán al conjunto de parámetros θ .

2.2.1.5. Funciones de Activación

En las secciones anteriores se detalló que es necesario agregar funciones de activación en los perceptrones u operaciones para inducir una no linealidad, y con eso una representación de la solución. Las funciones de activación más utilizadas en las ANN se detallan a continuación:

- **Sigmoid:**

- Fórmula: $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$
- Recorrido: $]0, 1[$
- Características: Transforma cualquier valor de entrada en un valor entre 0 y 1.

- **ReLU (Rectified Linear Unit):**

- Fórmula: $\text{ReLU}(x) = \max(0, x)$
- Recorrido: $[0, +\infty[$
- Características: Mantiene activadas las neuronas con valores positivos y desactiva las neuronas con valores negativos.

- **Tangente Hiperbólica:**

- Fórmula: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Recorrido: $]-1, 1[$
- Características: Transforma cualquier valor de entrada en un valor entre -1 y 1.

- **Softmax:**

- Fórmula: $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$
- Recorrido: $]0, 1]$ (sumando 1)
- Características: Normaliza la salida de manera que la suma de todas las salidas sea igual a 1, lo que la hace útil para problemas de clasificación.

La elección de la función de activación a utilizar depende exclusivamente del problema que se está resolviendo.

2.2.2. Función de Pérdida

Para poder entrenar la red neuronal de tal forma que pueda resolver un problema, es necesario crear una función de pérdida y minimizarla bajo un proceso de optimización. La función de pérdida será evaluada en *batches* de entrenamiento durante el proceso de minimización, de forma que se encuentren los parámetros θ de la red óptimos.

La función de pérdida puede estar compuesta por varios términos, los cuales deben estar ponderados por pesos w_j para regular la importancia de cada uno, tal como se visualiza en la siguiente ecuación.

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = \sum_j w_j \mathcal{L}_j(\boldsymbol{\theta}; \mathcal{S}_j) \quad (2.82)$$

En donde \mathcal{L} corresponde a la función de pérdida, \mathcal{S}_j al *batch* de entrenamiento del j -ésimo término \mathcal{L}_j de la función de pérdida y w_j su peso asociado. El *batch* de entrenamiento \mathcal{S} será la unión de los subconjuntos \mathcal{S}_j .

$$\mathcal{S} = \bigcup_j \mathcal{S}_j \quad (2.83)$$

Comúnmente, se utiliza la función de error cuadrático medio (MSE) o la función de entropía cruzada para la función de pérdida. La elección de esta función depende exclusivamente de la naturaleza del problema y del tipo de datos con los que se está trabajando.

2.2.3. Métodos de Optimización

Al tener la función de pérdida \mathcal{L} construida, se puede utilizar para encontrar el conjunto de parámetros $\boldsymbol{\theta}$ óptimos de la red neuronal mediante el proceso de optimización. Para este proceso, se utilizan métodos basados en el descenso de gradiente con algunas variaciones, o métodos de Cuasi-Newton [29]. Además, es común agregar métodos estocásticos al proceso de optimización para aumentar la generalización de la solución. Lo anterior permite evitar mínimos locales, los cuales son fáciles de encontrar debido a la gran cantidad de parámetros que se están optimizando paralelamente. Por ejemplo, en las iteraciones se puede ir entregando muestras aleatorias del conjunto de entrenamiento a la función de pérdida para inducir aleatoriedad.

A continuación se detallarán 3 de los algoritmos más utilizados para la optimización de redes neuronales, 2 basados en descenso de gradiente y 1 basado en los métodos de Cuasi-Newton.

2.2.3.1. Descenso de Gradiente

Este método es el más simple, y se basa en avanzar en la dirección contraria al gradiente de la función de pérdida. De esta forma se actualizan los parámetros $\boldsymbol{\theta}$ en cada iteración. La tasa de aprendizaje se regula con el parámetro λ , el cual puede ser constante o dependiente de las iteraciones.

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \lambda \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_k; \mathcal{S}) \quad (2.84)$$

Este método al ser bastante simple, tiende a caer en mínimos locales, generando una conver-

gencia bastante lenta

2.2.3.2. ADAM (Adaptive Moment Estimation)

El método ADAM [30] combina los conceptos de *momentum* y *RMSprop*. Calcula la tasa de aprendizaje adaptativa para cada parámetro y utiliza estimaciones del primer y segundo momento de los gradientes para escalar la tasa de aprendizaje. ADAM ha demostrado ser efectivo en una variedad de problemas de aprendizaje profundo debido a su capacidad para adaptar dinámicamente la tasa de aprendizaje para cada parámetro y para mantener tasas de aprendizaje independientes para cada parámetro.

A diferencia del método común de descenso de gradiente, ADAM considera más parámetros y etapas en una iteración. Un detalle de su funcionamiento se encuentra a continuación:

Primero se debe estimar el primer momento \mathbf{m}_k y el segundo momento \mathbf{v}_k del gradiente de la función de pérdida $\mathbf{g}_k = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_k)$

$$\begin{aligned}\mathbf{m}_k &= \beta_1 \mathbf{m}_{k-1} + (1 - \beta_1) \mathbf{g}_k \\ \mathbf{v}_k &= \beta_2 \mathbf{v}_{k-1} + (1 - \beta_2) \mathbf{g}_k^2\end{aligned}\tag{2.85}$$

En esta ecuación \mathbf{g}_k^2 corresponde al gradiente \mathbf{g}_k con todas sus entradas al cuadrado.

Luego, se deben corregir los sesgos de ambos estimadores con las siguientes ecuaciones:

$$\begin{aligned}\hat{\mathbf{m}}_k &= \frac{\mathbf{m}_k}{1 - \beta_1^k} \\ \hat{\mathbf{v}}_k &= \frac{\mathbf{v}_k}{1 - \beta_2^k}\end{aligned}\tag{2.86}$$

Por último se actualizan los parámetros $\boldsymbol{\theta}$.

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\lambda}{\sqrt{\hat{\mathbf{v}}_k + \epsilon}} \hat{\mathbf{m}}_k\tag{2.87}$$

Notar que este algoritmo necesita 4 parámetros, siendo λ la tasa de aprendizaje, β_1 y β_2 los coeficientes de decaimiento exponencial para el primer y segundo momento respectivamente, y ϵ un término de suavizado para evitar la división por cero.

2.2.3.3. BFGS (Broyden–Fletcher–Goldfarb–Shanno)

El método de BFGS es un método iterativo de segundo orden utilizado para resolver problemas no lineales de minimización. Este algoritmo es un método de Cuasi-Newton, debido a que busca aproximar la matriz Hessiana de la función a minimizar.

De manera general, todos los métodos de Cuasi-Newton realizan predicciones del punto de optimización de una función en cada iteración, de modo que el punto $\boldsymbol{\theta}_{k+1}$ esté relacionado con el punto en la iteración anterior $\boldsymbol{\theta}_k$ de la siguiente forma:

$$\boldsymbol{\theta}_{k+1} = \alpha_k \mathbf{p}_k + \boldsymbol{\theta}_k \quad (2.88)$$

En donde α_k es el tamaño del paso, y \mathbf{p}_k la dirección de búsqueda. Para obtener la dirección de búsqueda \mathbf{p}_k , se debe resolver el siguiente sistema lineal:

$$\mathbf{B}_k \mathbf{p}_k = -\mathbf{g}_k \quad (2.89)$$

En donde $\mathbf{g}_k = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_k)$ corresponde al gradiente de la función a minimizar evaluado en $\boldsymbol{\theta}_k$ y \mathbf{B}_k es la aproximación de la matriz Hessiana de \mathcal{L} en $\boldsymbol{\theta}_k$. Además, el tamaño del paso α_k es encontrado realizando un “line search” en la dirección \mathbf{p}_k , minimizando $\mathcal{L}(\boldsymbol{\theta}_k + \alpha \mathbf{p}_k)$ sobre el escalar α .

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}_k + \alpha \mathbf{p}_k) \quad (2.90)$$

La gran variación que existe entre los métodos de Cuasi-Newton es la forma de calcular la aproximación de la matriz Hessiana \mathbf{B}_k . Para el caso de BFGS, esta es calculada en cada iteración de la siguiente forma (notación matricial):

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} \quad (2.91)$$

En donde $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$ y $\mathbf{s}_k = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$. Sin embargo, en vez de calcular la matriz Hessiana en cada paso, es conveniente calcular directamente su inversa para resolver la ecuación 2.89. Esta se puede obtener a partir de la siguiente ecuación:

$$\mathbf{H}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) \mathbf{H}_k \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \quad (2.92)$$

En donde \mathbf{H}_k se define como la inversa de \mathbf{B}_k . Notar que este método requiere guardar la inversa de la matriz Hessiana en cada iteración, necesaria para calcular la aproximación de esta en la iteración siguiente. Esto lo convierte en un método costoso en términos computacionales debido

a que se almacena una matriz densa. Sin embargo, existen mejoras de este algoritmo como L-BFGS [31], el cual utiliza valores del gradiente \mathbf{g}_k y de $\boldsymbol{\theta}_k$, en un historial de m iteraciones para construir la inversa \mathbf{H}_k . Para esto, simplemente se fija una partida \mathbf{H}_{k-m} , comúnmente como la matriz identidad, y se aplica la ecuación recursiva 2.92 durante $m+1$ iteraciones para obtener \mathbf{H}_{k+1} . Al solo necesitar guardar los vectores \mathbf{g}_k y de $\boldsymbol{\theta}_k$ se evita guardar una matriz densa, reduciendo la memoria.

2.2.4. Diferenciación Automática y Back Propagation

Como se dijo en la sección anterior, para optimizar la red neuronal se necesita calcular el gradiente de la función de pérdida. Además, la construcción misma de la función de pérdida puede contener derivadas de las variables de salida respecto a las variables de entrada. Es por esto que se necesita un algoritmo para poder calcular las derivadas eficientemente.

La técnica más utilizada es el algoritmo de *Back Propagation*, la cual consiste en crear un grafo computacional para realizar eficientemente una diferenciación automática [32]. Esto permite calcular derivadas de forma rápida minimizando el uso de memoria.

Un ejemplo sencillo del algoritmo se detallará a continuación. Para esto se considerará una red MLP, como la que se muestra a continuación:

$$\begin{aligned} \mathbf{h}^1 &= \sigma^1(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) & k = 1 & \text{capa entrada} \\ \mathbf{h}^k &= \sigma^k(\mathbf{W}^k \mathbf{h}^{k-1} + \mathbf{b}^k) & 2 \leq k \leq L & \text{capas ocultas} \\ u_{\theta} &= \sigma^{L+1}(\mathbf{W}^{L+1} \mathbf{h}^L + \mathbf{b}^{L+1}) & k = L+1 & \text{capa salida} \end{aligned} \quad (2.93)$$

Se mostrará como calcular la derivada $\frac{\partial \mathcal{L}}{\partial w_{ij}^k}$ siendo w_{ij}^k una entrada de la matriz de pesos \mathbf{W}^k de la capa k -ésima. Recordar que w_{ij}^k es el peso de la conexión entre la neurona j de la capa $k-1$ y la neurona i de la capa k .

Se utilizará la notación de a_i^k para denotar una entrada del vector $\mathbf{A}^k = \mathbf{W}^k \mathbf{h}^{k-1} + \mathbf{b}^k$ y r^k la cantidad de nodos de la capa k .

$$a_i^k = \sum_{j=1}^{r^k} w_{ij}^k h_j^{k-1} + b_i^k \quad (2.94)$$

Por simplicidad, se asumirá que la función de pérdida solo depende del output de la red, $\mathcal{L} = F(u_{\theta})$.

Utilizando la regla de la cadena, fácilmente se puede notar que:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^k} = \frac{\partial \mathcal{L}}{\partial u_\theta} \frac{\partial u_\theta}{\partial a_i^k} \frac{\partial a_i^k}{\partial w_{ij}^k} \quad (2.95)$$

La tercera derivada de la expresión anterior se puede descomponer como:

$$\begin{aligned} \frac{\partial a_i^k}{\partial w_{ij}^k} &= \frac{\partial}{\partial w_{ij}^k} \left(\sum_{l=0}^{r^{k-1}} w_{il}^k h_l^{k-1} + b_i^k \right) \\ \frac{\partial a_i^k}{\partial w_{ij}^k} &= h_j^{k-1} \end{aligned} \quad (2.96)$$

Por otra parte, descomponiendo la segunda derivada de la ecuación 2.95, se define δ_i^k (llamado valor error o valor delta) como:

$$\delta_i^k = \frac{\partial u_\theta}{\partial a_i^k} \quad (2.97)$$

Notar que el valor delta δ_i^k mide cuánto cambiará el output de la red por un cambio en el output del perceptrón i de la capa k (antes de la función de activación). Utilizando la definición de δ_i^k , por la regla de la cadena se obtiene:

$$\begin{aligned} \delta_i^k &= \frac{\partial u_\theta}{\partial a_i^k} \\ \delta_i^k &= \sum_{l=1}^{r^{k+1}} \frac{\partial u_\theta}{\partial a_l^{k+1}} \frac{\partial a_l^{k+1}}{\partial a_i^k} \end{aligned} \quad (2.98)$$

Además, dado que:

$$a_l^{k+1} = \sum_{j=1}^{r_k} w_{lj}^{k+1} \sigma^k(a_j^k) + b_l^{k+1} \quad (2.99)$$

Se obtiene:

$$\frac{\partial a_l^{k+1}}{\partial a_i^k} = w_{li}^{k+1} \sigma'^k(a_i^k) \quad (2.100)$$

De esta manera, el δ_i^k puede escribirse de la siguiente forma para $k = 1, 2, \dots, L$.

$$\delta_i^k = \sigma'^k(a_i^k) \sum_{l=1}^{r^{k+1}} w_{li}^{k+1} \delta_l^{k+1} \quad (2.101)$$

Además, para la capa de salida, δ_i^{L+1} se escribe como:

$$\delta_i^{L+1} = \sigma'^{L+1}(a_i^{L+1}) \quad (2.102)$$

Juntando la ecuación 2.102 y a ecuación 2.96, se puede reescribir la ecuación 2.95 como:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^k} = F'(u_\theta) h_j^{k-1} \sigma'^k(a_i^k) \sum_{l=1}^{r^{k+1}} w_{li}^{k+1} \delta_l^{k+1} \quad (2.103)$$

De manera análoga, la derivada de la función de pérdida respecto a un componente del vector bias de la capa k , b_i^k , puede calcularse como:

$$\frac{\partial \mathcal{L}}{\partial b_i^k} = F'(u_\theta) \sigma'^k(a_i^k) \sum_{l=1}^{r^{k+1}} w_{li}^{k+1} \delta_l^{k+1} \quad (2.104)$$

Juntando ambos resultados anteriores, la derivada respecto a un peso w_{ij}^k o a un bias b_i^k se calcula de la siguiente forma.

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w_{ij}^k} &= F'(u_\theta) h_j^{k-1} \delta_i^k \\ \frac{\partial \mathcal{L}}{\partial b_i^k} &= F'(u_\theta) \delta_i^k \end{cases} \quad (2.105)$$

Se nota claramente que las derivadas respecto a los parámetros (ya sea w_{ij}^k o b_i^k) dependen del output de una capa h_i^k y del valor de δ_i^k . Además, el valor delta en la capa k depende de los deltas en las capas posteriores $k+1, k+2, \dots, L+1$. Esto implica que en primera instancia, se requiere realizar un paso hacia adelante en la red neuronal para calcular todos los valores de h_i^k y a_i^k en las distintas capas, y luego se realiza un paso hacia atrás (de ahí *back propagation*) para ir calculando los valores de δ_i^k desde la última capa hasta la primera. De esta forma se pueden ir calculando todas las derivadas para formar el gradiente de la función de pérdida. Este mismo algoritmo puede ser adaptado para calcular las derivadas respecto a las variables de entrada.

2.3. Physics Informed Neural Networks

En esta sección se describirá el estado del arte respecto a los métodos que utilizan Redes Neuronales Profundas, *Deep Neural Networks* (ver sección 2.2), 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 obtener su solución [6, 7, 33, 34]. La utilización de PINNs para la resolución de ecuaciones diferenciales parciales trae grandes ventajas, tales como: 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, mecánica de sólidos, transferencia de calor, electromagnetismo [35, 36, 37, 38], e incluso en problemas inversos. Además, existen trabajos teóricos para evaluar su convergencia y posibles errores [39, 40, 41, 42]. Sin embargo, al ser un tema nuevo implica que existe mucho por investigar [8, 43] para que este método pueda competir y/o acoplarse a los métodos tradicionales de FEM, BEM, entre otros [44]. A continuación, se dará una breve descripción de los métodos conocidos como PINNs.

Normalmente, un problema de valor de contorno estacionario² puede ser planteado de la siguiente forma [45].

$$\begin{cases} \mathcal{D}\mathbf{u} = \mathbf{f}(\mathbf{x}) & \mathbf{x} \in \Omega \\ \mathcal{B}\mathbf{u} = \mathbf{g}(\mathbf{x}) & \mathbf{x} \in \partial\Omega \end{cases} \quad (2.106)$$

En donde \mathcal{D} es un operador diferencial actuando sobre \mathbf{u} en todo el dominio, y \mathcal{B} un operador diferencial que actúa sobre \mathbf{u} en la frontera, siendo \mathbf{u} 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.

Entonces, de manera general, casi todos los métodos PINN buscan encontrar una solución aproximada de la solución exacta del problema \mathbf{u} mediante una red neuronal \mathcal{N} , ver figura 2.11, a partir de la minimización de la función de pérdida \mathcal{L} para obtener los parámetros $\boldsymbol{\theta}^*$ óptimos.

$$\mathbf{u} \approx \mathbf{u}_\theta = \mathcal{N}(\mathbf{x}; \boldsymbol{\theta}) \quad (2.107)$$

²Aquí se detalla como resolver problemas estacionarios, pero el método se puede aplicar de forma equivalente para resolver problemas transientes

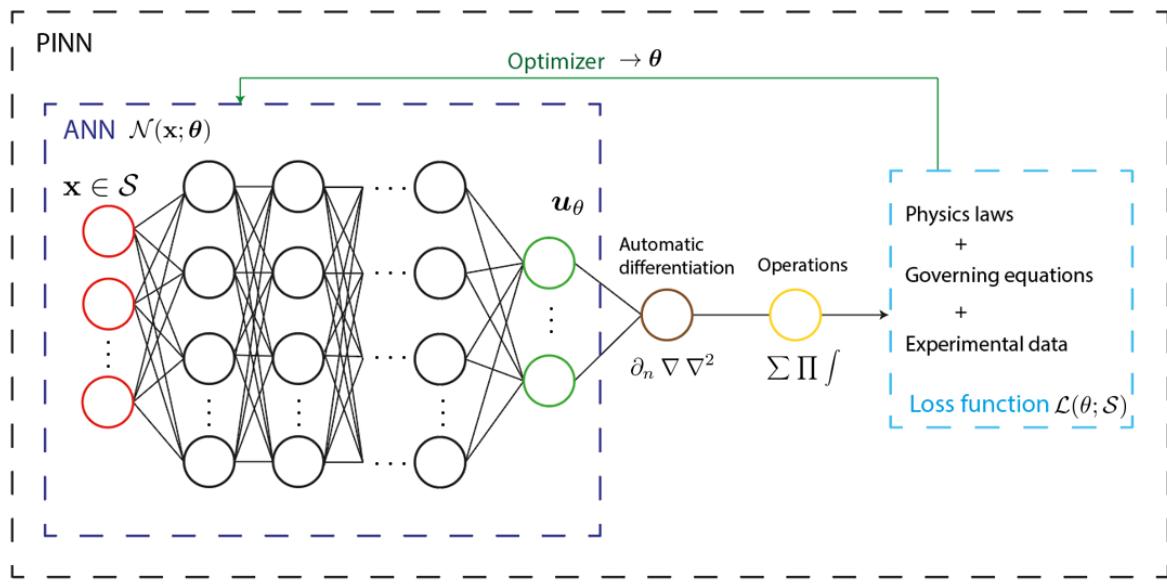


Figura 2.11: Esquema de métodos PINN

Los distintos métodos PINNs se diferencian en la forma que adopta la función de pérdida \mathcal{L} y como se relaciona la red neuronal con la solución del problema. Algunos métodos PINNs se detallan a continuación:

- Métodos basados en puntos de colocación: Utilizan los residuales de la ecuación 2.106 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 [6, 7, 33]. Asimismo, se han realizado avances en descomposición de dominios, los cuales buscan resolver problemas en dominios complejos utilizando más de una red neuronal por subdominio. Entre estos se encuentra: CPINNs, *Conservative Physics Informed Neural Networks* [46], XPINNs, *Extended Physics Informed Neural Networks* [47], APINNs, *Augmented Physics Informed Neural Networks* [48] y DPINNs, *Distributed Physics Informed Neural Networks* [49]. Adicionalmente, se han realizado trabajos en PIELM, *Physics Informed Extreme Learning Machine* [50], en donde se realiza la descomposición de dominios utilizando redes con 1 capa entrenable, permitiendo utilizar otros métodos de optimización más rápidos.
- Métodos basados en la formulación variacional: Utilizan la formulación variacional de la ecuación 2.106 para formar la función de pérdida. Este método ha llevado a una rama nueva de PINN, llamado VPINN, *Variational Physics Informed Neural Networks* [51], y *Deep Ritz Method* [52].

- Métodos basados en la formulación integral en la frontera: Utilizan la formulación integral en la frontera de la ecuación 2.106 para formar la función de pérdida. Para este método se han desarrollado ramas como BINN o BINet, *Boundary Integral Neural Networks* [53, 54].
- Métodos basados en el aprendizaje de operadores: Este método consiste en formular una red neuronal como un aproximador de operadores no lineales, lo que permite resolver PDEs. Se han desarrollado arquitecturas específicas para esta aplicación como DeepONet, *Deep Operator Network* [55].

2.3.1. Métodos Basados en Puntos de Colocación

Este método 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} .

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = w_{\partial\Omega} \mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}; \mathcal{S}_{\partial\Omega}) + w_{\Omega} \mathcal{L}_{\Omega}(\boldsymbol{\theta}; \mathcal{S}_{\Omega}) + w_{data} \mathcal{L}_{data}(\boldsymbol{\theta}; \mathcal{S}_{data}) \quad (2.108)$$

En la función de pérdida, el primer término $\mathcal{L}_{\partial\Omega}$ corresponde al componente que incluye las condiciones de borde, y el segundo término \mathcal{L}_{Ω} corresponde al residual de la ecuación diferencial. Notar además que estos términos se multiplican por respectivos pesos w_k 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 [33, 56]. Adicionalmente, se puede añadir un término \mathcal{L}_{data} para contabilizar datos conocidos disponibles \mathbf{u}_{data} . 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, $\mathcal{S} = \mathcal{S}_{\Omega} \cup \mathcal{S}_{\partial\Omega} \cup \mathcal{S}_{data}$. De esta forma, cada término de la función de pérdida se puede calcular de la siguiente manera.

$$\begin{aligned} \mathcal{L}_{\Omega}(\boldsymbol{\theta}; \mathcal{S}_{\Omega}) &= \frac{1}{N_{\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega}} \left\| \mathcal{D}\mathbf{u}_{\theta}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_i) \right\|^2 \\ \mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}; \mathcal{S}_{\partial\Omega}) &= \frac{1}{N_{\partial\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\partial\Omega}} \left\| \mathcal{B}\mathbf{u}_{\theta}(\mathbf{x}_i) - \mathbf{g}(\mathbf{x}_i) \right\|^2 \\ \mathcal{L}_{data}(\boldsymbol{\theta}; \mathcal{S}_{data}) &= \frac{1}{N_{data}} \sum_{\mathbf{x}_i \in \mathcal{S}_{data}} \left\| \mathbf{u}_{\theta}(\mathbf{x}_i) - \mathbf{u}_{data}(\mathbf{x}_i) \right\|^2 \end{aligned} \quad (2.109)$$

En donde N_j corresponde a la cantidad de muestras que tiene el conjunto \mathcal{S}_j .

2.3.1.1. Descomposición de Dominios

Los métodos basados en puntos de colocación permiten una descomposición de dominios, en donde se utilizan redes neuronales diferentes que replican la solución en cada subdominio. De forma esquemática para 2 subdominios se tiene lo siguiente:

$$\mathbf{u} \approx \mathbf{u}_\theta = \begin{cases} \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) & \mathbf{x} \in \Omega_1 \\ \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) & \mathbf{x} \in \Omega_2 \end{cases} \quad (2.110)$$

Se utilizará la notación de \mathbf{u}_θ^1 y \mathbf{u}_θ^2 para la solución en los distintos subdominios Ω_1, Ω_2 , y $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2$ los correspondientes conjuntos $\boldsymbol{\theta}$ para cada red neuronal de cada subdominio. La función de pérdida sigue la misma forma detallada anteriormente (ecuación 2.109), 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}_Γ^j que es calculado en los puntos del conjunto \mathcal{S}_Γ . Entonces, para cada red j , su función de pérdida queda como:

$$\mathcal{L}^j(\boldsymbol{\theta}^j; \mathcal{S}^j) = w_{\partial\Omega_j} \mathcal{L}_{\partial\Omega_j}^j(\boldsymbol{\theta}^j; \mathcal{S}_{\partial\Omega_j}) + w_{\Omega_j} \mathcal{L}_{\Omega_j}^j(\boldsymbol{\theta}^j; \mathcal{S}_{\Omega_j}) + w_\Gamma \mathcal{L}_\Gamma^j(\boldsymbol{\theta}; \mathcal{S}_\Gamma) \quad (2.111)$$

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

$$\mathcal{L}_\Gamma^j(\boldsymbol{\theta}; \mathcal{S}_\Gamma) = \frac{1}{N_\Gamma} \sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} \left\| \mathcal{C}_j \mathbf{u}_\theta^j(\mathbf{x}_i) - \overline{\mathcal{C}\mathbf{u}_\theta}(\mathbf{x}_i) \right\|^2 \quad (2.112)$$

En este término, el operador \mathcal{C} actúa sobre \mathbf{u}_θ en la interfaz de ambos dominios. Este operador puede contabilizar la continuidad en la función, su gradiente, o del residual, dependiendo del set de ecuaciones a resolver. La solución de cada dominio en la interfaz puede aproximarse al promedio entre las redes que comparten esa interfaz $\overline{\mathcal{C}\mathbf{u}_\theta}$, o al output de las redes aledañas.

Este método tiene la ventaja que puede ser aplicado a cualquier PDE de la forma 2.106, descomponiendo el dominio en subdominios a conveniencia. Por otra parte, este tipo de metodología sirve bastante en problemas con interfaces, en donde la solución a cada lado de la interfaz puede ser totalmente distinta. Lo anterior evidencia la necesidad de utilizar 2 redes diferentes para replicar la solución a cada lado de la interfaz.

2.3.2. Métodos Basados en la Formulación Variacional

De manera general, estos métodos minimizan la forma débil de la ecuación 2.106, o el funcional de energía asociado. A modo de ejemplo, se mostrará la aplicación utilizando la forma débil. Se puede obtener la forma débil de la ecuación 2.106 multiplicando la ecuación por una función de ponderación \mathbf{v} , e integrando sobre todo el dominio Ω :

$$\int_{\Omega} \mathbf{v} \cdot \mathcal{D}\mathbf{u}_{\theta} dV = \int_{\Omega} \mathbf{v} \cdot \mathbf{f}(\mathbf{x}) dV \quad (2.113)$$

De esta manera, la función de pérdida de la ANN corresponderá al residual de la ecuación anterior.

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = \int_{\Omega} \mathbf{v} \cdot \mathcal{D}\mathbf{u}_{\theta} dV - \int_{\Omega} \mathbf{v} \cdot \mathbf{f}(\mathbf{x}) dV \quad (2.114)$$

Una práctica comúnmente empleada es reducir el requerimiento de continuidad de la función \mathbf{u}_{θ} , lo que se aplica integrando por partes y utilizando el teorema de la divergencia. A modo de ejemplo, si el operador \mathcal{D} se define como $\mathcal{D} := \nabla \cdot \mathcal{A} + \mathcal{Z}$, siendo \mathcal{A}, \mathcal{Z} operadores diferenciales de menor orden, la ecuación 2.114 puede reescribirse de la siguiente forma:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = & - \int_{\Omega} (\nabla \cdot \mathbf{v}) : \mathcal{A}\mathbf{u}_{\theta} dV + \int_{\Omega} \mathbf{v} \cdot \mathcal{Z}\mathbf{u}_{\theta} dV \\ & + \oint_{\partial\Omega} \mathbf{v} \cdot \mathcal{A}\mathbf{u}_{\theta} \cdot d\mathbf{S} - \int_{\Omega} \mathbf{v} \cdot \mathbf{f}(\mathbf{x}) dV \end{aligned} \quad (2.115)$$

En la ecuación anterior se puede visualizar como el requerimiento de diferenciación de \mathbf{u}_{θ} es disminuido. Si la condición de borde de la ecuación 2.106 no se satisface completamente con la integral de frontera (por ejemplo condiciones de Dirichlet), se pueden agregar términos idénticos a los utilizados en la sección 2.3.1 a la función de pérdida \mathcal{L} .

Es importante destacar que para calcular la función de pérdida definida en la ecuación 2.115 se necesitará una cuadratura para evaluar las integrales. Esta cuadratura entregará los puntos de colocación \mathcal{S} de la función de pérdida. Si se quiere evitar trabajar con cuadraturas fijas, se podría implementar una integración de Monte Carlo para inducir aleatoriedad y generalización de la solución.

2.3.3. Métodos Basados en la Formulación Integral en la Frontera

En el caso de que se pueda obtener la formulación integral de la ecuación 2.106, esta puede ser utilizada para formar la función de pérdida. Para obtener esta formulación se debe aplicar el

producto interno de la ecuación 2.106 con su función de Green.

$$\int_{\Omega} G(\mathbf{x}, \mathbf{x}') \mathcal{D}\mathbf{u}(\mathbf{x}') dV' = \int_{\Omega} G(\mathbf{x}, \mathbf{x}') \mathbf{f}(\mathbf{x}') dV' \quad (2.116)$$

En donde G se considera como la función de Green de la ecuación 2.106, la cual satisface la siguiente relación:

$$\mathcal{D}G(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x} - \mathbf{x}') \quad (2.117)$$

Para llevar la ecuación 2.116 a la frontera del dominio, se deben aplicar las identidades de Green. De esta forma, se obtiene la siguiente expresión en el borde $\partial\Omega$, con $\mathbf{x} \in \partial\Omega$.

$$\begin{aligned} \frac{1}{2} \mathbf{u}_\theta(\mathbf{x}) &= \oint_{\partial\Omega} G(\mathbf{x}, \mathbf{x}') \partial_n \mathbf{u}_\theta(\mathbf{x}') dS' \\ &\quad - \oint_{\partial\Omega} \partial_n G(\mathbf{x}, \mathbf{x}') \mathbf{u}_\theta(\mathbf{x}') dS' \\ &\quad + \int_{\Omega} G(\mathbf{x}, \mathbf{x}') \mathbf{f}(\mathbf{x}') dV' \end{aligned} \quad (2.118)$$

En la ecuación anterior 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:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = \frac{1}{N} \sum_{\mathbf{x}_i \in \mathcal{S}} \left\| \frac{1}{2} \mathbf{u}_\theta(\mathbf{x}_i) - \mathcal{V}\mathbf{u}_\theta(\mathbf{x}_i) + \mathcal{K}\mathbf{u}_\theta(\mathbf{x}_i) - \mathcal{T}\mathbf{f}(\mathbf{x}_i) \right\|^2 \quad (2.119)$$

Este método requiere de una colección de puntos \mathcal{S} , con $N = |\mathcal{S}|$, en donde se evalúa la función de pérdida. Adicionalmente, se necesitará un método de cuadratura para evaluar las integrales de superficie.

2.3.4. Métodos Basados en el Aprendizaje de Operadores

Para resolver la ecuación 2.106, este método intenta aprender un operador que mapee el espacio de funciones para el término fuente y la posición de evaluación, con el valor de la solución. Dicho esto, este método generará una red neuronal de la siguiente forma:

$$\mathbf{u} \approx \mathbf{u}_\theta = \mathcal{N}(\mathbf{x}, \mathbf{f}; \boldsymbol{\theta}) \quad (2.120)$$

Notar como la red neuronal necesita de input el punto de evaluación \mathbf{x} y la función \mathbf{f} .

Para esto, la arquitectura DeepONet especifica una red compuesta por 2 ramificaciones y una última operación:

- *Branch net*: Esta rama procesa la función \mathbf{f} en puntos de evaluación llamados “sensores”. Estos sensores estarán fijados como hiperparámetros y le dará información a la red neuronal sobre el comportamiento de esta función.
- *Trunk net*: Esta rama procesa el punto de evaluación \mathbf{x} . De esta forma, la red mapea estos puntos a una representación interna.
- Combinación: La salida de ambas redes es combinada para obtener el valor de \mathbf{u}_θ en el punto \mathbf{x} el cual aproxima la solución de la ecuación 2.106.

Notar entonces que este método permitirá obtener la solución de la ecuación 2.106 para cualquier función \mathbf{f} que se utilice como término fuente. Para poder aprender este operador, la función de pérdida se formula de la siguiente manera:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = w_{data}\mathcal{L}_{data}(\boldsymbol{\theta}; \mathcal{S}_{data}) + w_{phys}\mathcal{L}_{phys}(\boldsymbol{\theta}; \mathcal{S}_{phys}) \quad (2.121)$$

En donde el término \mathcal{L}_{data} corresponde a la diferencia con los datos conocidos y \mathcal{L}_{phys} corresponde a la función de pérdida definida en los métodos basados en puntos de colocación, loss físico (sección 2.3.1). El término \mathcal{L}_{data} es obligatorio en este método, debido a que la red neuronal se tiene que entrenar con soluciones reales de la ecuación 2.106 para varias funciones \mathbf{f} diferentes. Considerando \mathcal{S}_{data} un conjunto de puntos donde se conoce la solución del la PDE para distintas funciones \mathbf{f} , y $N_{data} = |\mathcal{S}_{data}|$, la función de pérdida \mathcal{L}_{data} queda como:

$$\mathcal{L}_{data}(\boldsymbol{\theta}; \mathcal{S}_{data}) = \frac{1}{N_{data}} \sum_{\mathbf{x}_i, \mathbf{f}_i \in \mathcal{S}_{data}} \left\| \mathbf{u}_\theta(\mathbf{x}_i, \mathbf{f}_i) - \mathbf{u}_{data}(\mathbf{x}_i, \mathbf{f}_i) \right\|^2 \quad (2.122)$$

El entrenamiento se suele separar en 2 fases: Una primera fase utilizando solo el término \mathcal{L}_{data} , y una segunda donde se añade el término \mathcal{L}_{phys} . Para utilizar el término \mathcal{L}_{data} se tendrá que utilizar otro método para resolver la PDE antes del entrenamiento. Por otra parte, el término \mathcal{L}_{phys} no es obligatorio en esta aplicación, pero se suele agregar para aumentar la generalización de la solución para la ecuación en específico. Lo anterior permite extrapolar situaciones en donde la función \mathbf{f} del problema no se encuentra en el conjunto de entrenamiento \mathcal{S}_{data} .

Capítulo 3

Metodología

3.1. PINNs Aplicado a PDEs Elípticas con Interfaz

Actualmente existen trabajos en donde se aplica PINNs para resolver PDEs elípticas con interfaz. Para mostrar estos avances, se formulará el siguiente problema a modo de ejemplo.

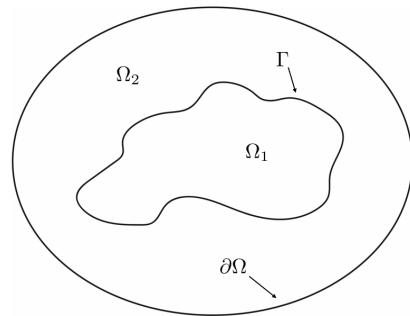


Figura 3.1: Representación del dominio con interfaz Γ

Considerar 2 dominios Ω_1, Ω_2 que comparten una interfaz Γ , en donde se tiene el siguiente problema con interfaz, dado por una PDE elíptica lineal.

$$\begin{aligned}
 -\nabla \cdot (a_1 \nabla u) + b_1 u &= f_1(\mathbf{x}) \quad \mathbf{x} \in \Omega_1 \\
 -\nabla \cdot (a_2 \nabla u) + b_2 u &= f_2(\mathbf{x}) \quad \mathbf{x} \in \Omega_2 \\
 [\![u]\!] &= \varphi(\mathbf{x}) \quad \mathbf{x} \in \Gamma \\
 [\![a \partial_n u]\!] &= \omega(\mathbf{x}) \quad \mathbf{x} \in \Gamma \\
 u &= g(\mathbf{x}) \quad \mathbf{x} \in \partial\Omega
 \end{aligned} \tag{3.1}$$

En donde la operación $[\![\cdot]\!]$ representa el salto en la interfaz Γ . Además, se consideran los co-

3. Metodología

eficientes a_i, b_i constantes en cada subdominio Ω_i . Notar que esta formulación coincide con la ecuación de Poisson-Boltzmann linealizada en cualquiera de sus formas de regularización, tomando los valores adecuados de las constantes a_i, b_i y las funciones φ, ω, f_i, g (considerando una condición de Dirichlet en un dominio acotado).

En la mayoría de los trabajos [57, 58, 59, 60, 61, 62] recomiendan aproximar la solución de la ecuación 3.1 con 2 redes neuronales, 1 por subdominio. Esto ayuda a representar de buena manera la solución del problema, debido a que se puede elegir la arquitectura de la red en cada subdominio de acuerdo al comportamiento de la solución. Además, logra replicar los saltos que puedan existir en la interfaz, ya sea para la función u o su derivada $\partial_n u$.

$$u \approx u_\theta = \begin{cases} \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) & \mathbf{x} \in \Omega_1 \\ \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) & \mathbf{x} \in \Omega_2 \end{cases} \quad (3.2)$$

Como notación, se usará u_θ^i para la solución entregada por la red \mathcal{N}_i del subdominio Ω_i con parámetros entrenables $\boldsymbol{\theta}^i$. Para entrenar la red neuronal mediante el proceso de minimización, se puede formular la siguiente función de pérdida, donde $\boldsymbol{\theta} = \boldsymbol{\theta}^1 \cup \boldsymbol{\theta}^2$. Esta formulación coincide con la descomposición de dominios definida en la sección 2.3.1.1.

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = & w_{\Omega_1} \mathcal{L}_{\Omega_1}(\boldsymbol{\theta}^1; \mathcal{S}_{\Omega_1}) + w_{\Omega_2} \mathcal{L}_{\Omega_2}(\boldsymbol{\theta}^2; \mathcal{S}_{\Omega_2}) + w_{\partial\Omega} \mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) \\ & + w_{\Gamma_D} \mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) + w_{\Gamma_N} \mathcal{L}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) \end{aligned} \quad (3.3)$$

En la ecuación anterior se muestra que la función de pérdida tendrá términos asociados a los residuales en cada subdominio \mathcal{L}_{Ω_1} y \mathcal{L}_{Ω_2} , un término para la condición de borde $\mathcal{L}_{\partial\Omega}$ y dos términos asociados a las condiciones en la interfaz \mathcal{L}_{Γ_D} y \mathcal{L}_{Γ_N} . Además, cada término de la función de pérdida tendrá un peso asociado w_j y un conjunto de puntos de entrenamiento \mathcal{S}_j con $N_j = |\mathcal{S}_j|$, donde $j = \Omega_1, \Omega_2, \partial\Omega, \Gamma$.

$$\mathcal{S} = \bigcup_j \mathcal{S}_j \quad j = \Omega_1, \Omega_2, \partial\Omega, \Gamma \quad (3.4)$$

La construcción de cada término de la función de pérdida (ecuación 3.3) se detalla a continuación:

- Residuales \mathcal{L}_{Ω_1} y \mathcal{L}_{Ω_2} :

$$\mathcal{L}_{\Omega_1}(\boldsymbol{\theta}^1; \mathcal{S}_{\Omega_1}) = \frac{1}{N_{\Omega_1}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_1}} \left[\nabla \cdot (a_1 \nabla u_\theta^1(\mathbf{x}_i)) - b_1 u_\theta^1 + f_1(\mathbf{x}_i) \right]^2 \quad (3.5)$$

$$\mathcal{L}_{\Omega_2}(\boldsymbol{\theta}^2; \mathcal{S}_{\Omega_2}) = \frac{1}{N_{\Omega_2}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_2}} \left[\nabla \cdot (a_2 \nabla u_\theta^2(\mathbf{x}_i)) - b_2 u_\theta^2 + f_2(\mathbf{x}_i) \right]^2 \quad (3.6)$$

- Condición de borde $\mathcal{L}_{\partial\Omega}$:

$$\mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) = \frac{1}{N_{\partial\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\partial\Omega}} \left[u_\theta^2(\mathbf{x}_i) - g(\mathbf{x}_i) \right]^2 \quad (3.7)$$

- Condiciones en la interfaz \mathcal{L}_{Γ_D} y \mathcal{L}_{Γ_N} :

$$\mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) = \frac{1}{N_\Gamma} \sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} \left[u_\theta^2(\mathbf{x}_i) - u_\theta^1(\mathbf{x}_i) - \varphi(\mathbf{x}_i) \right]^2 \quad (3.8)$$

$$\mathcal{L}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) = \frac{1}{N_\Gamma} \sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} \left[a_2 \partial_n u_\theta^2(\mathbf{x}_i) - a_1 \partial_n u_\theta^1(\mathbf{x}_i) - \omega(\mathbf{x}_i) \right]^2 \quad (3.9)$$

Con esta función de pérdida, a partir de un método de optimización, se buscarán los parámetros para ambas redes óptimos $\boldsymbol{\theta}^*$, de tal forma que u_θ replique la solución del problema.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}) \quad (3.10)$$

En trabajos recientes, se ha demostrado que el problema de interfaz detallado en la ecuación 3.1 converge a la solución única del problema en H^2 a medida que aumenta el número de muestras, siempre que el tamaño de ambas redes crezca con el número de muestras [40].

$$\lim_{N_{\Omega_1} \rightarrow \infty} u_\theta^1 = u^* \text{ en } H^2(\Omega_1) \quad \lim_{N_{\Omega_2} \rightarrow \infty} u_\theta^2 = u^* \text{ en } H^2(\Omega_2) \quad (3.11)$$

Donde N_{Ω_i} corresponde al número de muestras en cada dominio, u^* la solución del problema y H^2 al espacio de Sobolev asociado.

Sin embargo, este resultado requiere que la función de pérdida sea regularizada con Lipschitz. A modo de ejemplo, la regularización que utilizan es la siguiente:

$$\begin{aligned} \mathcal{L}^R(\boldsymbol{\theta}; \mathcal{S}) &= \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) + w_{\Omega_1}^R \mathcal{R}_{\Omega_1}(\boldsymbol{\theta}^1; \mathcal{S}_{\Omega_1}) + w_{\Omega_2}^R \mathcal{R}_{\Omega_2}(\boldsymbol{\theta}^2; \mathcal{S}_{\Omega_2}) + w_{\partial\Omega}^R \mathcal{R}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) \\ &\quad + w_{\Gamma_D}^R \mathcal{R}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) + w_{\Gamma_N}^R \mathcal{R}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) \end{aligned} \quad (3.12)$$

Donde $\mathcal{L}^R(\boldsymbol{\theta}; \mathcal{S})$ corresponde a la función de pérdida regularizada con Lipschitz, $\mathcal{L}(\boldsymbol{\theta}; \mathcal{S})$ corresponde a la función de pérdida definida en la ecuación 3.3 y w_j^R corresponden a los pesos de los

3. Metodología

funcionales regularizados \mathcal{R}_j , los cuales están dados por:

$$\begin{aligned}\mathcal{R}_{\Omega_j}(\boldsymbol{\theta}^j; \mathcal{S}_{\Omega_j}) &= \frac{1}{N_{\Omega_j}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_j}} \left[\nabla \cdot (a_j \nabla u_{\theta}^j(\mathbf{x}_i)) - b_j u_{\theta}^j \right]^2 \quad \text{con } j = 1, 2 \\ \mathcal{R}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) &= \frac{1}{N_{\partial\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\partial\Omega}} \left[u_{\theta}^2(\mathbf{x}_i) \right]^2 \\ \mathcal{R}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{\Gamma}) &= \frac{1}{N_{\Gamma}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Gamma}} \left[u_{\theta}^2(\mathbf{x}_i) - u_{\theta}^1(\mathbf{x}_i) \right]^2 \\ \mathcal{R}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{\Gamma}) &= \frac{1}{N_{\Gamma}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Gamma}} \left[a_2 \partial_n u_{\theta}^2(\mathbf{x}_i) - a_1 \partial_n u_{\theta}^1(\mathbf{x}_i) \right]^2\end{aligned}\tag{3.13}$$

Pese a que esta demostración necesita la regularización de Lipschitz, detalla teóricamente que mediante redes neuronales se puede converger a la solución real. Además, se muestra numéricamente que el no utilizar esta regularización también converge a la solución real, pero no existe una demostración teórica de esto.

Por otra parte, se ha demostrado que el error de generalización, al utilizar 2 redes neuronales y utilizando la función de pérdida definida en la ecuación 3.3, está acotado [58]. Las cotas que se plantean son las siguientes:

- Cota superior:

$$\|u_{\theta} - u^*\|_{H^2} \lesssim \mathcal{O} \left(\sum_j |\mathbb{D}_j|^2 N_j^{-\frac{2s-4}{d+2s-4}} \log(N_j) \right)\tag{3.14}$$

- Cota inferior:

$$\|u_{\theta} - u^*\|_{H^2} \gtrsim \mathcal{O} \left(\sum_j |\mathbb{D}_j| N_j^{-\frac{2s-4}{d+2s-4}} \right)\tag{3.15}$$

Donde $|\mathbb{D}_j|$ corresponde a la medida de Lebesgue de los conjuntos $\mathbb{D}_j = \Omega_1, \Omega_2, \partial\Omega, \Gamma$, y los parámetros d y s dependen de la dimensión del problema y la suavidad de la solución. Además, se desprende que el tamaño de cada muestra debe ser proporcional al área (o volumen) de cada subdominio, interfaz o borde, para balancear los distintos términos de la cota superior del error. De esta forma, los 4 términos dados la ecuación 3.16 deben ser proporcionales entre si.

$$|\mathbb{D}_j|^2 N_j^{-\frac{d+4s-8}{d+2s-4}} \left(1 - \frac{2s-4}{d+2s-4} \log(N_j) \right) \quad j = \Omega_1, \Omega_2, \partial\Omega, \Gamma\tag{3.16}$$

3.1.1. Aplicación a Poisson-Boltzmann

Para resolver la ecuación de Poisson-Boltzmann, existen métodos que utilizan esta formulación presentada para PDEs elípticas. Por ejemplo, se han desarrollados métodos como INN *Interfaced Neural Networks*, el cual utiliza una modificación de la función de pérdida definida en la ecuación 3.3 para aplicar el algoritmo MGD (*Multi Gradient Descent*) [60]. INN fue probado para resolver la ecuación de Poisson-Boltzmann linealizada para moléculas pequeñas con buenos resultados. Asimismo, existe un trabajo en donde se utiliza una *Multi-scale Fusion Network* [61] como arquitectura de red neuronal para resolver la PBE. Esta red neuronal fue capaz de replicar los cambios bruscos que pueden tener estas soluciones en distintas proteínas, con errores entre 2 % y 5 % en la predicción de la energía de solvatación.

Por otra parte, existen métodos como MscaleDNN [63], *Multi-scale Deep Neural Network*, que resuelven PDEs elípticas (incluida la ecuación de Poisson-Boltzmann) en su forma variacional utilizando el *Deep Ritz Method* (sección 2.3.2). En este método utilizan una red diseñada para capturar todas las escalas de frecuencias de la solución. Debido a que este método resuelve la formulación variacional, la función de pérdida detallada en la ecuación 3.3 no es aplicable. Para este caso la función de pérdida utilizada se muestra a continuación:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = w_{\Omega} \frac{1}{N_{\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega}} & \left[\frac{a(\mathbf{x}_i) |\nabla u_{\theta}(\mathbf{x}_i)|^2}{2} - b(\mathbf{x}_i) u_{\theta} u_{\theta} - f(\mathbf{x}_i) u_{\theta}(\mathbf{x}_i) \right]^2 \\ & + w_{\partial\Omega} \frac{1}{N_{\partial\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\partial\Omega}} \left[u_{\theta}(\mathbf{x}_i) - g(\mathbf{x}_i) \right]^2 \end{aligned} \quad (3.17)$$

Se puede visualizar que esta formulación variacional debilita la ecuación 3.1. Además, en este método se plantea utilizar solo 1 red neuronal para el dominio completo. Esto evita especificar la interfaz, solo queda implícitamente en las funciones $a(\mathbf{x})$ y $b(\mathbf{x})$ (funciones por tramos para cada subdominio).

Por último, hay métodos como IONet, *Interfaced Operator Network* [64], en que se busca aprender el operador que resuelve la PDE elíptica con interfaz (sección 2.3.4). De esta forma, se entrena con soluciones conocidas del problema para distintas constantes a_i más un loss físico, de tal forma que replique la solución. Esta aplicación fue probada en la ecuación de Poisson-Boltzmann para moléculas pequeñas con buenos resultados, pero necesita que la ecuación sea resuelta con otro método (en este caso FEM) para poder entrenar a la red neuronal.

En la siguiente sección se muestra como se implementará PINNs en este trabajo para resolver la ecuación de Poisson-Boltzmann aplicada a distintas macromoléculas.

3.2. Implementación de PINNs Aplicado a Poisson-Boltzmann

En esta sección se detallará como se implementará PINNs para resolver la ecuación de Poisson-Boltzmann. Esta construcción será aplicable a la ecuación de Poisson-Boltzmann lineal y no lineal en cualquiera de sus esquemas o formas de regularización.

Para la implementación se utilizarán 2 redes neuronales (\mathcal{N}_1 y \mathcal{N}_2), una para aproximar la solución en el soluto Ω_1 y la otra para el solvente Ω_2 . De esta forma, se utilizará la misma función de pérdida definida en la ecuación 3.3, dada por:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = & w_{\Omega_1} \mathcal{L}_{\Omega_1}(\boldsymbol{\theta}^1; \mathcal{S}_{\Omega_1}) + w_{\Omega_2} \mathcal{L}_{\Omega_2}(\boldsymbol{\theta}^2; \mathcal{S}_{\Omega_2}) + w_{\partial\Omega} \mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) \\ & + w_{\Gamma_D} \mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{\Gamma}) + w_{\Gamma_N} \mathcal{L}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{\Gamma}) \end{aligned} \quad (3.18)$$

Cada término de esta función de pérdida \mathcal{L}_j (con $j = \Omega_1, \Omega_2, \partial\Omega, \Gamma$) variará respecto al esquema de regularización de la ecuación de PB (sección 2.1.4). Además, se incluirán nuevos términos a esta función de pérdida para considerar datos conocidos, datos experimentales, o restricciones físicas adicionales. En todos los casos, se utilizará un conjunto de puntos de colocación \mathcal{S} , el cual se construirá según la sección 3.4. Este conjunto será de la forma: $\mathcal{S} = \mathcal{S}_{\Omega_1} \cup \mathcal{S}_{\Omega_2} \cup \mathcal{S}_{\Gamma} \cup \mathcal{S}_{\partial\Omega}$, siendo \mathcal{S}_{Ω_1} y \mathcal{S}_{Ω_2} los puntos de colocación del soluto y solvente respectivamente, \mathcal{S}_{Γ} los puntos de colocación en la interfaz y $\mathcal{S}_{\partial\Omega}$ los puntos de colocación en el borde del dominio del solvente.

Para todos los casos, se utilizará la notación de ϕ para el potencial electrostático, ψ para el potencial de reacción, siendo $\phi_{\theta}, \psi_{\theta}$ sus aproximaciones mediante redes neuronales respectivamente. Recordar que ambos potenciales están relacionados por el potencial de Coulomb \mathcal{G} de la siguiente forma:

$$\phi(\mathbf{x}) = \mathcal{G}(\mathbf{x}) + \psi(\mathbf{x}) \quad (3.19)$$

Para la minimización de ambas redes, se trabajará con una red global \mathcal{N} la cual tendrá 2 ramificaciones dadas por las sub redes¹ \mathcal{N}_1 y \mathcal{N}_2 . Ambas ramificaciones tendrán parámetros entrenables diferentes, dados por los conjuntos $\boldsymbol{\theta}^1$ y $\boldsymbol{\theta}^2$.

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\theta}) = \begin{cases} \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) & \mathbf{x} \in \Omega_1 \\ \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) & \mathbf{x} \in \Omega_2 \end{cases} \quad (3.20)$$

El input de estas redes será una posición \mathbf{x} en el dominio $\Omega = \Omega_1 \cup \Omega_2$, donde la ramificación a utilizar dependerá de la ubicación de este punto de evaluación. El utilizar esta formulación

¹Por simplicidad, ambas ramificaciones serán llamadas sub redes, o simplemente redes.

para unificar ambas redes permitirá utilizar solo 1 optimizador para minimizar la función de pérdida compartida dada en la ecuación 3.18.

Con esta definición, el proceso de minimización de la red neuronal se puede visualizar en la siguiente figura:

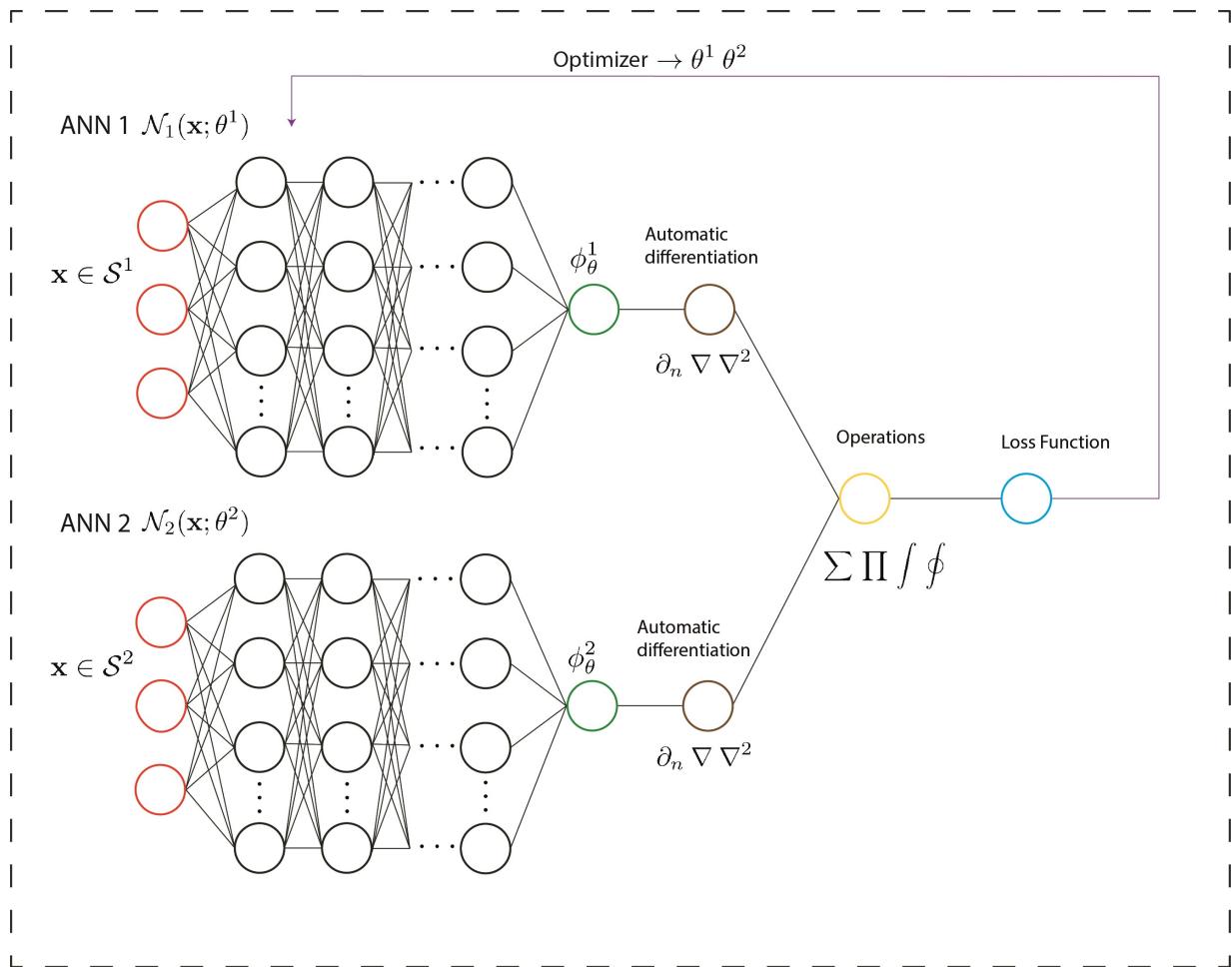


Figura 3.2: Esquema de método PINN con 2 dominios y 1 optimizador

En la siguiente sección se mostrará la construcción de los distintos términos de la función de pérdida para la ecuación de PB lineal. De todas maneras, para la forma no lineal el procedimiento es equivalente.

3.2.1. Ecuación de Poisson-Boltzmann con Formulación Directa

La formulación directa de la ecuación de PB (ecuación 2.26) resuelve para el potencial total en todo el dominio Ω . Es por esto que se diseñará el problema para que ambas redes tengan como output el potencial total en el subdominio de aplicación (soluto o solvente).

$$\phi_{\theta} = \begin{cases} \phi_{\theta}^1 = \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) & \mathbf{x} \in \Omega_1 \\ \phi_{\theta}^2 = \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) & \mathbf{x} \in \Omega_2 \\ \frac{\phi_{\theta}^1 + \phi_{\theta}^2}{2} & \mathbf{x} \in \Gamma \end{cases} \quad (3.21)$$

Esta construcción de PINNs lleva a los siguientes términos de la función de pérdida.

- Dominio soluto:

$$\mathcal{L}_{\Omega_1}(\boldsymbol{\theta}^1; \mathcal{S}_{\Omega_1}) = \frac{1}{N_{\Omega_1}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_1}} \left[\nabla^2 \phi_{\theta}^1(\mathbf{x}_i) + \frac{1}{\epsilon_1} \sum_k q_k \delta(\mathbf{x}_i - \mathbf{x}_k) \right]^2 \quad (3.22)$$

Para evitar la singularidad producto de las cargas puntuales, se aproximará la función delta de Dirac con una función Gaussiana con desviación $\sigma = 0.04 \text{ [\AA]}$ [65].

$$\delta(\mathbf{x}) \approx \frac{1}{(2\pi)^{3/2} \sigma^3} e^{-\frac{1}{2\sigma^2} \|\mathbf{x}\|^2} \quad (3.23)$$

- Dominio solvente:

$$\mathcal{L}_{\Omega_2}(\boldsymbol{\theta}^2; \mathcal{S}_{\Omega_2}) = \frac{1}{N_{\Omega_2}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_2}} \left[\nabla^2 \phi_{\theta}^2(\mathbf{x}_i) - \kappa^2 \phi_{\theta}^2(\mathbf{x}_i) \right]^2 \quad (3.24)$$

- Condición de borde: Como condición de borde, se utilizará la función de Green de Yukawa, la cual es una buena aproximación al estar lejos de la molécula (sección 2.1.8.3).

$$\phi(\mathbf{x}_i) \approx \frac{1}{4\pi\epsilon_2} \sum_k \frac{q_k e^{-\kappa|\mathbf{x}_i - \mathbf{x}_k|}}{|\mathbf{x}_i - \mathbf{x}_k|} \quad (3.25)$$

De esta manera:

$$\mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) = \frac{1}{N_{\partial\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\partial\Omega}} \left[\phi_{\theta}^2(\mathbf{x}_i) - \frac{1}{4\pi\epsilon_2} \sum_k \frac{q_k e^{-\kappa|\mathbf{x}_i - \mathbf{x}_k|}}{|\mathbf{x}_i - \mathbf{x}_k|} \right]^2 \quad (3.26)$$

- Condiciones en la interfaz: Se incluyen ambas restricciones en la interfaz, continuidad del potencial y del desplazamiento eléctrico:

$$\mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{\Gamma}) = \frac{1}{N_{\Gamma}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Gamma}} \left[\phi_{\theta}^2(\mathbf{x}_i) - \phi_{\theta}^1(\mathbf{x}_i) \right]^2 \quad (3.27)$$

$$\mathcal{L}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) = \frac{1}{N_\Gamma} \sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} \left[\epsilon_2 \partial_n \phi_\theta^2(\mathbf{x}_i) - \epsilon_1 \partial_n \phi_\theta^1(\mathbf{x}_i) \right]^2 \quad (3.28)$$

3.2.2. Ecuación de Poisson-Boltzmann con Esquema de Regularización 1

El esquema de regularización 1 de la ecuación de PB (ecuación 2.30) resuelve para el potencial de reacción en todo el dominio Ω . Es por esto que se diseñará el problema para que ambas redes tengan como output el potencial de reacción ψ en el subdominio de aplicación (sólido o solvente).

$$\begin{aligned} \psi_\theta^1 &= \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) \\ \psi_\theta^2 &= \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) \end{aligned} \quad (3.29)$$

De esta forma, la predicción del potencial total estará dada por la siguiente relación:

$$\phi_\theta = \begin{cases} \phi_\theta^1 = \mathcal{G}(\mathbf{x}) + \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) & \mathbf{x} \in \Omega_1 \\ \phi_\theta^2 = \mathcal{G}(\mathbf{x}) + \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) & \mathbf{x} \in \Omega_2 \\ \frac{\phi_\theta^1 + \phi_\theta^2}{2} & \mathbf{x} \in \Gamma \end{cases} \quad (3.30)$$

Notar que a cada output de la red se le debe sumar la función \mathcal{G} correspondiente al potencial de Coulomb (ver ecuación 2.28), para obtener el potencial total ϕ .

El uso del potencial de reacción como output de la red permite obtener una solución sin singularidades, que son producto de las cargas puntuales. Los términos de la función de pérdida para este esquema se detallan a continuación.

- Dominio sólido:

$$\mathcal{L}_{\Omega_1}(\boldsymbol{\theta}^1; \mathcal{S}_{\Omega_1}) = \frac{1}{N_{\Omega_1}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_1}} \left[\nabla^2 \psi_\theta^1(\mathbf{x}_i) \right]^2 \quad (3.31)$$

- Dominio solvente:

$$\mathcal{L}_{\Omega_2}(\boldsymbol{\theta}^2; \mathcal{S}_{\Omega_2}) = \frac{1}{N_{\Omega_2}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_2}} \left[\nabla^2 \psi_\theta^2(\mathbf{x}_i) - \kappa^2 (\psi_\theta^2(\mathbf{x}_i) + \mathcal{G}(\mathbf{x}_i)) \right]^2 \quad (3.32)$$

3. Metodología

- Condición de borde: También se utilizará la función de Green de Yukawa.

$$\mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) = \frac{1}{N_{\partial\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\partial\Omega}} \left[\psi_\theta^2(\mathbf{x}_i) - \frac{1}{4\pi\epsilon_2} \sum_k \frac{q_k e^{-\kappa|\mathbf{x}_i - \mathbf{x}_k|}}{|\mathbf{x}_i - \mathbf{x}_k|} + \mathcal{G}(\mathbf{x}_i) \right]^2 \quad (3.33)$$

- Condiciones en la interfaz: Se incluyen ambas restricciones en la interfaz, continuidad del potencial y del desplazamiento eléctrico:

$$\mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) = \frac{1}{N_\Gamma} \sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} \left[\psi_\theta^2(\mathbf{x}_i) - \psi_\theta^1(\mathbf{x}_i) \right]^2 \quad (3.34)$$

$$\mathcal{L}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_\Gamma) = \frac{1}{N_\Gamma} \sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} \left[\epsilon_2 \frac{\partial}{\partial \mathbf{n}} (\psi_\theta^2(\mathbf{x}_i) + \mathcal{G}(\mathbf{x}_i)) - \epsilon_1 \frac{\partial}{\partial \mathbf{n}} (\psi_\theta^1(\mathbf{x}_i) + \mathcal{G}(\mathbf{x}_i)) \right]^2 \quad (3.35)$$

3.2.3. Ecuación de Poisson-Boltzmann con Esquema de Regularización 2

El esquema de regularización 2 de la ecuación de PB (ecuación 2.31) resuelve para el potencial de reacción en el dominio del soluto Ω_1 y el potencial total en el dominio del solvente Ω_2 . Esto implica que ambas redes tendrán como output potenciales distintos.

$$\begin{aligned} \psi_\theta^1 &= \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) \\ \phi_\theta^2 &= \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) \end{aligned} \quad (3.36)$$

De esta forma, la predicción del potencial total estará dada por la siguiente relación:

$$\phi_\theta = \begin{cases} \phi_\theta^1 = \mathcal{G}(\mathbf{x}) + \mathcal{N}_1(\mathbf{x}; \boldsymbol{\theta}^1) & \mathbf{x} \in \Omega_1 \\ \phi_\theta^2 = \mathcal{N}_2(\mathbf{x}; \boldsymbol{\theta}^2) & \mathbf{x} \in \Omega_2 \\ \frac{\phi_\theta^1 + \phi_\theta^2}{2} & \mathbf{x} \in \Gamma \end{cases} \quad (3.37)$$

Notar como solo se utiliza la función \mathcal{G} en el dominio Ω_1 . A diferencia del esquema de regularización 1, aquí se aproxima el potencial de reacción solo en el soluto, lugar donde existen las singularidades.

A continuación se presentan los términos de la función de pérdida para esta formulación.

- Dominio soluto:

$$\mathcal{L}_{\Omega_1}(\boldsymbol{\theta}^1; \mathcal{S}_{\Omega_1}) = \frac{1}{N_{\Omega_1}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_1}} \left[\nabla^2 \psi_{\theta}^1(\mathbf{x}_i) \right]^2 \quad (3.38)$$

- Dominio solvente:

$$\mathcal{L}_{\Omega_2}(\boldsymbol{\theta}^2; \mathcal{S}_{\Omega_2}) = \frac{1}{N_{\Omega_2}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega_2}} \left[\nabla^2 \phi_{\theta}^2(\mathbf{x}_i) - \kappa^2 \phi_{\theta}^2(\mathbf{x}_i) \right]^2 \quad (3.39)$$

- Condición de borde: También se utilizará la función de Green de Yukawa.

$$\mathcal{L}_{\partial\Omega}(\boldsymbol{\theta}^2; \mathcal{S}_{\partial\Omega}) = \frac{1}{N_{\partial\Omega}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\partial\Omega}} \left[\phi_{\theta}^2(\mathbf{x}_i) - \frac{1}{4\pi\epsilon_2} \sum_k \frac{q_k e^{-\kappa|\mathbf{x}_i - \mathbf{x}_k|}}{|\mathbf{x}_i - \mathbf{x}_k|} \right]^2 \quad (3.40)$$

- Condiciones en la interfaz: Se incluyen ambas restricciones en la interfaz, continuidad del potencial y del desplazamiento eléctrico:

$$\mathcal{L}_{\Gamma_D}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{\Gamma}) = \frac{1}{N_{\Gamma}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Gamma}} \left[\phi_{\theta}^2(\mathbf{x}_i) - \psi_{\theta}^1(\mathbf{x}_i) - \mathcal{G}(\mathbf{x}_i) \right]^2 \quad (3.41)$$

$$\mathcal{L}_{\Gamma_N}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{\Gamma}) = \frac{1}{N_{\Gamma}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Gamma}} \left[\epsilon_2 \frac{\partial}{\partial \mathbf{n}} \left(\phi_{\theta}^2(\mathbf{x}_i) \right) - \epsilon_1 \frac{\partial}{\partial \mathbf{n}} \left(\psi_{\theta}^1(\mathbf{x}_i) + \mathcal{G}(\mathbf{x}_i) \right) \right]^2 \quad (3.42)$$

3.2.4. Términos Adicionales de la Función de Pérdida

A pesar que con los términos detallados anteriormente (residuales, condición de borde y relaciones en la interfaz) el problema puede ser resuelto, se probarán distintos términos adicionales para revisar la variación en la precisión del método.

- Datos conocidos: Esto implica forzar puntos del dominio \mathcal{S}_{data} con soluciones conocidas ϕ^{\dagger} . Esto puede ser aplicable a cada red, pero se utilizará solo para el dominio del solvente.

$$\mathcal{L}_{data}(\boldsymbol{\theta}^2; \mathcal{S}_{data}) = \frac{1}{N_{data}} \sum_{\mathbf{x}_i \in \mathcal{S}_{data}} \left[\phi_{\theta}^2(\mathbf{x}_i) - \phi^{\dagger}(\mathbf{x}_i) \right]^2 \quad (3.43)$$

Pese a que a priori no se conoce la solución de la ecuación de Poisson-Boltzmann (salvo en el Ion de Born o multipolo esférico), no se puede utilizar ϕ^{\dagger} como la solución del problema. Sin embargo, se utilizarán las aproximaciones del potencial en el solvente como la función de Green de Yukawa, ver sección 2.1.8.3. Esta aproximación predice de buena manera el potencial lejos de la interfaz.

3. Metodología

- Ley de Gauss: Pese a que la Ley de Gauss se satisface al resolver la ecuación de Poisson-Boltzmann, puede ser útil agregar este término a la función de pérdida para darle importancia al valor que tiene que tener el desplazamiento eléctrico en la interfaz.

Para agregar este término, se utilizará la siguiente cuadratura para la integral de superficie:

$$\oint_{\Gamma} \overline{\epsilon \partial_n \phi_{\theta}(\mathbf{x}') dS(\mathbf{x}')} \approx \sum_{\mathbf{x} \in \mathcal{S}_{centr}} \overline{\epsilon \partial_n \phi_{\theta}(\mathbf{x}')} A_i \quad (3.44)$$

En donde $\overline{\epsilon \partial_n \phi_{\theta}}$ corresponde al promedio del desplazamiento eléctrico entre ambas redes en la interfaz (notar que para los casos regularizados, este potencial total es obtenido sumando el potencial de Coulomb). Además, \mathcal{S}_{centr} corresponde a los centroides de los elementos de la malla superficial, y A_i su respectiva área. De esta manera, el término para la Ley de Gauss queda como:

$$\mathcal{L}_{Gauss}(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2; \mathcal{S}_{centr}) = \left| \sum_{\mathbf{x}_i \in \mathcal{S}_{centr}} \overline{\epsilon \partial_n \phi_{\theta}(\mathbf{x}_i)} A_i - \sum_k q_k \right|^2 \quad (3.45)$$

- Mediciones experimentales: Otro término que se puede añadir a la función de pérdida es el cálculo del *effective near surface potential* definido en la sección 2.1.7. Esto puede ayudar a fijar el potencial electrostático en el solvente.

Para agregar este término, se utilizará la siguiente cuadratura para calcular las integrales de volumen en el solvente², en donde v_i corresponde al volumen de cada elemento y \mathcal{S}_{Ω^*} el conjunto de puntos con los centroides de cada elemento.

$$\phi_{ENS,\theta}(\mathbf{x}_H) = \frac{-k_b T}{2q_e} \ln \left(\frac{\sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega^*}} v_i |\mathbf{x}_i - \mathbf{x}_h|^{-6} e^{-\frac{q_e \phi_{\theta}(\mathbf{x}_i)}{k_b T}}}{\sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega^*}} v_i |\mathbf{x}_i - \mathbf{x}_h|^{-6} e^{\frac{q_e \phi_{\theta}(\mathbf{x}_i)}{k_b T}}} \right) \quad (3.46)$$

De esta manera, se puede agregar un término asociado a cada átomo de hidrógeno de la siguiente forma:

$$\mathcal{L}_E(\boldsymbol{\theta}^2; \mathcal{S}_H) = \frac{1}{N_H} \sum_{\mathbf{x}_h \in \mathcal{S}_H} \left[\phi_{ENS,\theta}(\mathbf{x}_h) - \phi_{ENS}^{\dagger}(\mathbf{x}_h) \right]^2 \quad (3.47)$$

²Se notó que en las iteraciones tempranas se tiene una predicción totalmente errónea del potencial, lo que genera que las exponenciales tiendan a infinito. Para arreglar este fenómeno se aproximan las exponenciales con expansiones de Taylor.

Siendo \mathcal{S}_H conjunto de puntos con la posición de los átomos de hidrógeno, $\phi_{ENS,\theta}(\mathbf{x}_h)$ el *effective near surface potential* predicho por la red neuronal, y $\phi_{ENS}^\dagger(\mathbf{x}_h)$ la medición experimental.

Es importante destacar que la adición de este término puede generar inestabilidades no deseadas en el proceso de entrenamiento debido a las exponenciales presentes. Por esta razón, se probará también la aproximación de ϕ_{ENS} dado en [22].

$$\phi_{ENS,\theta}(\mathbf{x}_H) = \frac{1}{N_{\Omega^*}} \sum_{\mathbf{x}_i \in \mathcal{S}_{\Omega^*}} \phi_\theta(\mathbf{x}_i) \quad (3.48)$$

Siendo Ω^* la región que rodea al átomo de hidrógeno, y en donde la suma $\sum_i |\mathbf{x}_i - \mathbf{x}_h|^{-6}$ llega al 68 % de su valor. Notar que esta aproximación refleja claramente su naturaleza como promedio alrededor de los átomos de hidrógeno.

3.3. Cálculo de Operadores Diferenciales con Diferenciación Automática

En las funciones de pérdida definidas en la sección anterior, se utilizan operadores diferenciales del output de cada red neuronal como $\nabla^2\phi_\theta$ y $\partial_n\phi_\theta$. Para poder computar estos operadores, se hará uso de la diferenciación de la red neuronal. En esta sección se mostrará un ejemplo de como calcular estos operadores diferenciales haciendo uso de la diferenciación automática. De todas formas, para el código de PINNs se utilizarán versiones de este algoritmo que vienen incluidas en las librerías de redes neuronales.

Se tomará como ejemplo una red *Fully Connected* como la que se muestra en la ecuación 3.49. Cada capa k tendrá una cantidad r^k de neuronas, donde por razones obvias, $r^{L+1} = 1$.

$$\begin{aligned} \mathbf{h}^1 &= \sigma^1(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) & k = 1 & \text{capa entrada} \\ \mathbf{h}^k &= \sigma^k(\mathbf{W}^k \mathbf{h}^{k-1} + \mathbf{b}^k) & 2 \leq k \leq L & \text{capas ocultas} \\ \phi_\theta &= \sigma^{L+1}(\mathbf{W}^{L+1} \mathbf{h}^L + \mathbf{b}^{L+1}) & k = L+1 & \text{capa salida} \end{aligned} \quad (3.49)$$

Para construir los operadores diferenciales, primero se calculará la primera derivada del output ϕ_θ de la red respecto a un input x_i perteneciente al vector de entrada \mathbf{x} .

$$\frac{\partial \phi_\theta}{\partial x_i} = \sum_{j=1}^{r^1} \frac{\partial \phi_\theta}{\partial a_j^1} \frac{\partial a_j^1}{\partial x_i} \quad (3.50)$$

3. Metodología

Se debe recordar la definición del valor δ_i^k , definido en la ecuación 2.102.

$$\delta_i^k = \frac{\partial \phi_\theta}{\partial a_i^k} \quad (3.51)$$

El cual se puede calcular resolviendo la ecuación recursiva para $k = 1, 2, \dots, L$:

$$\delta_i^k = \sigma'^k(a_i^k) \sum_{l=1}^{r^{k+1}} w_{li}^{k+1} \delta_l^{k+1} \quad \text{con} \quad \delta_i^{L+1} = \sigma'^{L+1}(a_i^{L+1}) \quad (3.52)$$

De esta manera, se identifica δ_j^1 en la ecuación 3.50. Por otra parte, se debe trabajar el segundo término, el cual por la arquitectura de la red queda como:

$$\frac{\partial a_j^1}{\partial x_i} = w_{ji}^1 \quad (3.53)$$

De esta manera, la primera derivada del output de la red ϕ_θ puede calcularse de la siguiente forma:

$$\frac{\partial \phi_\theta}{\partial x_i} = \sum_{j=1}^{r^1} \delta_j^1 w_{ji}^1 \quad (3.54)$$

Para el caso de la segunda derivada, se derivará la expresión dada en la ecuación 3.54 nuevamente por un input de la red x_n :

$$\begin{aligned} \frac{\partial}{\partial x_n} \left(\frac{\partial \phi_\theta}{\partial x_i} \right) &= \frac{\partial}{\partial x_n} \sum_{j=1}^{r^1} \delta_j^1 w_{ji}^1 \\ \frac{\partial^2 \phi_\theta}{\partial x_n \partial x_i} &= \sum_{j=1}^{r^1} w_{ji}^1 \frac{\partial \delta_j^1}{\partial x_n} \end{aligned} \quad (3.55)$$

Para simplificar la notación, se definirá el valor lambda asociado a la derivada del valor delta como:

$$\lambda_{in}^k = \frac{\partial \delta_i^k}{\partial x_n} \quad (3.56)$$

De esta manera, trabajando con la definición del valor delta:

$$\begin{aligned}\lambda_{in}^k &= \frac{\partial \delta_i^k}{\partial x_n} \\ \lambda_{in}^k &= \frac{\partial}{\partial x_n} \left(\sigma'^k(a_i^k) \sum_{l=1}^{r^{k+1}} w_{li}^{k+1} \delta_l^{k+1} \right) \\ \lambda_{in}^k &= \sum_{l=1}^{r^{k+1}} w_{li}^{k+1} \left[\sigma'^k(a_i^k) \frac{\partial \delta_l^{k+1}}{\partial x_n} + \delta_l^{k+1} \sigma''^k(a_i^k) \frac{\partial a_i^k}{\partial x_n} \right]\end{aligned}\tag{3.57}$$

Notar que en la expresión anterior aparece nuevamente la derivada del valor delta, pero de la capa siguiente. Esto implica que se puede escribir la ecuación en forma recursiva. El único término que faltaría es la derivada de a_i^k con respecto a x_n . Por simplicidad, este término se definirá como M_{in}^k .

$$\begin{aligned}M_{in}^k &= \frac{\partial a_i^k}{\partial x_n} \\ M_{in}^k &= \sum_{l=1}^{r^{k-1}} \frac{\partial a_i^k}{\partial a_l^{k-1}} \frac{\partial a_l^{k-1}}{\partial x_n}\end{aligned}\tag{3.58}$$

Para esta última expresión, se puede utilizar el resultado mostrado en la ecuación 2.100 para la derivada $\frac{\partial a_i^k}{\partial a_l^{k-1}}$. Con esto, y utilizando la definición de M_{in}^k , se puede plantear la siguiente ecuación recursiva, con $k = 2, 3, \dots, L+1$:

$$M_{in}^k = \sum_{l=1}^{r^{k-1}} w_{il}^k \sigma'^{k-1}(a_l^{k-1}) M_{ln}^{k-1} \quad \text{con} \quad M_{in}^1 = w_{in}^1\tag{3.59}$$

Con esta definición, la ecuación recursiva para el valor lambda, con $k = 1, 2, \dots, L$ queda como:

$$\lambda_{in}^k = \sum_{l=1}^{r^{k+1}} w_{li}^{k+1} [\sigma'^k(a_i^k) \lambda_{ln}^{k+1} + \delta_l^{k+1} \sigma''^k(a_i^k) M_{in}^k] \quad \text{con} \quad \lambda_{in}^{L+1} = \sigma''^{L+1}(a_i^{L+1}) M_{in}^{L+1}\tag{3.60}$$

De esta manera, la segunda derivada del output de la red ϕ_θ puede calcularse de la siguiente forma:

$$\frac{\partial^2 \phi_\theta}{\partial x_n \partial x_i} = \sum_{j=1}^{r^1} w_{ji}^1 \lambda_{jn}^1\tag{3.61}$$

Notar que en cada iteración se tendrán que realizar pasos hacia adelante y hacia atrás de la red para calcular ϕ_θ y sus primeras y segundas derivadas. Esto es porque se tiene que calcular δ_i^k , λ_{in}^k y M_{in}^k para todos los nodos de la red.

3. Metodología

A partir de las ecuaciones 3.54 y 3.61 se pueden calcular todos los operadores diferenciales de interés, como el laplaciano y gradientes $\nabla^2\phi_\theta, \partial_n\phi_\theta$.

3.4. Puntos de Colocación y Mallas

En diversos estudios se detalla la importancia del muestreo de puntos de colocación en la precisión y convergencia del método de PINNs [66, 67]. Se ha demostrado que un muestreo aleatorio es importante para una generalización de la solución, sin embargo, la elección de la distribución de probabilidad para obtener estos puntos no es trivial, ya que afecta en el entrenamiento. En esta misma línea, existen técnicas que regulan la cantidad de puntos en zonas donde los residuales son altos.

Debido a que se trabajará con una geometría irregular dada por la superficie de la molécula, se mallará todo el dominio para así simplificar el proceso de obtención de puntos de colocación. La cantidad de puntos de colocación estará estrechamente relacionado con el número de elementos de la malla. De esta forma, si se tiene una malla uniforme, se podrán generar muestreos uniformes por todo el dominio. Además, la utilización de mallas permite refinar las zonas de interés donde se espera que los residuales sean altos (por ejemplo en la interfaz), lo que entregará un mayor número de puntos de colocación.

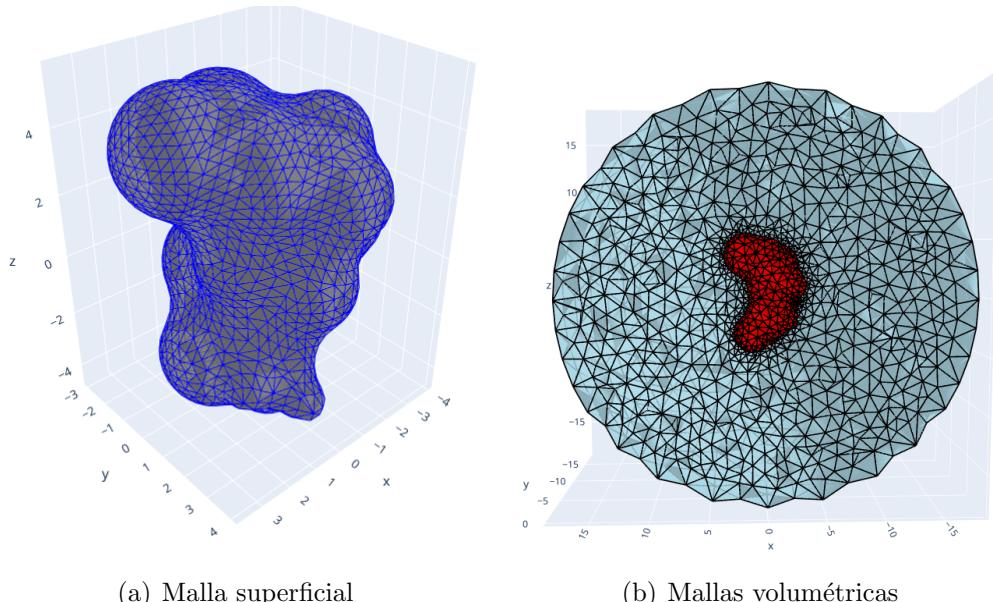


Figura 3.3: Ejemplos de mallas para la arginina, renderizado con [9]

Entonces, el dominio computacional completo estará compuesto por 4 mallas: 2 superficiales (superficie SES de la molécula y una esfera en el contorno del solvente), y 2 volumétricas (sóluto

y solvente). Para ambas mallas superficiales se utilizarán elementos triangulares, y para ambas mallas volumétricas se utilizarán elementos tetraédricos.

Para poder calcular todos los términos de la función de pérdida, se realizarán 2 tipos de muestreos que utilizan las mallas descritas (aleatorio y fijo). En las siguientes secciones se detallará cada muestreo y en qué términos de la función de pérdida serán aplicados.

3.4.1. Muestreo Aleatorio

Como se detalló anteriormente, para aumentar la generalización de la solución entregada por la red neuronal se deben añadir procesos estocásticos en la elección de los puntos de colocación. Para esto, en cada iteración se extraerá aleatoriamente un punto dentro de cada elemento (triangular o tetraédrico) de las 4 mallas. Esto permitirá tener una muestra uniforme alrededor del dominio con aleatoriedad.

Para la obtención de cada punto dentro de cada elemento, se realizará lo siguiente. De manera general, cada elemento tendrá N vértices donde N podrá ser igual a 3 o 4 dependiendo de la malla a la que pertenezca el elemento, superficial o volumétrica.

Para seleccionar un punto aleatorio dentro de un elemento Z , se necesitan N números aleatorios λ obtenidos de una distribución uniforme $\lambda_j \sim \text{Unif}(0, 1)$, con $j = 1, 2, \dots, N$, en donde N corresponde al número de vértices del elemento Z .

A partir de estos N números aleatorios, se deben calcular los números normalizados $\hat{\lambda}_j$ como:

$$\hat{\lambda}_j = \frac{\lambda_j}{\sum_k^N \lambda_k} \quad (3.62)$$

Esto implica que:

$$\sum_{k=1}^N \hat{\lambda}_k = 1 \quad (3.63)$$

Considerando \mathbf{v}_j el vértice j -ésimo del mismo elemento Z , se puede obtener un punto aleatorio dentro del elemento a partir de la combinación lineal de los vértices \mathbf{v}_j y los números aleatorios normalizados $\hat{\lambda}_j$, es decir:

$$\mathbf{x} = \sum_{j=1}^N \lambda_j \mathbf{v}_j \quad (3.64)$$

A partir de esto, \mathbf{x} siempre estará dentro del elemento Z . Este procedimiento es repetido para

3. Metodología

todos los elementos de las 4 mallas para obtener los puntos de colocación.

Adicional a estos puntos de colocación, se incluirán más puntos alrededor de las cargas puntuales. Esto con el propósito de representar de buena manera la aproximación de la función delta Dirac (solo en el caso de la formulación directa de PB). Para cada carga puntual q_k , se añadirán N_q puntos de colocación adicionales obtenidos de una distribución Gaussiana $\mathbf{x} \sim \text{Normal}(\mathbf{x}_q, \sigma^2)$, en donde \mathbf{x}_q es la ubicación de la carga y σ la desviación estándar.

Algoritmo 3.1 Algoritmo para obtener puntos de colocación aleatorios \mathcal{S}

Input: Objetos de mallas superficiales y volumétricas, y posición cargas X_q

```
for malla in mallas do                                ▷ Recorrer mallas
    elementos = malla.elementos
    for elemento in elementos do                      ▷ Recorrer elementos de cada malla
        N = elemento.num_vertices
        Obtener N coeficientes:  $\lambda_j \sim \text{Unif}(0, 1)$ 
        Normalizar cada coeficiente:  $\hat{\lambda}_j = \lambda_j / \sum_k^N \lambda_k$ 
        v = elemento.vertices
        Obtener punto:  $\mathbf{x} = \sum_{j=1}^N \hat{\lambda}_j \mathbf{v}_j$ 
        Agregar punto  $\mathbf{x}$  a conjunto  $\mathcal{S}$ 
    end for
end for
for  $x_q$  in  $X_q$  do                                ▷ Recorrer posición de las cargas
    Obtener  $N_q$  puntos:  $\mathbf{x} \sim \text{Normal}(x_q, \sigma^2)$ 
    Agregar puntos  $\mathbf{x}$  a conjunto  $\mathcal{S}$ 
end for
```

Output: Conjunto de puntos de colocación \mathcal{S}

En la figura 3.4 se muestran los puntos de colocación generados con el algoritmo 3.1.

Notar que la distribución de puntos en las 4 zonas de interés depende de la característica de la malla a generar. Es por esto que se buscará generar una buena malla superficial de la molécula, con refinamiento en las zonas que se predice un alto error. Lo anterior aumentará el tamaño de la muestra en la vecindad de la interfaz (sólido y solvente), lugar donde también se espera que los residuales sean altos. El método para obtener el muestreo concuerda con los trabajos teóricos respecto a la distribución de puntos [66, 67].

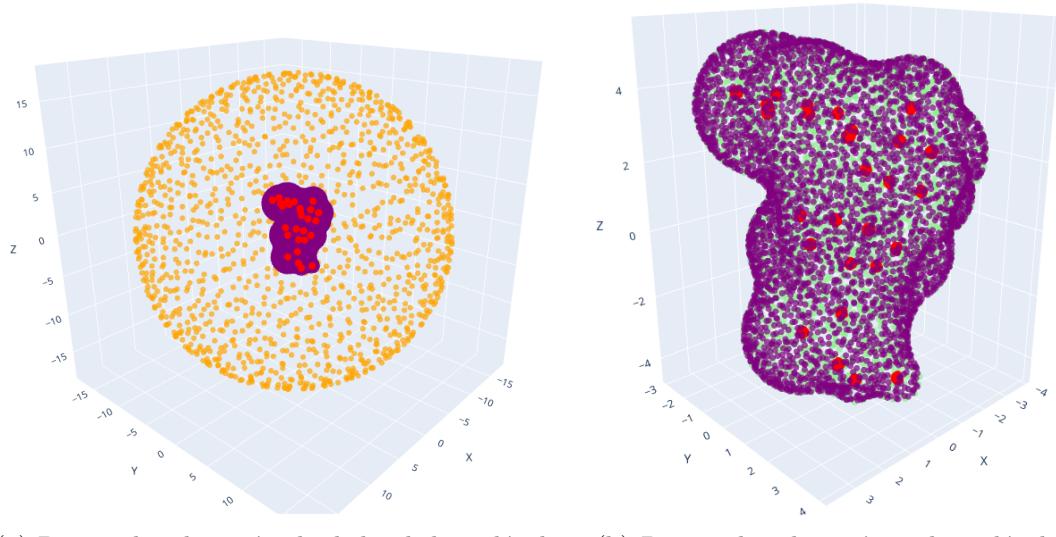


Figura 3.4: Ejemplo de puntos de colocación para la arginina generados con algoritmo 3.1, renderizado con [9]

3.4.2. Muestreo Fijo

Para el resto de los términos de la función de pérdida se utilizará un muestreo fijo, lo que implica que los puntos de colocación serán los mismos en todas las iteraciones. Esto será aplicado al término de la Ley de Gauss, al término de las mediciones experimentales y a los datos conocidos a fijar. A estos términos no se le puede agregar la aleatoriedad detallada anteriormente debido a 2 razones:

- Para el término respecto a los datos conocidos, se prefiere utilizar un muestreo fijo. Esto es debido a que, en ciertas situaciones, no se tendrá una buena y rápida forma de obtener una aproximación del potencial electrostático. Pese a que la función de Green de Yukawa (sección 2.1.8.3) puede ser un método rápido, no es una buena aproximación para el potencial de todas las moléculas. Su diferencia con la solución real crece cerca de la interfaz para moléculas grandes y de alta carga. Si se considera un muestreo fijo para este término, los resultados pueden ser extrapolados a otros métodos de aproximación más costosos, pero que se realizarán solo en la primera iteración.
- Los términos de la Ley de Gauss y las mediciones experimentales corresponden a integrales. Esto implica que se emplean cuadraturas donde los puntos de evaluación tienen que quedar fijos. Además, estos puntos no deberían ser elegidos al azar ya que su elección influye en la aproximación de la integral. Para la integral de superficie de la Ley de Gauss se utilizarán los centroides de los elementos de generados en la superficie de la molécula.

Para las mediciones experimentales se generará una malla estructurada de volumen constante donde la evaluación será también el centroide. La cuadratura utilizada por ambos términos fue detallada en la sección 3.2.4.

3.5. Implementaciones Adicionales

3.5.1. Arquitecturas

Para las pruebas que se realizarán se utilizarán 3 arquitecturas. Se utilizará la *Multi Layer Perceptron* definida en la sección 2.2.1, y la ResNet también definida en 2.2.1. Por último, se probará con una arquitectura con buenos resultados para PINNs [33], a la que se llamará *Modified Multi-Layer Perceptron*, ModMLP.

La principal diferencia de esta arquitectura con la MLP radica en que se introducen 2 *encoders*, \mathbf{U} y \mathbf{V} , definidos de la siguiente forma:

$$\begin{aligned}\mathbf{U} &= \sigma^u(\mathbf{W}^u \mathbf{x} + \mathbf{b}^u) && \text{Encoder U} \\ \mathbf{V} &= \sigma^v(\mathbf{W}^v \mathbf{x} + \mathbf{b}^v) && \text{Encoder V}\end{aligned}\tag{3.65}$$

Estos *encoders* operan sobre cada capa oculta. De esta forma, la arquitectura ModMLP se expresa de la siguiente forma:

$$\begin{aligned}\mathbf{h}^1 &= \sigma^1(\mathbf{W}^1 \mathbf{x} + \mathbf{b}^1) && i = 1 \text{ capa entrada} \\ \mathbf{g}^i &= \sigma^i(\mathbf{W}^i \mathbf{h}^{i-1} + \mathbf{b}^i) && 2 \leq i \leq L \text{ capas ocultas} \\ \mathbf{h}^i &= \mathbf{g}^i \odot \mathbf{U} + (1 - \mathbf{g}^i) \odot \mathbf{V} \\ \phi_{\theta} &= \sigma^{L+1}(\mathbf{W}^{L+1} \mathbf{h}^L + \mathbf{b}^{L+1}) && i = L + 1 \text{ capa salida}\end{aligned}\tag{3.66}$$

Donde \odot corresponde a una multiplicación término a término. Notar que esta arquitectura generará un conjunto $\boldsymbol{\theta}$ mayor al de la MLP debido a las nuevas conexiones. Vale destacar que en ningún caso se usará una función de activación en la capa de salida.

3.5.1.1. Factorización Aleatoria de los Pesos de la Red

El *random weight factorization* [68] consiste en factorizar los pesos asociados a las conexiones de las neuronas de la siguiente forma, previo al entrenamiento.

$$\mathbf{W}^l = \text{diag}(\mathbf{s}^l) \cdot \mathbf{V}^l \quad (3.67)$$

Notar que cada matriz de pesos \mathbf{W}^l es factorizada por un vector \mathbf{s}^l y una matriz \mathbf{V}^l . Para obtener esta factorización, \mathbf{s} se inicializa como $\exp(\mathbf{u})$ en donde \mathbf{u} se obtiene de una distribución normal: $\mathbf{u} \sim \text{Normal}(\mu, \sigma^2)$ con media μ y desviación estándar σ .

Por cada matriz \mathbf{W} que se tenía anteriormente, ahora se tendrá un vector \mathbf{s} y una matriz \mathbf{V} . Posterior a esta factorización, el conjunto de parámetros $\boldsymbol{\theta}$ tendrá la siguiente forma (ejemplo para MLP):

$$\boldsymbol{\theta} = \{\mathbf{s}^1, \mathbf{V}^1, \mathbf{b}^1, \dots, \mathbf{s}^{L+1}, \mathbf{V}^{L+1}, \mathbf{b}^{L+1}\} \quad (3.68)$$

Esta factorización será implementada para verificar su impacto en el método a estudiar.

3.5.1.2. Capas de Escalamiento

Se detallan 2 capas de escalamiento para normalizar los inputs y outputs de la red neuronal, mejorando su convergencia en el entrenamiento. Esto permite que entren y salgan de la red valores entre -1 y 1.

El escalamiento del input se realiza previo al paso de las capas ocultas. Un ejemplo de la operación que ocurre en esta capa se detalla a continuación, donde valores \mathbf{x}_{min} y \mathbf{x}_{max} corresponden a hiperparámetros de cada red y dependen exclusivamente del dominio del problema.

$$\mathbf{h} = 2 \frac{(\mathbf{x} - \mathbf{x}_{min})}{(\mathbf{x}_{max} - \mathbf{x}_{min})} - 1 \quad (3.69)$$

El escalamiento del output se realiza posterior al paso de las capas ocultas. Un ejemplo de la operación que ocurre en esta capa se detalla a continuación, donde los valores \mathbf{y}_{min} e \mathbf{y}_{max} corresponden a hiperparámetros de cada red y se deben estimar en base a aproximaciones de la solución real en cada dominio.

$$\mathbf{y} = \frac{\mathbf{h} + 1}{2} (\mathbf{y}_{max} - \mathbf{y}_{min}) + \mathbf{y}_{min} \quad (3.70)$$

3.5.1.2.1 Estimación de las Cotas del Potencial

Para estimar los valores de \mathbf{y}_{min} e \mathbf{y}_{max} , se realizará una aproximación del potencial mediante la superposición de la solución conocida del Ion de Born (sección 2.1.8.1), asumiendo que cada carga de la macromolécula es un Ion de Born independiente. Lo anterior no implica que la capa de escalamiento escale la solución entre -1 y 1, pero será una aproximación para trabajar con órdenes cercanos. Si se define la función BORN como:

$$\text{BORN}(q_i, R_i) = \frac{q_i}{4\pi} \left(\frac{1}{\epsilon_2(1 + \kappa R_i)R_i} - \frac{1}{\epsilon_1 R_i} \right) \quad (3.71)$$

El potencial de reacción mínimo y máximo (los cuales se usarán para obtener \mathbf{y}_{min} e \mathbf{y}_{max}) se podrá estimar con las siguientes ecuaciones, siendo $R'_{ji} = \|\mathbf{x}_i - \mathbf{x}_j\|$ con \mathbf{x}_j la posición de la carga puntual q_j .

$$\begin{aligned} \psi_{max} &= \max_i \left(\text{BORN}(q_i, R_i), \text{BORN}(q_i, R_i) + \sum_{j \neq i} \text{BORN}(q_j, R'_{ji}) \right) \\ \psi_{min} &= \min_i \left(\text{BORN}(q_i, R_i), \text{BORN}(q_i, R_i) + \sum_{j \neq i} \text{BORN}(q_j, R'_{ji}) \right) \end{aligned} \quad (3.72)$$

Lo anterior se puede visualizar de manera gráfica en la siguiente figura.

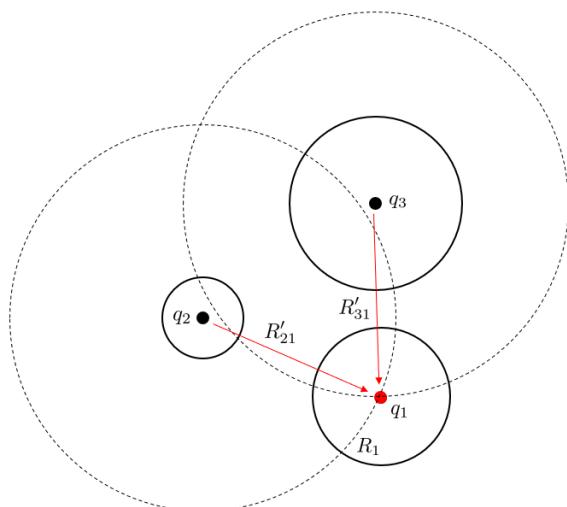


Figura 3.5: Esquema de la superposición de las distintas cargas puntuales

3.5.1.3. Capa de Fourier

Para minimizar un fenómeno llamado *bias espectral*, el cual produce bias en el aprendizaje de funciones de baja frecuencia, se tiende a agregar una capa de *Random Fourier Features* [69]. Esta capa es simplemente un mapeo hacia señales de alta frecuencia, que se realiza antes de ingresar a la red neuronal, posterior al escalamiento. El mapeo de esta capa se muestra a continuación:

$$\mathbf{h} = \begin{bmatrix} \cos(\mathbf{Bx}) \\ \sin(\mathbf{Bx}) \end{bmatrix} \quad (3.73)$$

En donde la matriz \mathbf{B} se crea obteniendo una muestra de una distribución Gaussiana de la forma: $\mathbf{B} \sim \text{Normal}(0, \sigma^2)$ (no entrenables). La incorporación de esta capa previa a la red permite mejorar bastante la representación para grandes gradientes y soluciones complejas.

3.5.1.4. Función de Activación Entrenable

Para una mejor representación de la solución, se le pueden añadir a las funciones de activación parámetros entrenables [70], los cuales pertenecerán al conjunto de parámetros $\boldsymbol{\theta}$ de la red neuronal. Un ejemplo sencillo, sería utilizar la función de activación:

$$\sigma(x) = \tanh(ax) \quad (3.74)$$

En donde el parámetro a es entrenable, y se debe incluir en el conjunto de parámetros $\boldsymbol{\theta}$.

$$\boldsymbol{\theta} = \{\mathbf{W}^1, \mathbf{b}^1, \mathbf{a}^1, \dots, \mathbf{W}^{L+1}, \mathbf{b}^{L+1}, \mathbf{a}^{L+1}\} \quad (3.75)$$

En la ecuación anterior \mathbf{a}^i corresponde al vector de parámetros entrenables para las funciones de activación de la capa i -ésima.

3.5.2. Algoritmo de Ponderación para la Función de Pérdida

Debido a la gran cantidad de términos distintos que tiene la función de pérdida, algunos pueden ponderar más que otros, predominando en el proceso de optimización. Si no se aplica ninguna corrección a este fenómeno, pueden haber términos que nunca lleguen a minimizarse.

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = \sum_j w_j \mathcal{L}_j(\boldsymbol{\theta}; \mathcal{S}_j) \quad (3.76)$$

3. Metodología

Es por esto que se agregará un algoritmo de ponderación de los términos \mathcal{L}_j a partir de una actualización automática de los pesos w_j [33].

La idea del algoritmo se basa en seleccionar pesos w_j de tal forma que todos los gradientes de los distintos términos \mathcal{L}_j tengan el mismo valor.

$$C = w_j \|\nabla_{\theta} \mathcal{L}_j\| \quad \forall j \quad (3.77)$$

Para esto, se calcula el estimador \hat{w}_j para cada término como:

$$\hat{w}_j = \frac{\sum_i \|\nabla_{\theta} \mathcal{L}_i\|}{\|\nabla_{\theta} \mathcal{L}_j\|} \quad (3.78)$$

Y luego, se corrige con la media móvil utilizando el peso anterior $w_{j,\text{old}}$ y un parámetro α a definir.

$$w_{j,\text{new}} = \alpha w_{j,\text{old}} + (1 - \alpha) \hat{w}_j \quad (3.79)$$

Este algoritmo se utilizará cada 1000 iteraciones utilizando un parámetro $\alpha = 0.7$.

Algoritmo 3.2 Algoritmo para ponderación de pesos

Input: Funciones de pérdida \mathcal{L}_j , pesos $w_{j,\text{old}}$, parámetro α .

```

 $M = \sum_i \|\nabla_{\theta} \mathcal{L}_i\|$ 
for  $j = 1$  to  $j = N_{\mathcal{L}}$  do
     $\hat{w}_j = \frac{M}{\|\nabla_{\theta} \mathcal{L}_j\|}$ 
     $w_{j,\text{new}} = \alpha w_{j,\text{old}} + (1 - \alpha) \hat{w}_j$ 
end for
```

Output: Pesos $w_{j,\text{new}}$

3.5.3. Adimensionalización

Adimensionalizar la ecuación a resolver ha demostrado buenos resultados en otros tipos de ecuaciones al compararlos con casos no adimensionales. Es por esto que se trabajará la ecuación de Poisson-Boltzmann en una de sus formas adimensionales.

Al aplicar esta adimensionalización (tabla 3.1) a la ecuación de Poisson-Boltzmann no lineal,

Tabla 3.1: Adimensionalización Poisson-Boltzmann

Variable	Adimensionalización
Longitud	$\mathbf{r}^* = \frac{\mathbf{r}}{r_c}$ con $r_c = 1 \text{ [\AA]}$
Carga	$q^* = \frac{q}{q_e}$ con q_e la carga del electrón
Potencial	$\phi^* = \frac{\phi \epsilon_0 r_c}{q_e}$ con ϵ_0 la constante permitividad del vacío

se obtiene la siguiente expresión:

$$-\nabla^{*2}\phi^* + \kappa^{*2}\beta^* \sinh(\phi^*/\beta^*) = \frac{1}{\epsilon_1^*} \sum_k q_k^* \delta^*(\mathbf{r}^* - \mathbf{r}_k^*) \quad (3.80)$$

En donde la constante β^* se calcula como: $\beta^* = T\epsilon_0 r_c k_b / q_e^2$, y k_b corresponde a la constante de Boltzmann. Además, la constante dieléctrica y la longitud inversa de Debye adimensional se expresan de la siguiente forma:

$$\epsilon_i^* = \frac{\epsilon_i}{\epsilon_0} \quad i = 1, 2 \quad ; \quad \kappa^* = r_c \kappa \quad (3.81)$$

Asimismo, para la forma lineal de la ecuación de Poisson-Boltzmann, la ecuación 3.80 se reduce a lo siguiente:

$$-\nabla^{*2}\phi^* + \kappa^{*2}\phi^* = \frac{1}{\epsilon_1^*} \sum_k q_k^* \delta^*(\mathbf{r}^* - \mathbf{r}_k^*) \quad (3.82)$$

De esta manera, la ecuación de Poisson-Boltzmann será trabajada con las variables adimensionales \mathbf{r}^* , q^* y ϕ^* , y las constantes adimensionales β^* (caso no lineal), ϵ_i^* y κ^* .

3.6. Validación de Resultados

3.6.1. Función de Pérdida de Entrenamiento y de Validación

Una buena validación de los resultados es revisar directamente la evaluación de la función de pérdida (loss) en los puntos de colocación. Esto va permitir revisar si se están minimizando los residuales, y la continuidad en la interfaz, por ejemplo. Sin embargo, este no puede ser el único indicador, dado que no asume una generalización de la solución a todo el dominio.

Debido a lo anterior, se tomará un conjunto de puntos de validación $\mathcal{S}^v = \mathcal{S}_{\Omega_1}^v \cup \mathcal{S}_{\Omega_2}^v \cup \mathcal{S}_{\Gamma}^v \cup \mathcal{S}_{\partial\Omega}^v$. En estos puntos solo se calculará la función de pérdida \mathcal{L} , pero no se realizará ningún paso de descenso de gradiente u optimización. De esta manera, se podrá revisar como evoluciona la

3. Metodología

función de pérdida en puntos donde no se está realizando el entrenamiento, dando un indicador de la generalización de la solución. Estos puntos de validación serán elegidos utilizando el mismo muestreo aleatorio definido en la sección 3.4, y se mantendrán fijos durante todas las iteraciones. De los términos de la función de pérdida definidos, solo se utilizará como validación el cálculo de los residuales, condiciones en la interfaz y la condición de borde.

3.6.2. Energía de Solvatación

Un buen indicador para validar el código respecto a la teoría física es la predicción de la energía de solvatación. La definición de la energía de solvatación se puede revisar en la sección 2.1.6. Los resultados obtenidos mediante el código de PINNs serán comparados con los resultados obtenidos con un código de BEM llamado PBJ, un código de FDM llamado APBS, y/o la solución analítica si se tiene disponible. El software de PBJ [71] utiliza el método de elementos de frontera, resolviendo la ecuación de Poisson-Boltzmann en su forma integral. Por otra parte, el software de APBS [72] utiliza el método de diferencias finitas, resolviendo en todo el dominio (sólido y solvente). Ambos softwares han demostrado en numerosos casos buenas predicciones de la energía de solvatación, incluso al compararlas con mediciones experimentales.

Para el código de PINNs, la forma de calcular la energía de solvatación dependerá de que forma de la ecuación de Poisson-Boltzmann se esté resolviendo (directa o esquemas de regularización). Estos se detallarán a continuación:

- **Formulación directa:** Según la ecuación 2.54, la energía de solvatación se calcula utilizando el potencial de reacción en la ubicación de las cargas. Debido a que para la formulación directa se utiliza una función Gaussiana para aproximar el delta de Dirac, el campo de reacción calculado mediante la resta de la predicción de la red neuronal ϕ_θ y el potencial de Coulomb tenderá a infinito. Es por esto que se tendrá que utilizar otra forma para calcular este potencial.

A partir de las ecuaciones integrales en la frontera (sección 2.1.5), se puede utilizar la predicción de la red neuronal en la interfaz de la molécula para calcular el potencial de reacción en la posición de las cargas. Esta ecuación se visualiza a continuación:

$$\psi_\theta(\mathbf{x}_k) = \frac{1}{4\pi\epsilon_1} \oint_{\Gamma} \overline{\epsilon \partial_n \phi_\theta} \frac{1}{|\mathbf{x}_k - \mathbf{x}'|} dS(\mathbf{x}') - \frac{1}{4\pi} \oint_{\Gamma} \overline{\phi_\theta} \frac{\partial}{\partial \mathbf{n}} \left(\frac{1}{|\mathbf{x}_k - \mathbf{x}'|} \right) dS(\mathbf{x}') \quad (3.83)$$

En donde \mathbf{x}_k corresponde a la posición de las cargas puntuales. Notar que la ecuación anterior utiliza el promedio de la predicción de las 2 redes para potencial electrostático

$\overline{\phi_\theta}$ y el promedio para la predicción del desplazamiento eléctrico $\overline{\epsilon\partial_n\phi_\theta}$.

$$\begin{aligned}\overline{\phi_\theta} &= \frac{\phi_\theta^1 + \phi_\theta^2}{2} \\ \overline{\epsilon\partial_n\phi_\theta} &= \frac{1}{2} \left(\epsilon_1 \frac{\partial\phi_\theta^1}{\partial\mathbf{n}} + \epsilon_2 \frac{\partial\phi_\theta^2}{\partial\mathbf{n}} \right)\end{aligned}\quad (3.84)$$

De esta manera, con el cálculo de ψ_θ por la ecuación integral 3.83, la energía de solvatación puede ser calculada como:

$$\Delta G_{solv}^\theta = \frac{1}{2} \sum_k q_k \psi_\theta(\mathbf{x}_k) \quad (3.85)$$

- **Esquemas de regularización 1 y 2:** Para ambos esquemas, la red neuronal entregará el potencial de reacción en la región del soluto. Debido a esto, se puede utilizar directamente el output de la red neuronal para el cálculo de la energía de solvatación a partir de la siguiente ecuación:

$$\Delta G_{solv}^\theta = \frac{1}{2} \sum_k q_k \psi_\theta(\mathbf{x}_k) \quad (3.86)$$

Con la predicción de la energía de solvatación, se calculará el error relativo respecto a la solución de referencia como:

$$\mathcal{E}_{G_{solv}} = \left| \frac{\Delta G_{solv}^\theta - \Delta G_{solv}^\dagger}{\Delta G_{solv}^\dagger} \right| \quad (3.87)$$

En donde ΔG_{solv}^θ y ΔG_{solv}^\dagger hacen referencia a la predicción de PINNs y a la de referencia respectivamente, y $\mathcal{E}_{G_{solv}}$ al error relativo en la energía de solvatación.

3.6.3. Potencial de Reacción

Las predicciones del potencial de la red neuronal también serán comparados con la solución que se tenga disponible. Al igual que para la energía de solvatación, se podrá utilizar la solución analítica en caso de que se tenga, o la de un método numérico como BEM o FDM a partir de los softwares de PBJ o APBS respectivamente.

Para esta comparación, se calculará el error relativo L_2 para la predicción del potencial de reacción en la interfaz de la molécula. Se elige esta zona ya que la interfaz es una región de interés en la electrostática y además se tiene la continuidad de ambas redes. El cálculo del error L_2 se muestra a continuación.

$$\mathcal{E}_\psi = \sqrt{\frac{\sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} [\bar{\psi}_\theta(\mathbf{x}_i) - \psi^\dagger(\mathbf{x}_i)]^2}{\sum_{\mathbf{x}_i \in \mathcal{S}_\Gamma} [\psi^\dagger(\mathbf{x}_i)]^2}} \quad (3.88)$$

En donde $\bar{\psi}_\theta$ y ψ^\dagger hacen referencia a la predicción de PINNs y a la de referencia para el potencial de reacción respectivamente, y \mathcal{E}_ψ al error relativo L_2 en la interfaz. Notar que se utiliza $\bar{\psi}_\theta$ para remarcar que se calcula el promedio de la predicción entre ambas redes.

Adicionalmente, se graficará el potencial de reacción en la interfaz de la molécula, y en los ejes de relevancia, para así revisar el comportamiento del potencial a lo largo del soluto y solvente.

3.7. Código Implementado

El código implementado se adaptó como una librería llamada XPPBE, la cual puede ser utilizada para resolver la ecuación de Poisson-Boltzmann para la electrostática molecular. El código completo puede ser revisado en el repositorio de GitHub [9]. XPPBE permite poder ingresar la información molecular a partir de un archivo `.pqr` (o `.pdb`), el cual tiene toda la información de la posición, carga y radio de todos los átomos constituyentes. A partir de este archivo, se generan las mallas y se comienza a resolver la ecuación de Poisson-Boltzmann utilizando PINNs. Todos los parámetros para la resolución utilizando PINNs (mallas, métodos, hiperparámetros, etc.) son definidos en un archivo `.yaml` (código B.2).

El uso de la librería es bastante sencillo. Un ejemplo de su uso se puede revisar en el siguiente bloque de código.

Código 3.1: Ejemplo de uso básico de la librería XPPBE

```
from xppbe import Simulation
simulation = Simulation(yaml_path, molecule_dir)
simulation.create_simulation()
simulation.adapt_model()
simulation.solve_model()
simulation.postprocessing(run_all=True)
```

Para resolver la ecuación de Poisson-Boltzmann, basta especificar la ruta hacia el archivo `.yaml` y el directorio donde se encontrará el archivo `.pqr` o `.pdb`. Para mayor detalle del funcionamiento y método de uso, revisar la sección B y/o el repositorio de GitHub [9].

XPPBE: PINN Solver for 3D Poisson-Boltzmann Equation

 XPPBE version 1.0.0 build passing python >=3.9,<3.10

Physics-Informed Neural Network solver for the Poisson-Boltzmann equation applied to real macromolecules in polarizable media.

$$\nabla^2 \phi_1 = -\frac{1}{\epsilon_1} \sum_k q_k \delta(x_k) \quad x \in \Omega_1$$

$$\nabla^2 \phi_2 = \kappa^2 \phi_2 \quad x \in \Omega_2$$

Figura 3.6: Repositorio de GitHub

La librería sigue los mismos métodos definidos en este documento. Un ejemplo del algoritmo se puede visualizar a continuación:

Algoritmo 3.3 Algoritmo del código implementado con PINNs: XPPBE

Input: Información molecular (archivo `.pqr`) y variables entrada definidas en archivo `.yaml`

Generar malla de la molécula a partir de la información molecular

Generar mallas volumétricas y malla superficial del contorno

Inicializar red $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$ a partir de los hiperparámetros definidos

for $k = 1$ to $k = n$ **do** ▷ Loop de entrenamiento

 Obtener una muestra \mathcal{S} de puntos aleatorios en el dominio ▷ Sección 3.4

 Calcular $\phi_\theta = \mathcal{N}(\mathcal{S}; \boldsymbol{\theta}_k)$ ▷ Sección 3.2

 Calcular $\partial_n \phi_\theta, \nabla \phi_\theta, \nabla^2 \phi_\theta$ con diferenciación automática ▷ Sección 3.3

 Calcular $\mathcal{L}(\boldsymbol{\theta}_k; \mathcal{S})$ ▷ Sección 3.2

 Calcular validación $\mathcal{L}(\boldsymbol{\theta}_k; \mathcal{S}^v)$ ▷ Sección 3.6.1

 Actualizar $\boldsymbol{\theta}_{k+1}$ con método de optimización usando $\boldsymbol{\theta}_k$ y $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ ▷ Sección 2.2.3

if $\text{mod}(k, r) == 0$ **then** ▷ Sección 3.5.2

 Actualizar pesos w_j asociado a cada \mathcal{L}_j ▷ Sección 3.5.2

end if ▷ Sección 3.5.2

end for

Output: Parámetros $\boldsymbol{\theta} = \boldsymbol{\theta}^1 \cup \boldsymbol{\theta}^2$ para post-procesamiento.

Como se visualiza en el algoritmo anterior, a partir de la información molecular y los parámetros a definir por el usuario, se obtiene el conjunto de parámetros $\boldsymbol{\theta}$, de tal forma que la red neuronal \mathcal{N} replique la solución de la ecuación de Poisson-Boltzmann.

3. Metodología

La librería se programó utilizando el siguiente árbol de directorios:

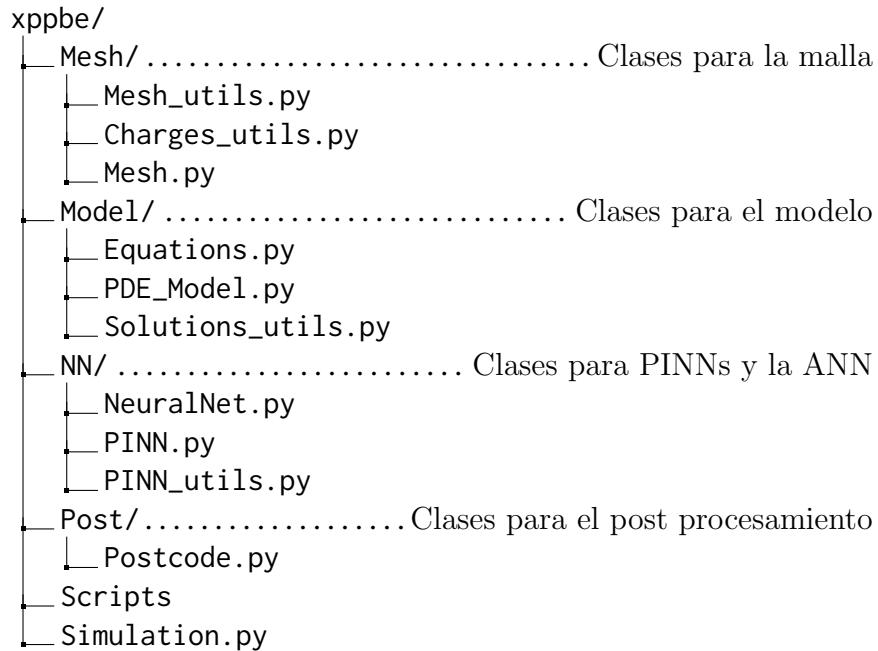


Figura 3.7: Estructura de archivos para el código implementado

Todas las clases que se visualizan: mallas, modelo, PINNs, ANN y post-procesamiento, son controladas por la clase `Simulation`.

3.7.1. Softwares y Librerías Utilizadas

El código completo de PINNs fue escrito en su totalidad en Python [73] debido a las librerías existentes para programar redes neuronales con este lenguaje. El código de XPPBE utiliza las siguientes librerías y softwares para su funcionamiento:

- Tensorflow [74]: Corresponde a una interfaz para programar y optimizar redes neuronales. Esta librería permite customizar la red y la función de pérdida dependiendo del problema. Además viene con el algoritmo de diferenciación automática y de back propagation.
- pdb2pqr [75]: Se utiliza para transformar los archivos .pdb extraídos del *Protein Data Bank* [76] a formatos .pqr.
- MSMS [77]: Se utiliza para generar la malla superficial SES de la molécula a partir de un archivo .xyzr (modificación del .pqr).
- NanoShaper [78]: Se utiliza para generar la malla superficial SES de la molécula a partir de un archivo .xyzr.
- Trimesh [79]: Se utiliza para el manejo de mallas superficiales creadas por MSMS o Na-

noShaper.

- PyGamer [80]: Se utiliza para la creación y manejo de mallas volumétricas a partir de mallas superficiales creadas por MSMS o NanoShaper.
- Scipy [81]: Se utiliza esta librería por los algoritmos y expresiones de computación científica que tiene implementado.
- Bempp [82]: Se utiliza para el cálculo de las integrales de superficie con singularidades, necesarias para el cálculo de la energía de solvatación para la ecuación no regularizada.
- Matplotlib [83] y Plotly [84]: Ambas librerías se utilizan para la visualización de información a partir de la creación de gráficos.

Capítulo 4

Resultados y Análisis

En esta sección se presentarán todos los resultados obtenidos para las distintas moléculas utilizando la librería implementada de XPPBE, para resolver la ecuación de Poisson-Boltzmann utilizando PINNs.

4.1. Pruebas Realizadas

Se entregará un resumen de las pruebas realizadas:

- Ion de Born: El Ion de Born se utilizó para validar el método implementado debido a su simplicidad y su solución analítica disponible. A partir de estos resultados, se podrá seleccionar parámetros importantes del código que se mantendrán con el resto de moléculas. Las pruebas para el Ion de Born corresponden a las siguientes:
 - Revisión de las distintas formulaciones de la ecuación de Poisson-Boltzmann. Considerando la formulación directa y los 2 esquemas de regularización. Además, se mostrará el impacto de la descomposición en 2 dominios para resolver la PBE.
 - Búsqueda de los mejores hiperparámetros para la red neuronal. Esto considera la arquitectura, cantidad de capas ocultas y cantidad de neuronas por capa. Además, se revisa el uso de la capa de Fourier, capa de escalamiento y función de activación entrenable.
 - Búsqueda de la mejor cantidad de puntos de colocación. Se revisa el impacto de la cantidad de puntos a utilizar por iteración (*batch size*), y el impacto del muestreo aleatorio en comparación al muestreo fijo.
 - Revisión de las distintas combinaciones de los términos de la función de pérdida.

En estas pruebas se revisa el impacto de la adición del término correspondiente a la Ley de Gauss, los datos experimentales y utilizar datos puntuales conocidos en el dominio. Además, se revisa el impacto de la utilización del algoritmo de ponderación de los términos de la función de pérdida.

- Revisión de la combinación de métodos de optimización (ADAM y L-BFGS).
- Multipolo esférico: Análogo al Ion de Born, debido a que se cuenta con la solución analítica, se pretende revisar el efecto de agregar más cargas puntuales a la molécula esférica. Además, se revisa si la carga neta, o las posiciones, afectan en la precisión del método.
- Moléculas reales: Con las validaciones realizadas para el Ion de Born y el multipolo esférico, se avanzó hacia moléculas reales. En estas pruebas se revisa si los resultados obtenidos anteriormente son extrapolables a geometrías complejas y con una mayor cantidad de cargas puntuales. Dentro de las moléculas que se mostrarán se encuentran:
 - Metanol.
 - Arginina.
 - Ubicuitina [85].
 - ADN extraído del complejo Homeodomino Antennapedia [86].

Hay que destacar que para el ADN se realizará una comparación de lo obtenido utilizando la formulación no lineal de PBE, para así evaluar la precisión de PINNs al realizar esta modificación.

Como se detalló en la sección 3.6, se utilizan los siguientes indicadores y/o gráficos para evaluar la solución obtenida:

- Predicción de la energía de solvatación ΔG_{solv}^θ y error relativo respecto a solución de referencia $\mathcal{E}_{G_{solv}}$. En los casos que se utilice un método numérico para obtener la solución de referencia, se realiza una convergencia de malla para obtener la predicción “real” de la energía de solvatación, para así calcular el error $\mathcal{E}_{G_{solv}}$.
- Predicción del potencial de reacción en la superficie de la molécula, y su error relativo L_2 respecto a la solución de referencia \mathcal{E}_ψ . En los casos que se utilice BEM para la solución de referencia, se calcula el potencial utilizando la misma malla superficial que PINNs, para el cálculo del error \mathcal{E}_ψ .
- Grafica del potencial de reacción en distintos ejes y en la interfaz.
- Cálculo del loss de entrenamiento \mathcal{L} y loss de validación \mathcal{L}^v .

4. Resultados y Análisis

Por último, para todas las simulaciones se tuvieron las siguientes consideraciones, salvo que se especifique lo contrario:

- Se resolvió la forma lineal de la ecuación de Poisson-Boltzmann en todos los casos, excepto para el ADN sección 4.7.
- Como base, se utilizaron los siguientes términos de la función de pérdida:
 - Residuales en soluto y solvente (denotados como R1 y R2).
 - Continuidad en el potencial y en el desplazamiento eléctrico en la interfaz (denotados como I_u e I_d).
 - Condición de borde del solvente (denotado como D2).

Se especificará el caso en que se agreguen más términos.

- Para la obtención de puntos de colocación se utilizó el método de muestreo aleatorio, salvo en la sección 4.2.2.
- El esquema de inicialización de los pesos de las redes neuronales es *Glorot Normal*.
- Para todas las funciones de activación se utilizó la tangente hiperbólica, especificando cuando se utilicen parámetros entrenables.
- En los casos que se utilizó la capa de Fourier, se fijó una desviación estándar $\sigma = 1$, y un número de *features* de 128.
- En los casos que se utilizó la factorización de los pesos de la red, se fijó una media de $\mu = 1$ y una desviación estándar $\sigma = 0.1$.
- Como método de optimización, en todos los casos se utilizó el método de ADAM, con una tasa de aprendizaje con decaimiento exponencial partiendo desde 10^{-3} y con una tasa de decaimiento de 0.9 cada 2000 iteraciones, salvo cuando se especifique el uso del algoritmo L-BFGS.
- En los casos que se utilizó el algoritmo de ponderación de la función de pérdida, se implementó cada 1000 iteraciones utilizando un parámetro $\alpha = 0.7$.
- Se utilizaron 20000 iteraciones para el proceso de entrenamiento, salvo para las pruebas de la ubicuitina y el ADN, en donde se utilizaron 40000.

4.2. Ion de Born

El Ion de Born corresponde una molécula esférica con una carga puntual en el centro. Esta molécula servirá como una primera validación de resultados ya que se conoce la solución analítica, la cual fue presentada en la ecuación 2.60.

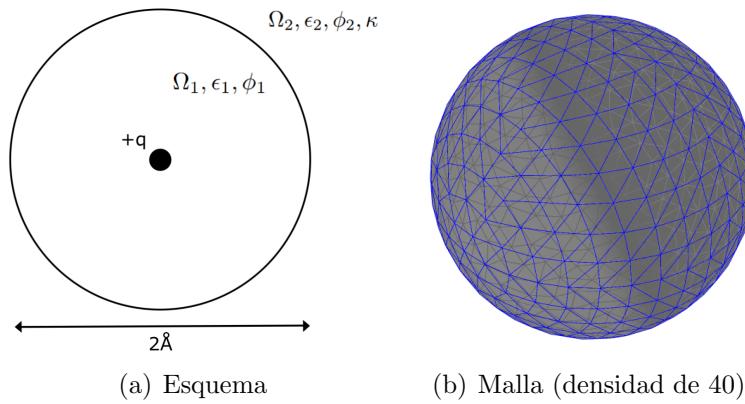


Figura 4.1: Esquema y malla para el Ion de Born

Para todas las simulaciones del Ion de Born, se utilizaron las siguientes dimensiones y/o propiedades, excepto cuando se diga lo contrario:

- Carga en el centro: $q = +1 [q_e]$
- Constante dieléctrica soluto: $\epsilon_1 = 1 [\epsilon_0]$
- Constante dieléctrica solvente: $\epsilon_2 = 80 [\epsilon_0]$
- Inverso de la longitud de Debye: $\kappa = 0.125 [1/\text{\AA}]$
- Temperatura: 300 [K]
- Radio molécula: $R = 1 [\text{\AA}]$
- Radio esfera frontera solvente: $R = 4 [\text{\AA}]$

En las siguientes secciones se resolverá la ecuación de Poisson-Boltzmann lineal para el Ion de Born en distintas situaciones. Estos casos revelarán como afecta el esquema de la ecuación, la cantidad de puntos de colocación, las arquitecturas de las redes neuronales, la combinación de métodos de optimización, y los términos de la función de pérdida, en la convergencia y precisión del método de PINNs.

4.2.1. Estudio de Esquemas y Descomposición de Dominios

4.2.1.1. Uso de la Descomposición de Dominios

El primer estudio que se realizó fue la variación de la formulación de la ecuación de Poisson-Boltzmann utilizando 2 dominios. Respecto a esto, se probó la formulación directa, y los esquemas de regularización 1 y 2 presentados en la sección 3.2.

Para los 3 casos se utilizaron 2 redes iguales (una para cada dominio) de arquitectura MLP, con 4 capas ocultas, 200 neuronas por capa, con capa de escalamiento del input y capa de Fourier. Además, se utilizó datos puntuales en el solvente (sección 3.2.4), dados por la función de Green de Yukawa.

Para los puntos de colocación, se utilizó el método de muestreo aleatorio definido en la sección 3.4 entregando las siguientes cantidades:

Tabla 4.1: Puntos de colocación para el estudio de esquemas¹

Nodos R1	Nodos R2	Nodos I	Nodos D2	Nodos K2	Nodos Q
3233	11146	1216	1280	300	100

A continuación, se presentan los resultados obtenidos.

Tabla 4.2: Resultados para el estudio de esquemas utilizando descomposición de dominios

Esquema PBE	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Directa	-0.11	9.99E-01	1.01E-02	5.96E+01	1.30E+02
Reg. 1	-165.37	7.19E-03	7.55E-02	1.26E-03	7.47E-04
Reg. 2	-165.04	5.20E-03	3.37E-03	6.15E-06	1.04E-05

En la tabla 4.2 se puede visualizar que el esquema de regularización 2 produce los errores más bajos en todos los indicadores, considerando el error en la energía de solvatación, el potencial en la interfaz, y los losses.

Además, es importante destacar que la formulación directa de la ecuación de Poisson-Boltzmann no es la apropiada para el método utilizado. La energía de solvatación está totalmente desfasada del valor teórico. Esto se puede deber a que se resuelve la ecuación considerando las singularidades producto de las cargas puntuales, lo que disminuye la convergencia del método.

¹Para todos los términos se utilizó el método de muestreo aleatorio, excepto para K2 (datos puntuales en el solvente), el cual se mantuvo fijo en todo el entrenamiento. Nodos Q (puntos cerca de las cargas) solo fue utilizado en la formulación directa.

En la figura 4.2 se pueden visualizar las predicciones del potencial de reacción de los 3 esquemas al compararlos con la solución analítica. En estos se nota que claramente la formulación directa es incapaz de representar el potencial en el soluto debido a la aproximación de las cargas puntuales. Sin embargo, logra una muy buena representación en la zona del solvente. Por otra parte, ambos esquemas de regularización estudiados logran replicar la solución de la ecuación de PB en todo el dominio, donde se nota que el esquema 2 logra una mejor representación sin desviaciones, a diferencia del esquema 1.

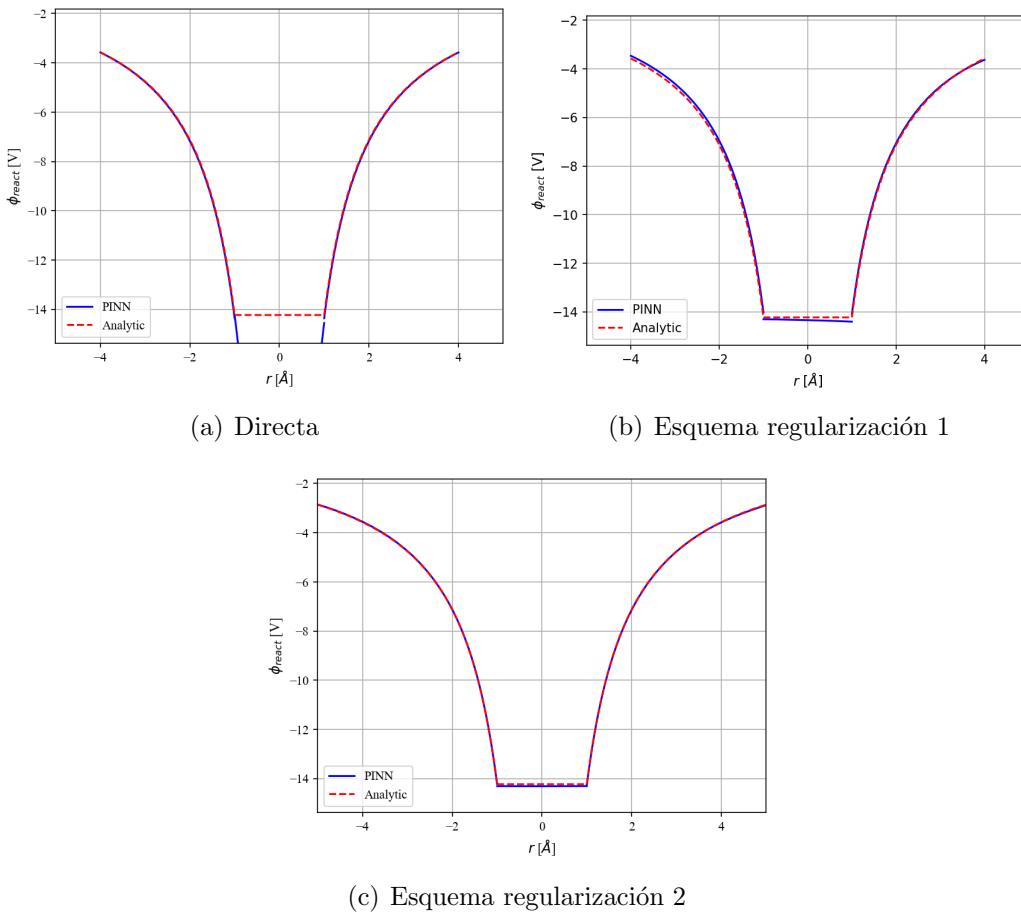


Figura 4.2: Potencial de reacción para los distintos esquemas de la ecuación de PB

Lo dicho anteriormente para el potencial se confirma para la energía de solvatación. En la figura 4.3 se puede visualizar como ambos esquemas de regularización convergen al valor correcto de la energía de solvatación, mientras que la formulación directa se estabiliza en un valor incorrecto.

Es importante destacar que el cálculo de la energía de solvatación para la formulación directa requiere de una buena predicción del gradiente del potencial en la interfaz para utilizar la ecuación 3.83. Esto agrega dificultades al método en comparación a los esquemas de regularización en donde se puede utilizar directamente la predicción de la red neuronal en la posición

4. Resultados y Análisis

de las cargas. Esto demuestra aún más la ventaja que tiene utilizar la forma regularizada de la ecuación de PB.

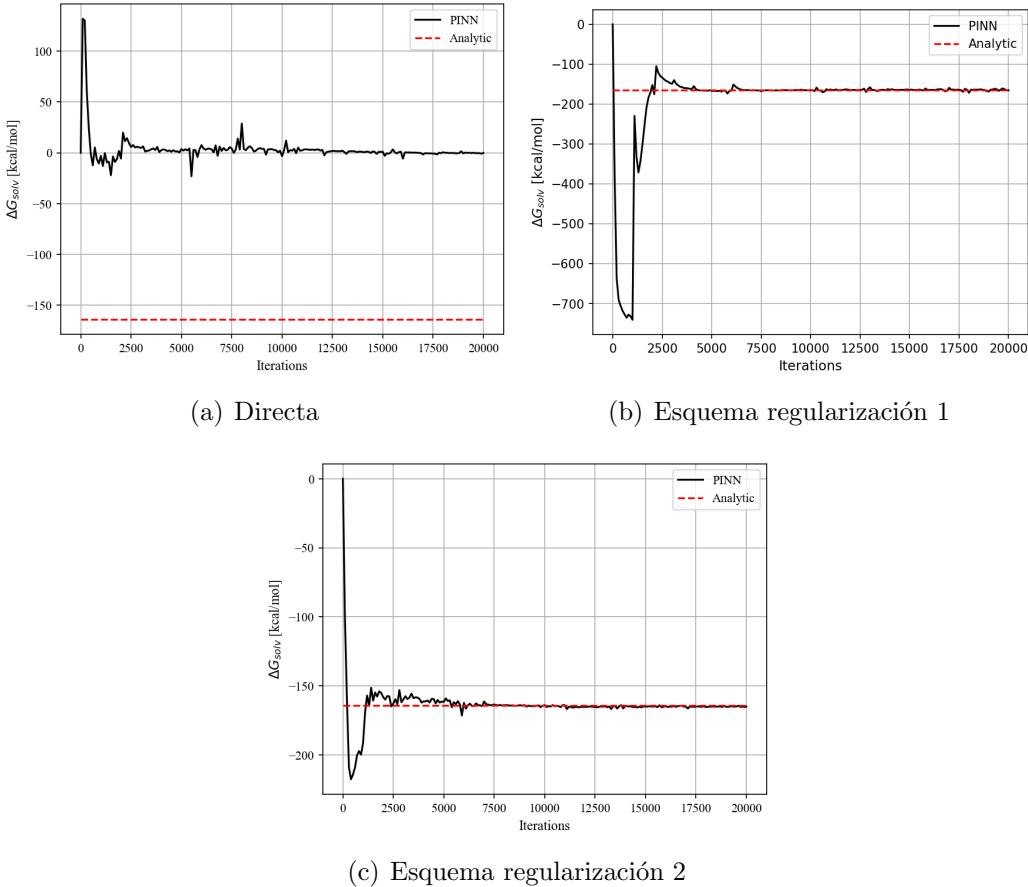


Figura 4.3: Energía de solvatación para los distintos esquemas de la ecuación de PB

4.2.1.2. Sin Descomposición de Dominios

En esta sección se presentará el mismo estudio de esquemas que la sección anterior, pero utilizando solo 1 red neuronal para todo el dominio. Esto implica que no se tendrán los términos de la interfaz en la función de pérdida (se cumplen con los residuales de la PBE). Además, se utilizaron los mismos puntos de colocación que en el caso de 2 dominios, tabla 4.1.

Tabla 4.3: Resultados para el estudio de esquemas sin descomposición de dominios

Esquema PBE	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Directa	-71.79	5.63E-01	8.88E-03	2.64E+02	2.17E+02
Reg. 1	-41.36	7.48E-01	1.60E+01	1.56E-04	9.13E-08
Reg. 2	0.38	1.00E+00	4.96E-01	6.30E-04	7.27E-04

En la tabla 4.3 se puede visualizar como ninguno de los 3 esquemas logra una buena predicción de la energía de solvatación. Incluso, se visualizan errores más altos comparados con el uso de descomposición de dominios.

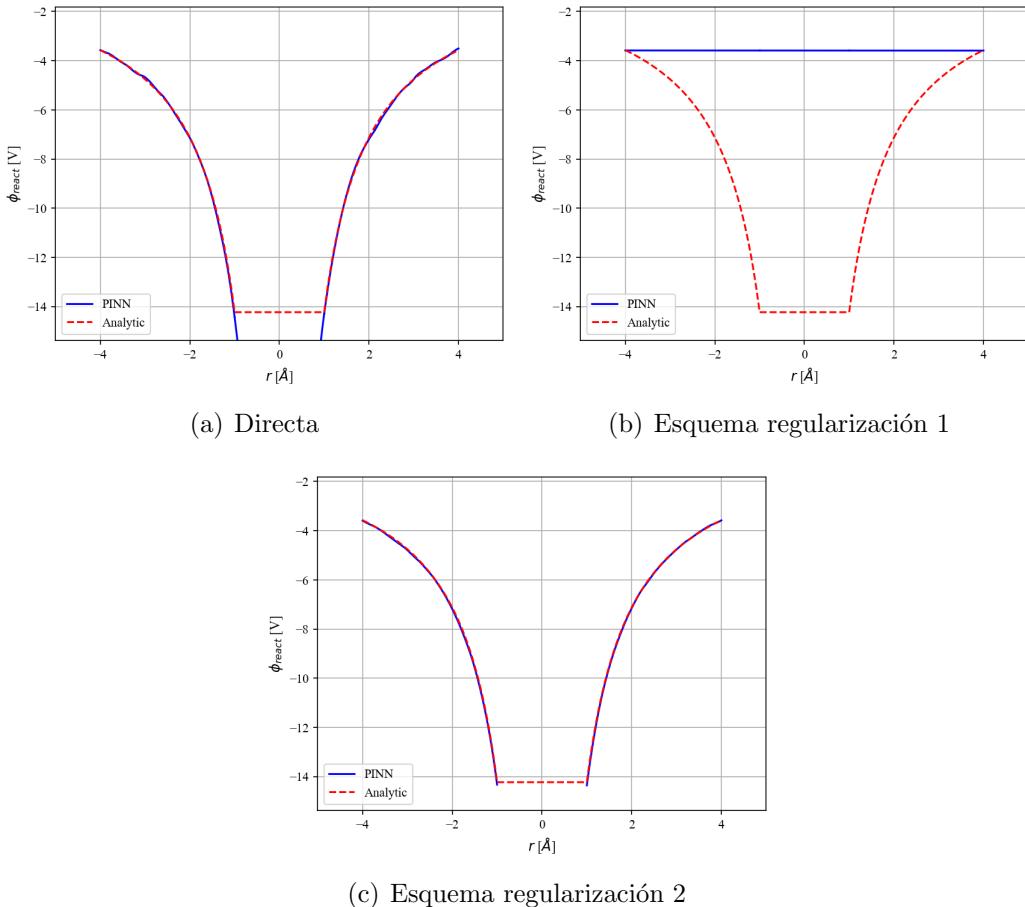


Figura 4.4: Potencial de reacción para los distintos esquemas de la ecuación de PB

En la figura 4.4 se grafica el potencial de reacción obtenido con los 3 esquemas estudiados. De estos gráficos, se nota que en ningún caso se pudo replicar el potencial en el interior de la molécula esférica. Sin embargo, se logra una buena representación en la zona del solvente para la formulación directa y el esquema de regularización 2.

Es importante destacar que no se encuentra óptimo utilizar 1 red para el esquema de regularización 2. Esto se debe a que las redes neuronales entregan un resultado continuo, cuestión que será imposible dado que este esquema genera una discontinuidad en el output de la red (potencial de reacción en el soluto y potencial total en el solvente). Esto puede ejemplificarse en la desviación en la predicción del potencial en el soluto.

4. Resultados y Análisis

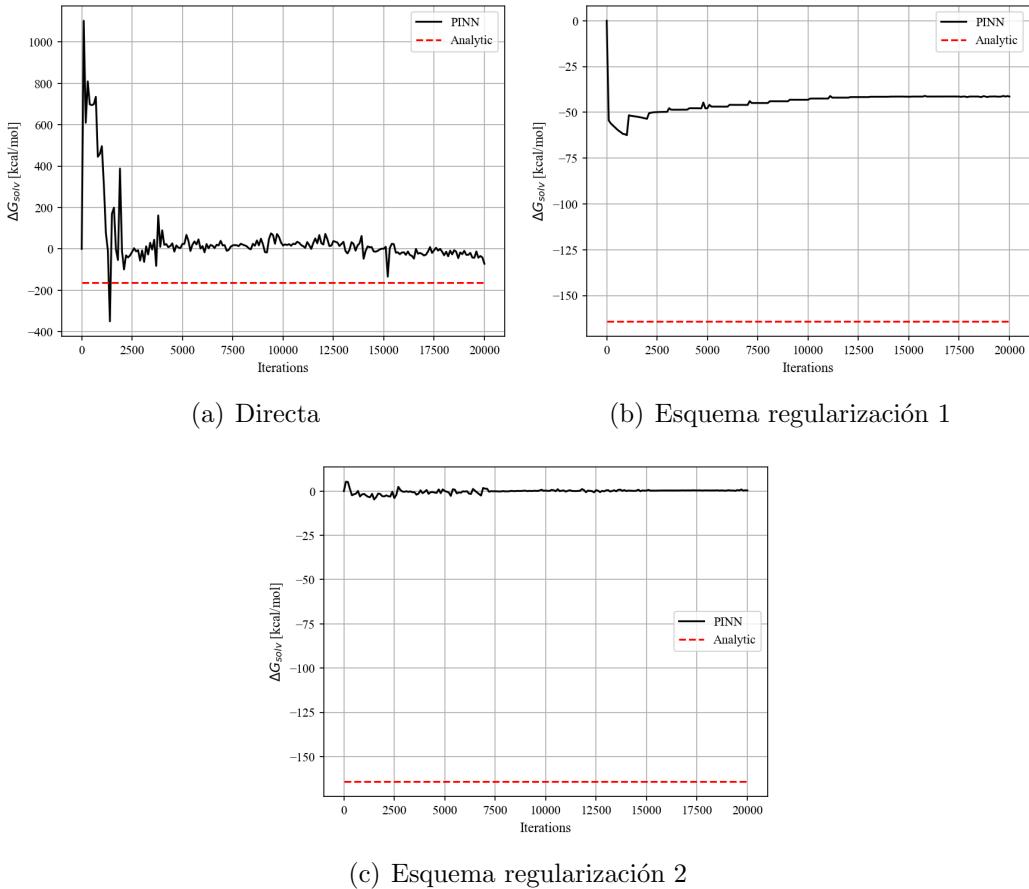


Figura 4.5: Energía de solvatación para los distintos esquemas de la ecuación de PB

Para la energía de solvatación (figura 4.5), notar que ninguno de los casos logra acercarse al valor teórico. Esto evidencia totalmente la incapacidad de utilizar 1 red neuronal para resolver este problema, de la forma en que se implementó el método.

Los resultados mostrados anteriormente demuestran que la descomposición en 2 dominios aplicado al esquema de regularización 2 de la ecuación de PB produce una mejor convergencia a la solución teórica. Por esta razón, todos los resultados a mostrar en las siguientes secciones utilizaron esta configuración.

4.2.2. Estudio de Puntos de Colocación

En esta sección se estudiará el uso de diferentes cantidades de puntos de colocación, y la diferencia entre el método de muestreo aleatorio y el método de muestreo fijo, sección 3.4.

Para todos los casos se resolvió el esquema de regularización 2 de la ecuación de PB. Se utilizaron 2 redes iguales (una para cada dominio) de arquitectura MLP, con 4 capas ocultas, 200 neuronas por capa, con capa de escalamiento del input y capa de Fourier. Además, se utilizó datos puntuales en el solvente dados por la función de Green de Yukawa.

Para ambos métodos, se utilizaron los siguientes puntos de colocación. Se denota “FB” para las simulaciones de muestreo fijo, y “SA” para las simulaciones de muestreo aleatorio.

Tabla 4.4: Cantidad de puntos de colocación para ambos métodos de muestreo²

Simulación	Nodos R1	Nodos R2	Nodos I	Nodos D2	Nodos K2
FB1 y SA1	371	2010	208	320	300
FB2 y SA2	1091	4229	488	320	300
FB3 y SA3	2276	9381	916	1280	300
FB4 y SA4	3233	11146	1216	1280	300

A continuación, se presentan los resultados obtenidos para los 8 casos.

Tabla 4.5: Resultados para el estudio de puntos de colocación

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
FB1	-177.76	8.27E-02	2.33E-01	1.88E+04	2.27E+06
FB2	-154.93	5.64E-02	1.08E-02	5.10E+03	1.80E+07
FB3	-164.63	2.73E-03	5.77E-03	2.00E-03	5.21E-03
FB4	-165.63	8.80E-03	9.49E-03	6.66E-04	3.81E-03
SA1	-167.25	1.87E-02	1.44E-02	9.30E-04	1.65E-03
SA2	-165.78	9.73E-03	4.26E-03	2.72E-04	5.71E-04
SA3	-164.98	4.86E-03	7.66E-03	4.42E-04	1.28E-03
SA4	-164.89	4.32E-03	6.15E-03	4.23E-04	4.74E-04

De los resultados mostrados en la tabla 4.5, el método de muestreo aleatorio tiene consistentemente errores más bajos en todos los indicadores: energía de solvatación, losses de entrenamiento y validación, y error L_2 en la interfaz, en la mayoría de los casos.

Revisando la figura 4.6 se puede visualizar que para la energía de solvatación, los casos de muestreo fijo generan oscilaciones de mayor amplitud a medida que aumentan las iteraciones,

²En estos casos, los Nodos K2 se mantuvieron fijos

4. Resultados y Análisis

si se compara con el método de muestreo aleatorio. Además, los casos de muestreo aleatorio generan oscilaciones de mayor frecuencia. Esto se puede deber a que se están usando puntos de colocación distintos en cada iteración. De todas formas, se nota que los casos de muestreo aleatorio generan una convergencia rápida, incluso para los casos con mallas más gruesas.

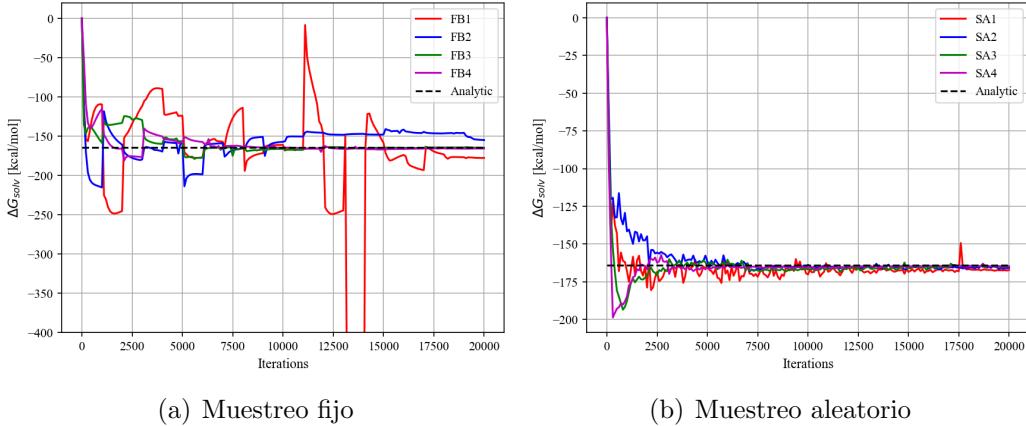


Figura 4.6: Evolución de la energía de solvatación para estudio de puntos de colocación

Respecto al loss de entrenamiento (figura 4.7), se puede revisar que para los métodos de muestreo aleatorio, los 4 casos disminuyen este valor a medida que avanzan las iteraciones. Esto no ocurre para los casos con muestreo fijo. Pareciera ser existe una cantidad de puntos de colocación mínima para que el loss comience a bajar con las iteraciones en este último caso.

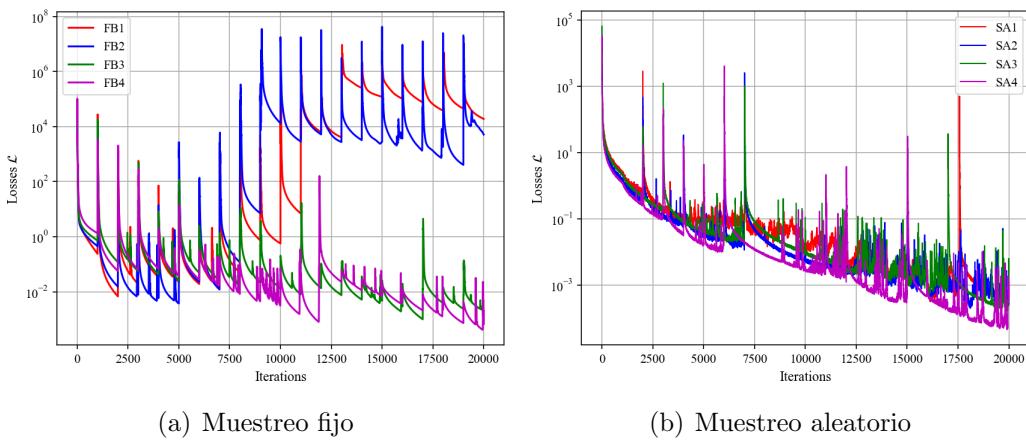


Figura 4.7: Evolución del loss de entrenamiento para estudio de puntos de colocación

Por último, revisando la figura 4.8, se nota como el método de muestreo aleatorio genera una mejor precisión respecto al método de muestreo fijo para la predicción del potencial de reacción. Se nota claramente la superioridad de SA1 vs FB1, y de SA4 vs FB4.

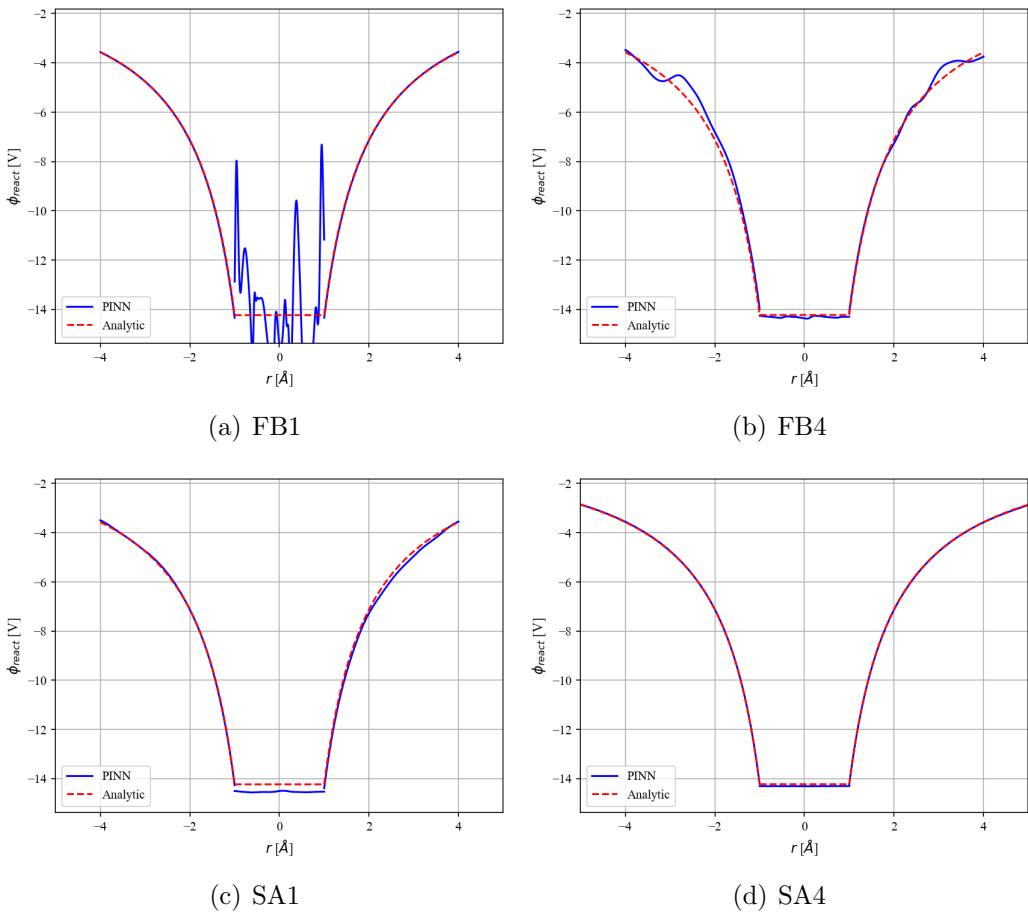


Figura 4.8: Potencial de reacción para estudio de puntos de colocación

El método de muestreo aleatorio genera una mejor generalización de la solución al siempre probar en puntos distintos, en comparación al método de muestreo fijo que mantiene los puntos fijos durante todo el proceso de optimización. El hecho de mantener los puntos fijos puede generar las oscilaciones que se visualizan en la figura 4.8 (quizás producto de *overfitting*), fenómeno que se minimiza al utilizar un muestreo aleatorio.

Comparando los distintas cantidades de puntos de colocación utilizando el método de muestreo aleatorio, se nota que el caso con mayor nodos (SA4) genera menor error en la energía de solvatación y en el loss de validación. Esto puede significar que mayor cantidad de nodos implica una mejor precisión, lo que claramente tiene sentido ya que se está evaluando el loss en más puntos por iteración.

Para las siguientes simulaciones del Ion de Born a mostrar, se utilizaron los mismos puntos de colocación que SA4, utilizando el método de muestreo aleatorio. Además, para el resto de moléculas a estudiar, se intentó escalar estas cantidades de puntos, de modo que se tengan

4. Resultados y Análisis

densidades de puntos parecidas.

4.2.3. Estudio de Arquitectura

En esta sección se estudiará el efecto de la arquitectura en la precisión del método para resolver la ecuación de Poisson-Boltzmann.

En todos los casos, se resolvió el esquema de regularización 2 de la ecuación de PB con 2 redes iguales por dominio. Además, se utilizó datos puntuales en el solvente dados por la función de Green de Yukawa. Por último, se aplicó el método de muestreo aleatorio para los puntos de colocación, utilizando los mismos valores que la simulación SA4 de la sección 4.2.2.

Dentro de este estudio, se mostrarán los resultados para la arquitectura, cantidad de neuronas, cantidad de capas ocultas, y el efecto de las distintas capas o métodos que se pueden agregar a la red. Los casos a estudiar se presentan a continuación:

Tabla 4.6: Casos de prueba para estudio de arquitectura³

Simulación	Arquitectura	C. Fourier	N° Capas	N° NpC	C. Escal.	Func. entr.	WF
C1	MLP	x	4	200	x	x	
C2	MLP		4	200	x	x	
C3	MLP	x	2	200	x	x	
C4	MLP	x	6	200	x	x	
C5	MLP	x	4	100	x	x	
C6	MLP	x	4	250	x	x	
C7	MLP	x	4	200		x	
C8	MLP	x	4	200	x		
C9	MLP	x	4	200			
C10	MLP		4	200			
C11	ResNet	x	4	200	x	x	
C12	ModMLP	x	2	200	x	x	
C13	MLP	x	4	200	x	x	x
C14	ModMLP	x	2	200	x	x	x
C15	ResNet	x	4	200	x	x	x
C16	ModMLP	x	4	200	x	x	
C17	ModMLP	x	4	200	x	x	x
C18	ModMLP	x	4	100	x	x	
C19	ModMLP	x	4	250	x	x	
C20	ModMLP	x	6	200	x	x	
C21	ModMLP		6	200	x	x	
C22	ModMLP		4	200	x	x	

³Nomenclatura: Neuronas por capa (NpC), capa escalamiento del input (C. Escal.), función activación entrenable (Func. entr.), factorización de pesos (WF).

En la tabla 4.7 se pueden revisar los resultados obtenidos. Notar que todos los casos generan una buena predicción de la energía de solvatación, obteniendo errores relativos del orden de 10^{-2} y menores. Los mismos órdenes de magnitud se replican para el error L_2 del potencial de reacción en la interfaz.

Tabla 4.7: Resultados para estudio de arquitectura

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
C1	-164.71	3.24E-03	3.92E-03	4.70E-04	6.22E-04
C2	-164.19	4.23E-05	1.89E-03	4.90E-03	4.72E-03
C3	-165.38	7.31E-03	6.77E-03	2.57E-03	2.59E-03
C4	-165.31	6.85E-03	5.34E-03	9.15E-05	4.03E-04
C5	-165.33	7.00E-03	5.44E-03	4.56E-04	4.12E-04
C6	-164.89	4.33E-03	3.33E-03	2.02E-04	1.05E-03
C7	-166.50	1.41E-02	1.15E-02	1.12E-03	1.32E-03
C8	-164.61	2.58E-03	2.12E-03	3.06E-04	8.91E-04
C9	-166.41	1.36E-02	1.20E-02	1.50E-03	2.11E-03
C10	-165.25	6.50E-03	6.62E-03	1.00E-02	9.85E-03
C11	-166.39	1.34E-02	6.28E-03	7.03E-04	1.15E-03
C12	-166.39	1.35E-02	2.91E-03	2.31E-03	2.47E-03
C13	-165.75	9.57E-03	1.67E-02	1.48E-03	2.44E-03
C14	-165.41	7.47E-03	8.69E-03	3.96E-04	7.36E-04
C15	-162.70	9.01E-03	3.09E-02	6.34E+08	7.00E+07
C16	-163.95	1.44E-03	1.43E-03	3.54E-04	8.61E-04
C17	-164.77	3.58E-03	1.67E-03	8.35E-04	1.05E-03
C18	-167.57	2.07E-02	5.42E-03	8.24E-03	3.60E-02
C19	-164.91	4.41E-03	4.80E-03	6.49E-03	8.42E-03
C20	-166.80	1.59E-02	8.59E-03	3.20E-02	2.16E-02
C21	-164.99	4.93E-03	3.23E-03	2.36E-05	3.53E-05
C22	-165.13	5.79E-03	3.21E-03	3.03E-05	1.98E-05

Revisando el efecto de la arquitectura, en la figura 4.9 se muestra el potencial de reacción para los casos C1 (MLP), C16 (ModMLP) y C11 (ResNet). El uso de la arquitectura ModMLP pareciera ser superior frente a la MLP y la ResNet. Además, logran errores menores en los 3 indicadores a estudiar, alcanzando errores del orden de 10^{-4} tanto para la energía de solvatación, como para el error L_2 en la interfaz.

4. Resultados y Análisis

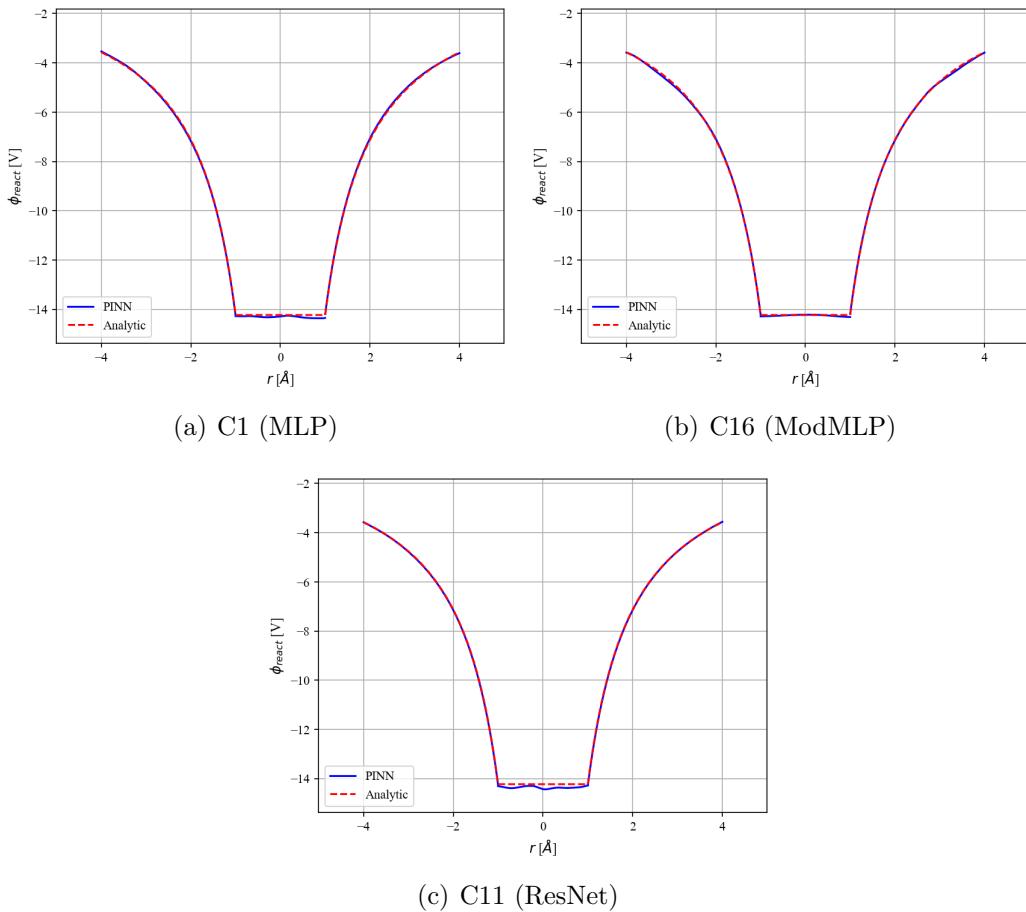


Figura 4.9: Potencial de reacción para las distintas arquitecturas (usando capa de Fourier)

Si se comparan los mismos casos de la figura anterior, pero con la adición de la factorización de los pesos de la red, los 3 resultados empeoran. Lo anterior se puede revisar en la figura 4.10. En este caso, ModMLP sigue siendo superior a MLP y ResNet, manteniendo la forma correcta de la solución, y con los indicadores más bajos. Además, la arquitectura de ResNet genera bastantes oscilaciones a lo largo de todo el dominio.

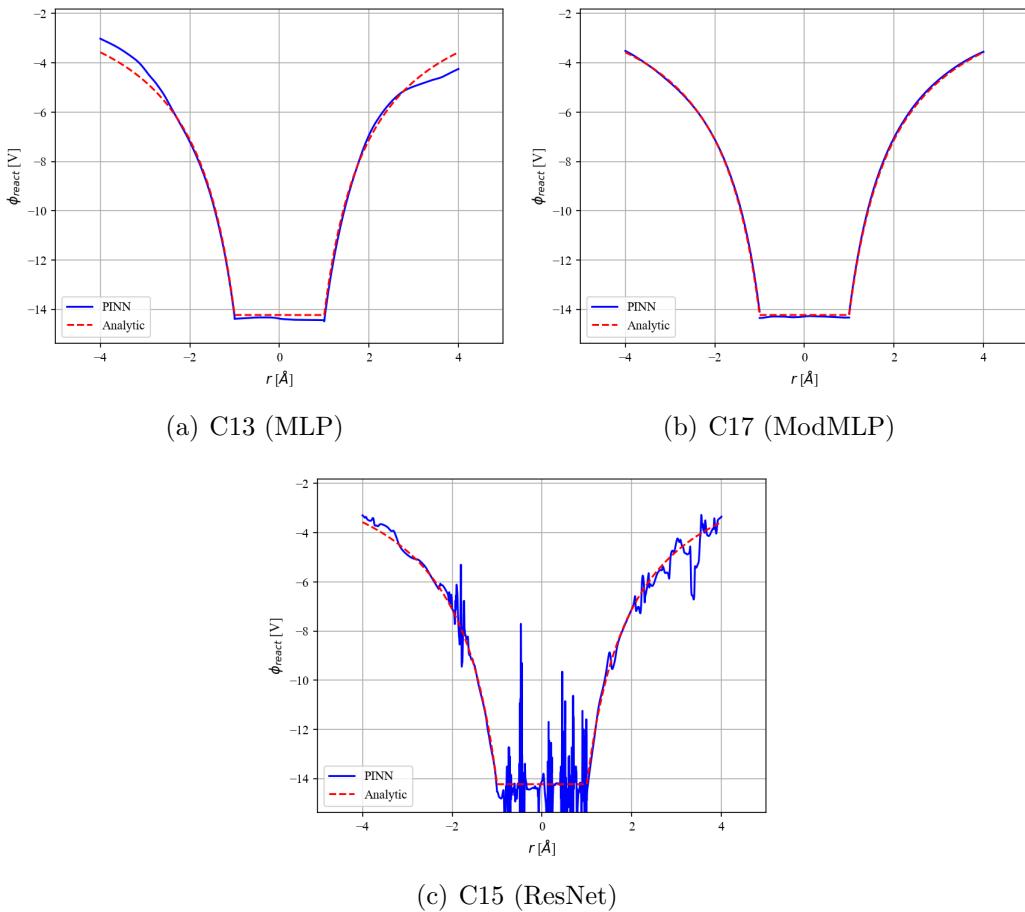


Figura 4.10: Potencial de reacción para las distintas arquitecturas (usando factorización de parámetros y capa Fourier)

Revisando las arquitecturas MLP y ModMLP, se nota claramente que la arquitectura MLP mejora su representación como lo muestra la figura 4.11 al no utilizar la capa de Fourier, incluso llegado a órdenes de 10^{-5} en la predicción de la energía de solvatación. Sin embargo, no se obtiene el mismo comportamiento para la arquitectura ModMLP. El hecho de que MLP mejore sin el uso de la capa de Fourier puede significar que para esta aplicación, la capa de Fourier no es necesaria, quizás por la simpleza del Ion de Born.

4. Resultados y Análisis

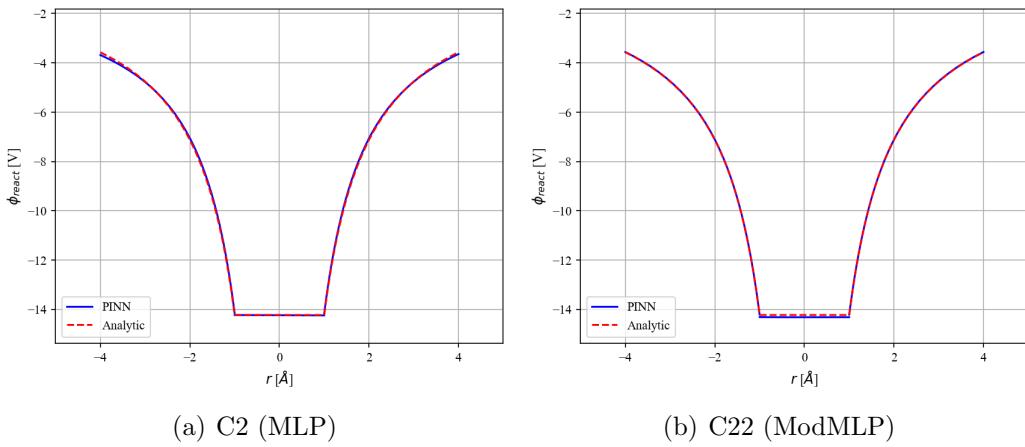


Figura 4.11: Potencial de reacción para las distintas arquitecturas (sin capa de Fourier)

Otros resultados relevantes se muestran en la figura 4.12. Estos resultados demuestran que no utilizar una función de activación adaptativa no tiene un gran efecto en los indicadores (tabla 4.7), pero se nota una leve desviación en el gráfico del potencial. Por otra parte, no utilizar una capa de escalamiento en el input tiene un gran efecto en la predicción de la energía de solvatación. Pareciera ser que la predicción del potencial dentro de la molécula se desplaza, generando errores del orden de 10^{-2} .

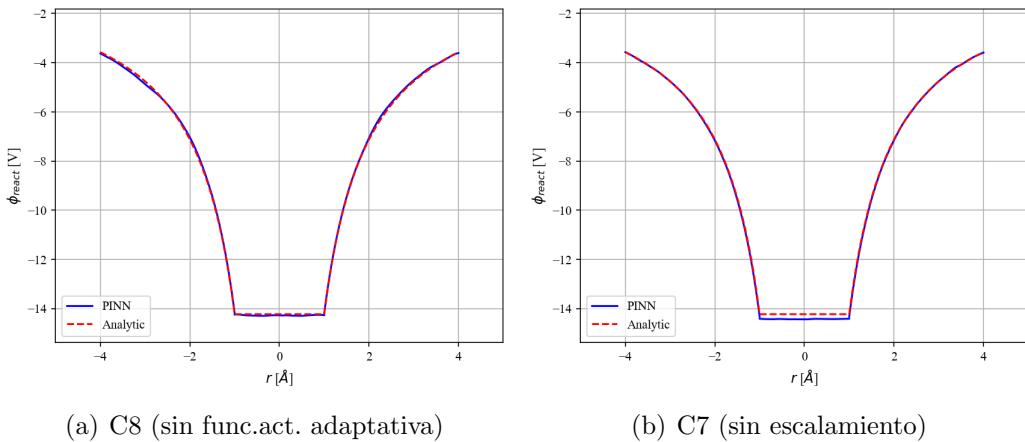


Figura 4.12: Potencial de reacción (usos C. Escal. y Func. entr.)

Por último, se mostrarán los resultados respecto al efecto del número de capas y el número de neuronas por capa en las arquitecturas MLP y ModMLP. En esta línea, en la figura 4.13 se puede revisar la predicción de la energía de solvatación y el loss de entrenamiento a medida que aumentan las iteraciones.

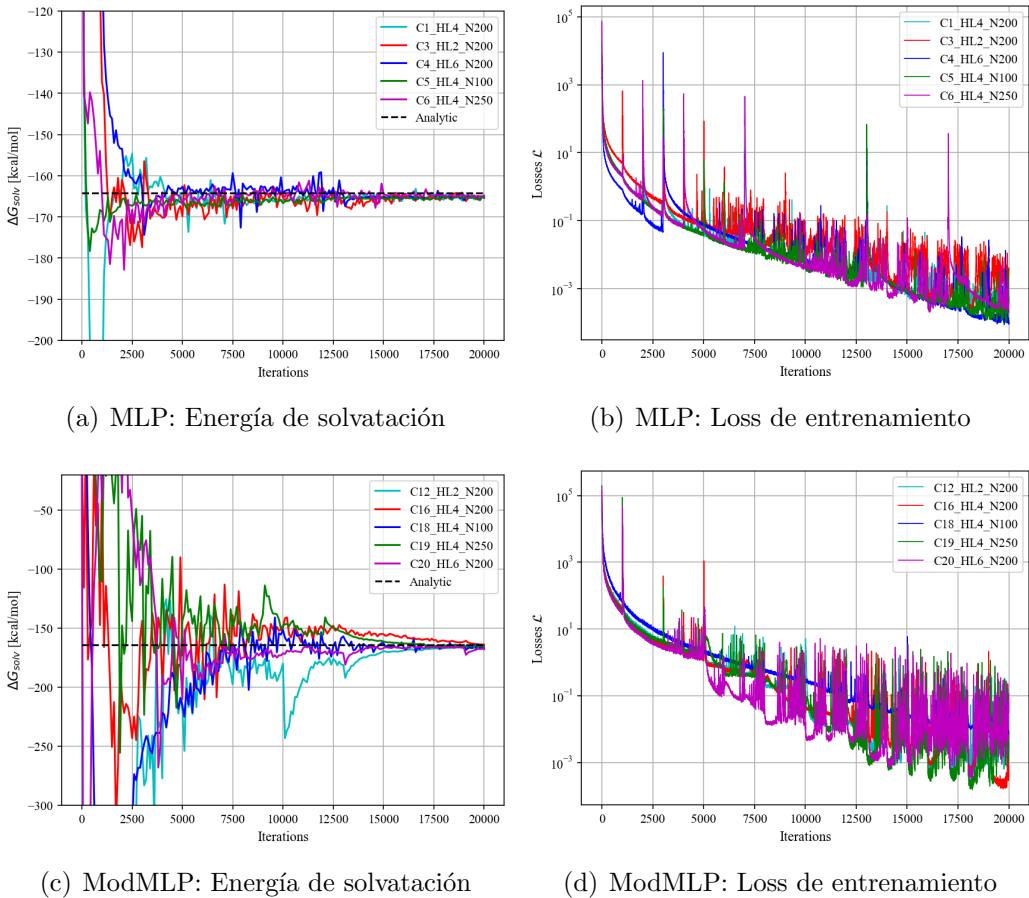


Figura 4.13: Energía de solvatación y loss de entrenamiento para revisión de número de capas y neuronas

Para el caso de la arquitectura MLP, las predicciones de la energía de solvatación son bastante similares, incluso manteniendo un error del orden de 10^{-3} en todos los casos. De todas formas, se visualizan mayores oscilaciones en los casos que se tienen mayor número de capas ocultas. Por otra parte, la evolución de la función de pérdida en estos casos sigue el mismo comportamiento.

Para el caso de la arquitectura ModMLP, tanto la predicción de la energía de solvatación, como el cálculo del loss de entrenamiento, generan oscilaciones de mayor amplitud al compararlo con la arquitectura MLP. Además, en algunos casos se llegan a órdenes de 10^{-2} en el error de la energía de solvatación. Pese a que los resultados mostrados en la figura 4.9 muestran a la arquitectura ModMLP como mejor opción, la figura 4.13 muestra que el entrenamiento es más ruidoso.

Para finalizar, el Ion de Born al ser una molécula bastante simple, puede entregar un sesgo incorrecto de los resultados presentados. Es por esto que se mantuvieron 6 arquitecturas para las pruebas del multipolo esférico. En estas 6 arquitecturas se fijó el número de capas ocultas

4. Resultados y Análisis

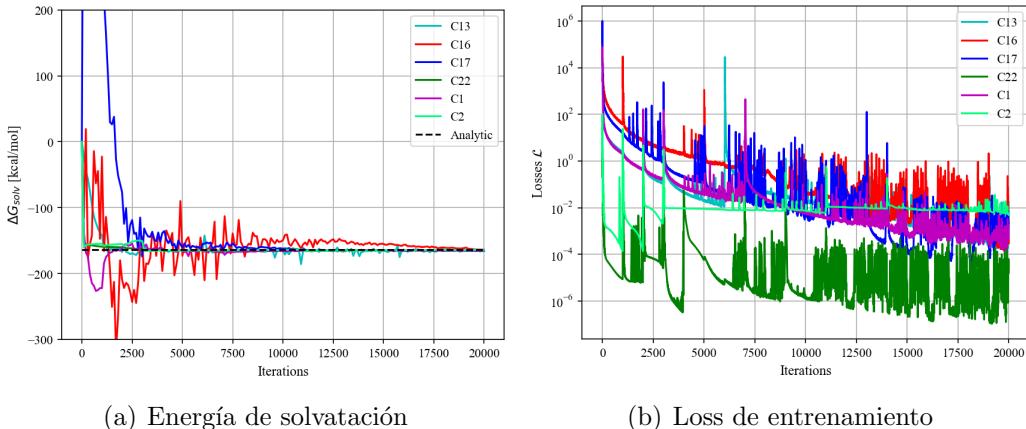
a 4 y el número de neuronas por capa a 200. Además, se utilizó la capa de escalamiento y funciones de activación entrenables.

Para estas 6 arquitecturas se utilizan las redes MLP o ModMLP, y con la variación del uso de la capa de Fourier y la factorización de los pesos de la red. En la tabla 4.8 se detallan estas 6 arquitecturas.

Tabla 4.8: Configuraciones a mantener para multipolo esférico

Simulación	Arquitectura	C. Fourier	WF
C1	MLP	x	
C2	MLP		
C13	MLP	x	x
C16	ModMLP	x	
C22	ModMLP		
C17	ModMLP	x	x

Por último, se muestra la evolución de la energía de solvatación y el loss de entrenamiento para estas 6 configuraciones a mantener.



(a) Energía de solvatación

(b) Loss de entrenamiento

Figura 4.14: Energía de solvatación y loss de entrenamiento para las configuraciones a mantener para el multipolo esférico

Estos mismos gráficos serán adjuntados para el multipolo esférico, de tal manera que se visualice si existe una extrapolación de los resultados a los casos con mayor cantidad de cargas.

4.2.4. Estudio de Funciones de Pérdida

En esta sección se revisará el efecto de los distintos términos que se pueden añadir a la función de pérdida. Como base, todos los casos a estudiar tendrán como términos: Residuales en ambos dominios (R_1 y R_2), continuidad en el potencial y el desplazamiento eléctrico en la interfaz (I_u e I_d) y la condición de borde en el solvente (D_2).

En todos los casos, se resolvió el esquema de regularización 2 de la ecuación de PB con 2 redes iguales por dominio de arquitectura MLP, con 4 capas ocultas, 200 neuronas por capa, con capa de escalamiento del input y capa de Fourier. Se utilizó el método de muestreo aleatorio para los puntos de colocación de R_1 , R_2 , I y D_2 .

Como notación, se utilizará K_2 para los datos puntuales en el solvente usando la función de Green de Yukawa, E_2 para el uso de datos experimentales⁴ de ϕ_{ENS} , y G el uso de la Ley de Gauss.

Tabla 4.9: Puntos de colocación para estudio de función de pérdida

Nodos R1	Nodos R2	Nodos R2 caso E2 ⁵	Nodos I	Nodos D2	Nodos K2	Nodos E2
2355	9066	10012	916	1280	152	423

En la tabla 4.10 se presentan los casos de prueba.

Tabla 4.10: Casos de prueba para estudio de función de pérdida

Simulación	K2	E2	G	Algor. Pesos
L1	x			x
L2				x
L3		x	x	x
L4	x	x		x
L5	x		x	x
L6	x	x	x	x
L7	x			
L8			x	x

Todos los resultados obtenidos fueron presentados en la tabla 4.11.

⁴Los “datos experimentales” de ϕ_{ENS} para el Ion de Born fueron obtenidos usando la solución analítica y la ecuación 3.46

⁵Para el caso de utilizar el término de datos experimentales, el dominio tendrá como contorno una esfera de radio $R = 5.5$ [Å], en comparación a los otros casos que tienen una esfera de radio $R = 4$ [Å]. Esto es debido a que ϕ_{ENS} se calcula utilizando predicciones a una distancia de 3.5 [Å] desde la interfaz de la molécula.

4. Resultados y Análisis

Tabla 4.11: Resultados para estudio de función de pérdida

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
L1	-165.36	7.19E-03	3.40E-03	1.35E-03	1.49E-03
L2	-165.24	6.44E-03	3.34E-03	2.89E-04	7.87E-04
L3	-482.98	1.94E+00	6.12E-02	2.99E+02	1.53E+00
L4	-165.02	5.10E-03	3.23E-03	6.25E-03	9.51E-07
L5	-165.46	7.80E-03	6.05E-03	6.25E+00	2.74E-03
L6	-404.07	1.46E+00	3.77E-02	4.13E+02	7.59E-01
L7	-165.11	5.65E-03	5.24E-03	2.19E-03	1.02E-02
L8	-166.18	1.21E-02	1.03E-02	5.37E+00	6.73E-03

Además, se grafica la evolución de la energía de solvatación y el loss de entrenamiento en todos los casos.

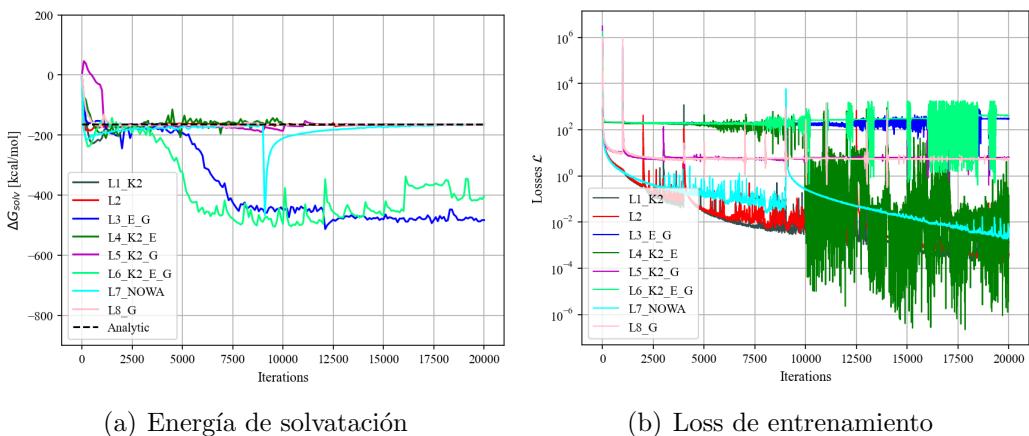


Figura 4.15: Energía de solvatación y loss de entrenamiento para estudio de función de pérdida

De lo obtenido, se nota que hay casos que no logran predecir una buena energía de solvatación, como lo son L3 y L6 (combinación de losses que contienen integrales). Además, hay varios casos en que el loss de entrenamiento no disminuye con las iteraciones. Estos casos serán estudiados con más profundidad.

En la figura 4.16 se presentan los gráficos del potencial de reacción para 3 casos, considerando el uso o no de K2 y el uso del algoritmo de ponderación de términos.

Respecto a esto, no se nota un gran impacto con el uso de datos puntuales (K2). Además, el error en la energía de solvatación y el potencial en la interfaz dan bastante similares, con órdenes de 10^{-3} . Sin embargo, el caso sin datos puntuales reduce en 1 orden de magnitud los losses de entrenamiento y validación.

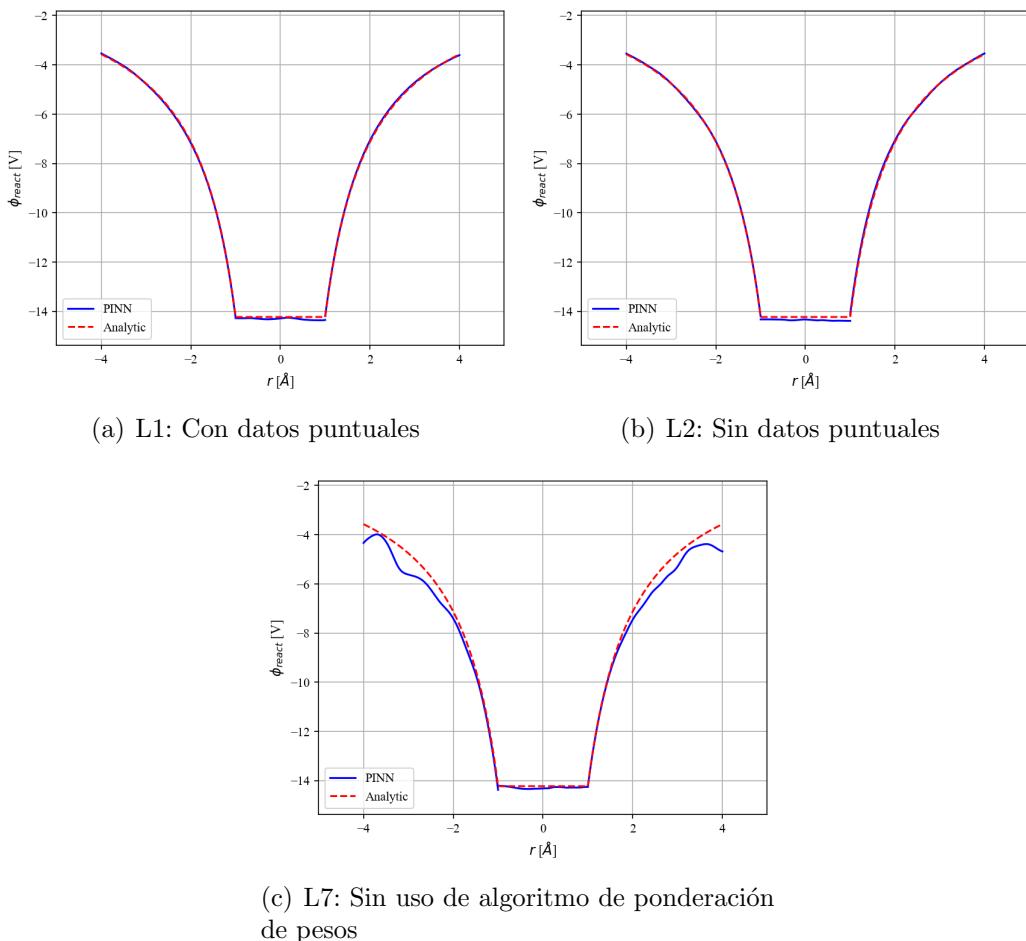


Figura 4.16: Potencial de reacción para distintos casos, estudio de función de pérdida

Por otra parte, notar que el hecho de no usar el algoritmo de ponderación de los términos de la función de pérdida empeora automáticamente la solución, generando oscilaciones no deseadas. Además, aumenta en 1 orden de magnitud el loss de validación, respecto a los 2 casos anteriores. Esto se puede deber principalmente a que no se están minimizando todos los términos en conjunto debido a que algunos pueden predominar respecto a otros.

En la figura 4.17 se puede revisar la evolución de los distintos términos de la función de pérdida. Como tendencia, se nota que en el dominio 1 (sólido), el término del residual (ecuación de Laplace) es el predominante. Este término es seguido por la continuidad en el desplazamiento eléctrico \mathbf{Id} . Era esperable que \mathbf{Id} sea de los términos altos debido a que se está forzando la continuidad en el desplazamiento eléctrico, la cual es la derivada de la red neuronal respecto a los parámetros de entrada.

En el dominio 2 (solvente), no existe un claro término predominante. El mayor loss está compartido por el residual, la continuidad en el desplazamiento eléctrico.

4. Resultados y Análisis

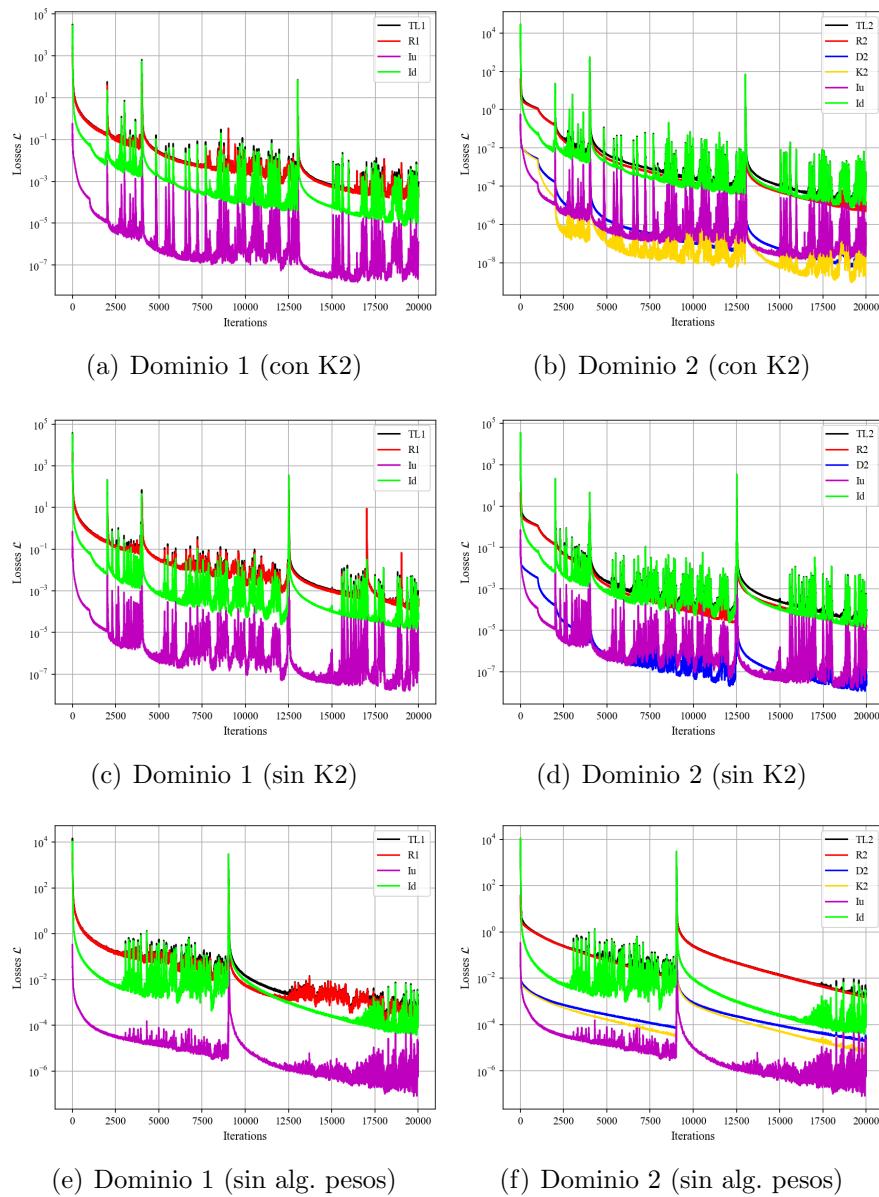


Figura 4.17: Términos de la función de pérdida para los casos probados

Respecto al algoritmo de ponderación de los términos de la función de pérdida, se puede visualizar en la figura 4.17 que añadir este algoritmo genera más oscilaciones en la evolución del loss, en comparación al no usarlo. Esto se puede deber a que cada un cierto número de iteraciones, la función objetivo cambia. Sin embargo, los indicadores para la energía de solvatación y el potencial mejoran al utilizar este método.

Los resultados para el uso de la Ley de Gauss y los datos experimentales se pueden revisar en la figura 4.18. Notar que los casos L4 y L5 generan una buena representación del potencial de reacción, con ambos errores del orden de 10^{-3} . Lo curioso ocurre al agregar estos 2 térmi-

nos a la vez (caso L6). Notar como el modelo pierde totalmente la capacidad de representar correctamente el potencial en el soluto.

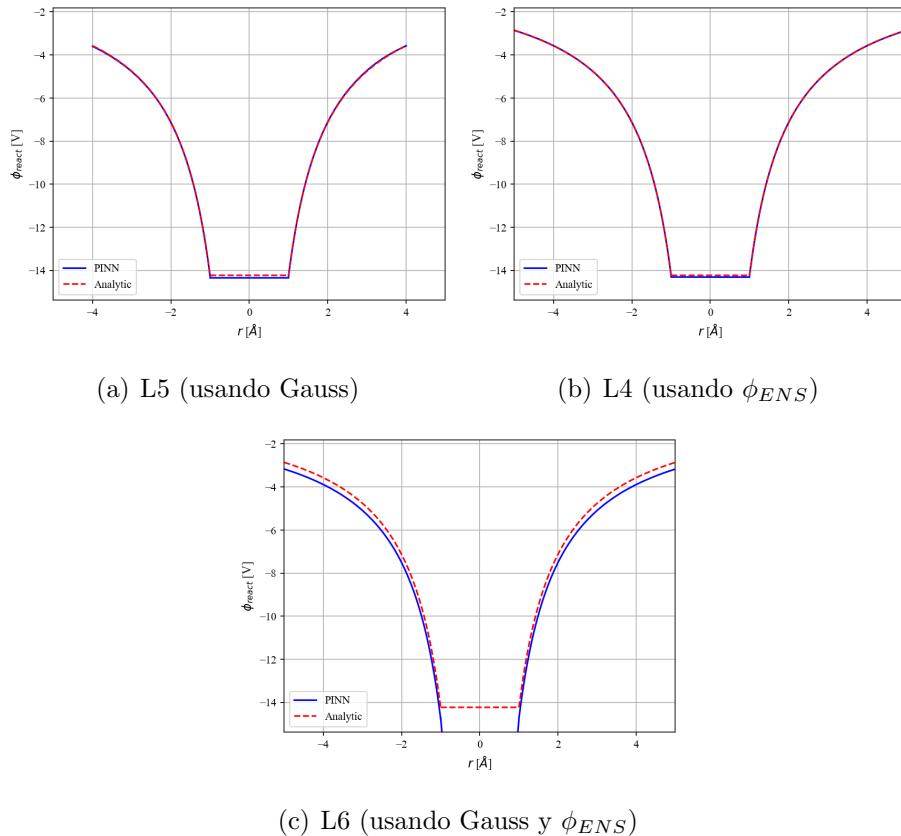


Figura 4.18: Potencial de reacción para casos con distintos términos en la función de pérdida

En la figura 4.19 se puede revisar la evolución de los distintos términos de la función de pérdida. Respecto a esto, se nota como el término de la Ley de Gauss no se minimiza, estabilizándose cerca en el orden de 10^0 . Además, pareciera ser que esta estabilización afecta también al término de Id. Esto es debido a que el término de Gauss utiliza solamente la predicción del gradiente del potencial en la interfaz. Por otra parte, el loss para los datos experimentales pareciera tener un comportamiento similar, pero con oscilaciones de gran amplitud. Por último, al juntar ambos términos (G y E2) se visualiza como los componentes de la función de pérdida tienen un comportamiento no deseado.

4. Resultados y Análisis

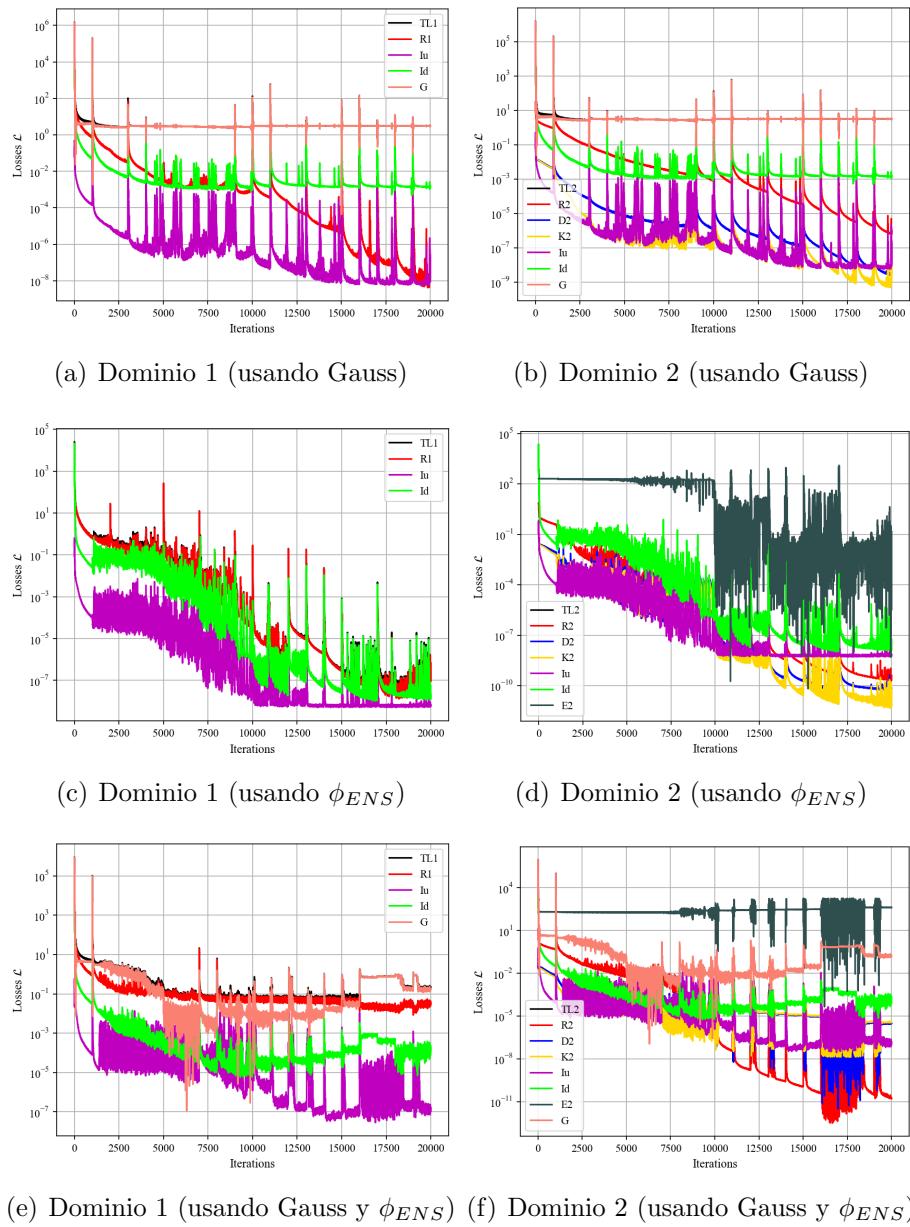


Figura 4.19: Términos de la función de pérdida para los casos probados

4.2.5. Estudio del Método de Optimización

En esta sección se revisará el efecto de la combinación de 2 métodos de optimización distintos, ADAM y L-BFGS. En todas las simulaciones mostradas anteriormente se utilizó el método de ADAM para minimizar la función de pérdida, en donde se obtuvo buenos resultados. Sin embargo, se pretende revisar si estos resultados pueden ser mejorados al incluir etapas de minimización adicionales con el método de L-BFGS, el cual es de segundo orden.

Para todos los casos a mostrar, se resolvió el esquema de regularización 2 de la ecuación de PB con 2 redes iguales por dominio, de arquitectura MLP, con 4 capas ocultas, 200 neuronas por capa, con capa de escalamiento del input y función de activación entrenable.

Respecto a la elección de los puntos de colocación, se utilizó un muestreo aleatorio en las etapas en que se aplica el algoritmo de ADAM. Para las etapas con L-BFGS se tuvo que utilizar un muestreo fijo, ya que este método necesita utilizar los mismos puntos de colocación en cada iteración. Para inducir aleatoriedad en la etapa de L-BFGS, se evaluará implementar el reinicio del algoritmo cada cierto número de iteraciones, de forma que se puedan utilizar nuevos puntos de colocación. Por último, para el método de L-BFGS se utilizó una tolerancia de 10^{-5} , y un historial de 50 iteraciones para la aproximación de la matriz Hessiana.

En esta línea, las pruebas que se mostrarán son las siguientes:

Tabla 4.12: Casos de prueba para estudio del método de optimización

Simulación	Iter ADAM	Iter máx L-BFGS	Reinicio L-BFGS
OP1	20000	0	No
OP2	10000	10000	No
OP3	10000	10000	c/2000
OP4	0	20000	c/10000

Los resultados para los 4 casos pueden ser revisados en la tabla 4.13. De la tabla se nota que los casos que utilizan el algoritmo de L-BFGS posterior a una primera etapa con ADAM (OP2 y OP3), generan errores más bajos en la energía de solvatación y en el potencial, en comparación con el caso que utiliza solo ADAM. Además, los losses disminuyen en 2 o más órdenes de magnitud. El comportamiento de los losses se puede visualizar en la figura 4.20.

4. Resultados y Análisis

Tabla 4.13: Resultados para estudio del método de optimización

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación	Iters
OP1	-165.55	8.32E-03	6.48E-03	4.15E-03	4.20E-03	20000
OP2	-165.52	8.17E-03	5.79E-03	1.98E-05	1.63E-05	14662
OP3	-165.19	6.12E-03	3.69E-03	4.08E-06	3.83E-06	20000
OP4	-152.77	6.95E-02	7.21E-02	1.15E-05	1.60E-05	130

Notar como los losses de entrenamiento y validación disminuyen cuando se inicia la etapa de minimización con L-BFGS. Además, notar como OP3 (caso con reinicio de L-BFGS) logra llegar a losses y errores menores debido a que se permite continuar con el proceso de minimización, además de incluir aleatoriedad. Adicionalmente, se nota que el proceso de minimización con ADAM es bastante ruidoso, en comparación al método de L-BFGS.

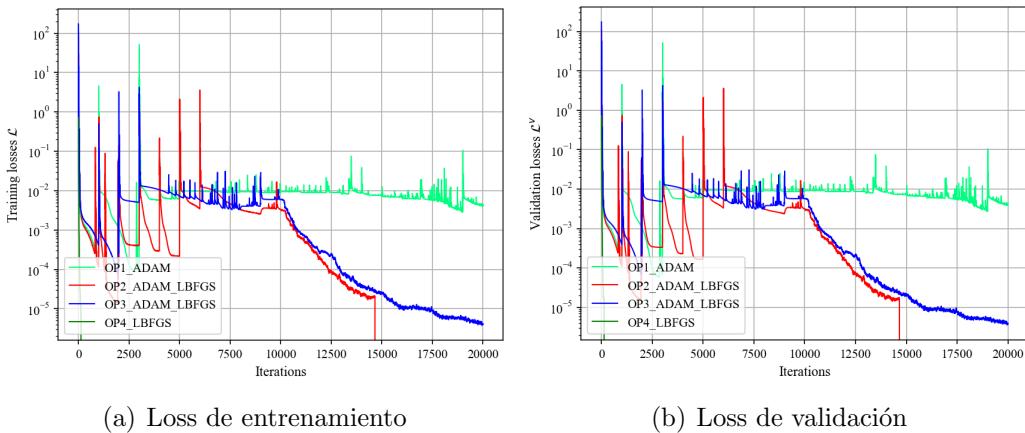


Figura 4.20: Loss de entrenamiento y de validación para estudio del método de optimización

Un caso importante a destacar es el uso del método de L-BFGS sin un paso de ADAM previo. Pese a que este método alcanza la tolerancia fijada en 130 iteraciones, genera errores en la energía de solvatación y en el potencial de un orden mayor que el resto de los casos. Revisando la figura 4.21, se visualiza exactamente la causa de estos mayores errores. El método de L-BFGS tiende a caer en mínimos locales cuando no se tiene un buen punto de partida. Es por esto que se recomienda la adición de este método posterior a una primera etapa con ADAM.

Por último, en la figura 4.21 se puede visualizar que los 3 casos que utilizan ADAM logran representar la solución analítica. Sin embargo, el caso OP3 (ADAM + L-BFGS con reinicio) minimiza cualquier desviación respecto a la solución de referencia, reflejando la importancia de utilizar ambos métodos.

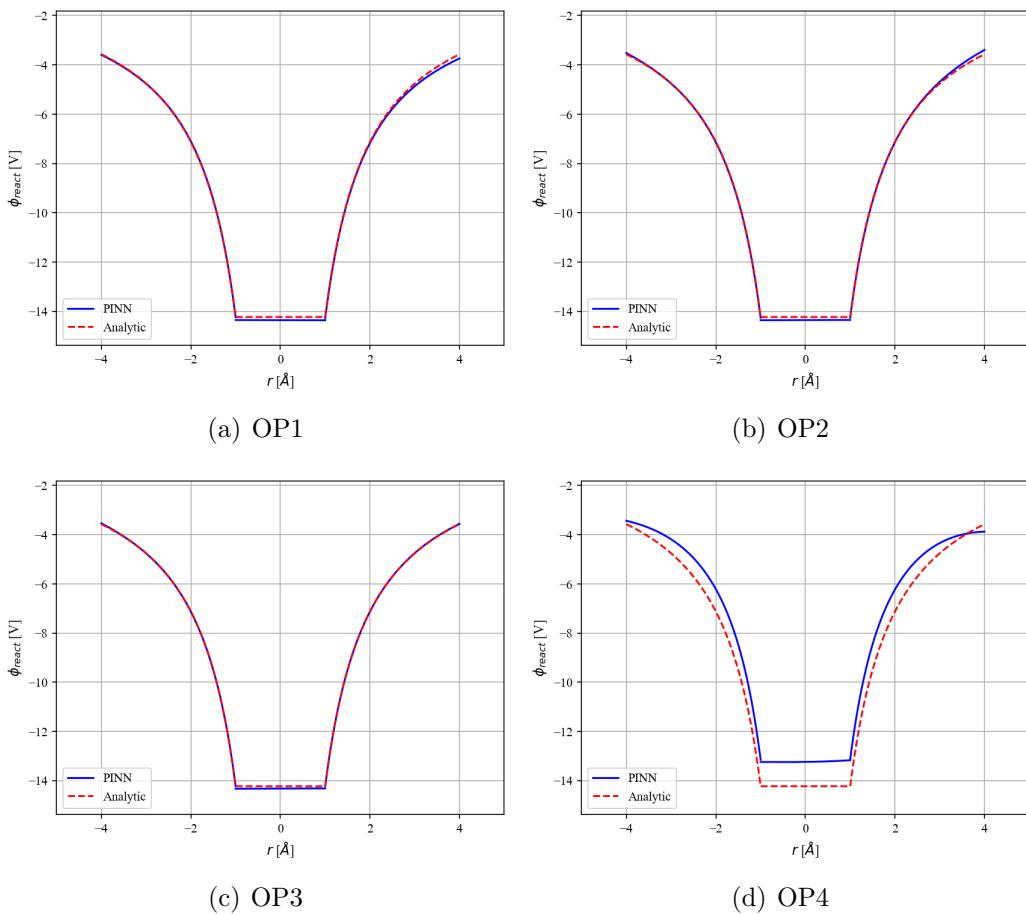


Figura 4.21: Potencial de reacción para estudio del método de optimización

4.3. Multipolo Esférico

El multipolo esférico corresponde a una molécula esférica con una colección de cargas puntuales dentro. Esta molécula servirá como una segunda validación de resultados ya que se conoce la solución analítica, la cual fue presentada en la sección 2.1.8.2.

Se utilizaron las siguientes dimensiones y/o propiedades en las simulaciones del multipolo:

- Constante dieléctrica soluto: $\epsilon_1 = 1 [\epsilon_0]$
- Constante dieléctrica solvente: $\epsilon_2 = 80 [\epsilon_0]$
- Inverso de la longitud de Debye: $\kappa = 0.125 [1/\text{\AA}]$
- Radio molécula: $R = 1.2 [\text{\AA}]$
- Radio esfera frontera solvente: $R = 4.2 [\text{\AA}]$

Para el multipolo esférico, se probaron 4 configuraciones de cargas. Las posiciones de estas cargas se pueden revisar en los archivos `.pqr` adjuntados en la sección A.1.

- 1 carga descentrada.
- 2 cargas positivas.
- 2 cargas de signos opuestos.
- 5 cargas de distinta magnitud y signo.

Para estas 4 configuraciones de cargas, se realizaron 6 simulaciones. En todos los casos se resolvió el esquema de regularización 2 de la ecuación de PB lineal, y como arquitectura se utilizaron las 6 configuraciones a mantener, obtenidas en los estudios del Ion de Born, dados en la tabla 4.8.

Tabla 4.14: Casos de prueba para el multipolo esférico

Simulación	Arquitectura	C. Fourier	WF
S1	MLP	x	
S2	MLP		
S3	MLP	x	x
S4	ModMLP	x	
S5	ModMLP		
S6	ModMLP	x	x

Recordar que estas 6 arquitecturas consideran 4 capas ocultas con 200 neuronas por capa. Además, incluyen la capa de escalamiento del input y funciones de activación entrenables. Para todos los casos se utilizaron los mismos puntos de colocación, incluyendo datos conocidos en el solvente dados por la función de Green de Yukawa.

Tabla 4.15: Puntos de colocación para el multipolo esférico

Nodos R1	Nodos R2	Nodos I	Nodos D2	Nodos K2
3690	11332	1352	1280	300

4.3.1. Carga Descentralizada

El primer caso de estudio del multipolo esférico es la carga descentralizada. Se utiliza este caso dado que se pierde la simetría existente en el Ion de Born. Esto puede entregar indicios si los resultados del Ion de Born son extrapolables para otras situaciones.

En la tabla 4.16 se pueden revisar los resultados para las 6 arquitecturas.

Tabla 4.16: Resultados para la carga descentralizada

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
S1	-158.65	2.61E-03	3.37E-03	2.74E-04	7.26E-04
S2	-157.98	6.82E-03	3.35E-02	6.55E-03	6.37E-03
S3	-172.99	8.75E-02	3.38E-02	5.53E-03	2.07E-02
S4	-159.34	1.73E-03	1.02E-02	5.98E-02	5.78E-02
S5	-159.50	2.70E-03	4.81E-03	3.18E-05	4.59E-05
S6	-160.21	7.18E-03	3.15E-03	4.54E-04	5.62E-04

Respecto a los resultados, se nota como todos los casos, excepto S3, tienen errores en la energía de solvatación del orden de 10^{-3} . Además, para el error en el potencial, solo S1, S5 y S6 logran errores del orden de 10^{-3} .

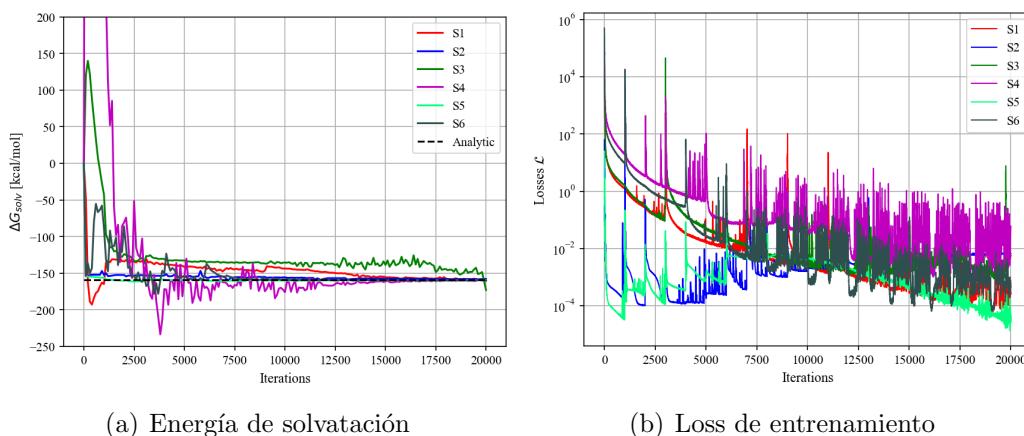


Figura 4.22: Energía de solvatación y loss de entrenamiento para carga descentralizada

Revisando la evolución de la energía de solvatación, se puede notar como todos los casos logran converger al valor teórico, excepto S3. Por parte del loss de entrenamiento, se nota que todos

4. Resultados y Análisis

descienden a medida que aumentan las iteraciones, pero el comportamiento de S2 y S5 (casos sin capa de Fourier) no fue monótono.

En las figuras 4.23 y 4.24 se puede revisar la predicción del potencial de reacción en 2 direcciones distintas. En ambas figuras se nota que S2 (sin uso de la capa de Fourier) no logra replicar la curvatura del potencial de reacción dentro de la molécula, generando líneas rectas. Recordar que para el Ion de Born la solución mejoraba sin el uso de la capa de Fourier. Sin embargo, ese resultado no fue extrapolable para este caso donde se pierde la simetría esférica.

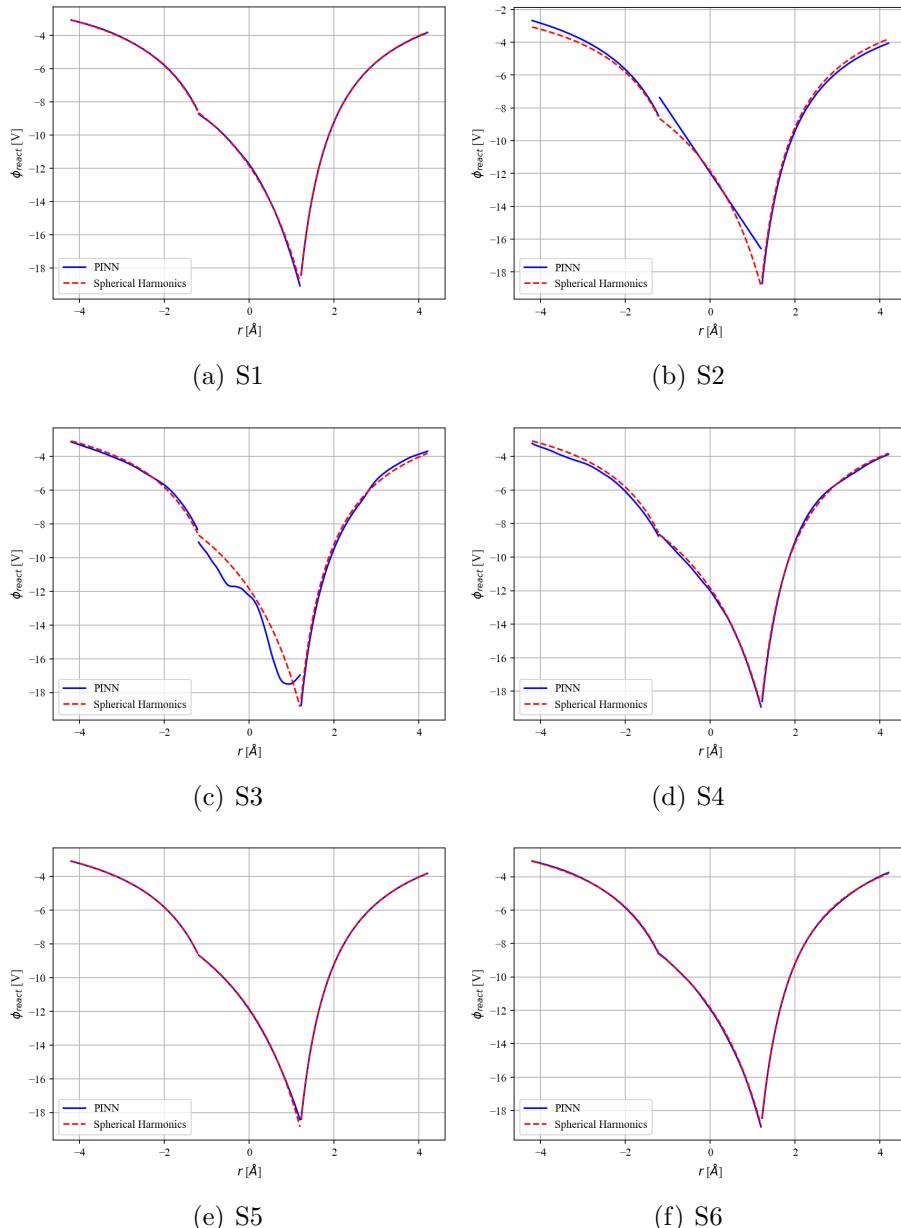


Figura 4.23: Potencial de reacción para carga descentrada, dirección $\theta = 0$, $\phi = \pi/2$

Otro caso que no logra una buena predicción del potencial es S3 (Fourier más factorización de pesos de la red). Se infiere que la arquitectura MLP y la factorización de pesos de la red no conviven correctamente. Hecho opuesto ocurre con la arquitectura ModMLP, para la cual S6 (Fourier más factorización de pesos de la red) pareciera generar una de las mejores representaciones. Además, es importante destacar que se puede evitar la capa de Fourier si se utiliza la arquitectura ModMLP, tal como lo muestran los resultados de S5.

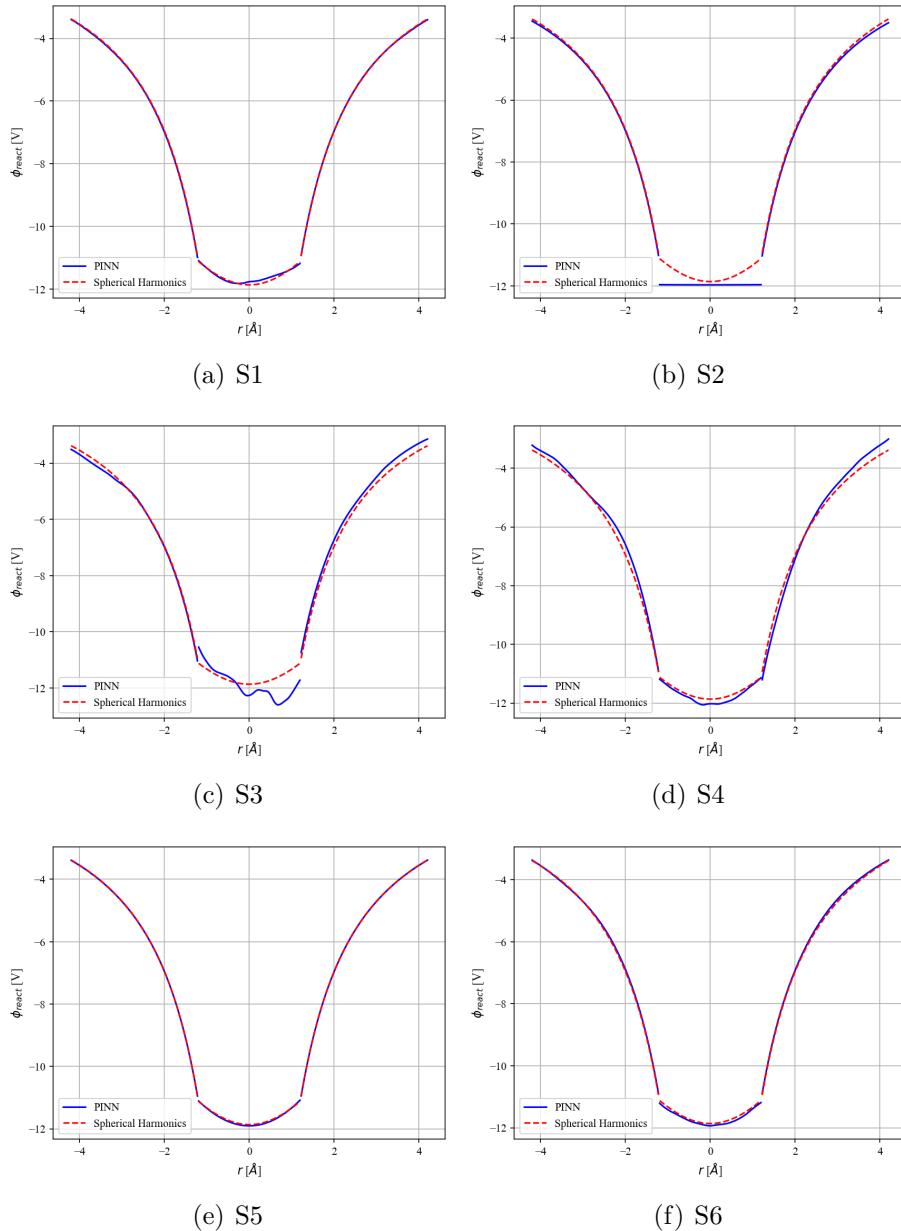


Figura 4.24: Potencial de reacción para carga descentrada, dirección $\theta = \pi/2$, $\phi = \pi$

4.3.2. 2 Cargas Positivas

El segundo caso del multipolo esférico es la configuración de 2 cargas positivas descentradas. Este escenario es una continuación del caso de la carga descentrada. Los resultados obtenidos se presentan en la tabla 4.17.

Tabla 4.17: Resultados para 2 cargas positivas

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
S1	-558.76	9.75E-03	1.06E-02	2.15E-04	1.71E-03
S2	-550.03	2.52E-02	6.17E-02	1.78E-02	1.75E-02
S3	-537.68	4.71E-02	1.64E-02	5.89E-04	3.20E-03
S4	-567.18	5.17E-03	6.17E-03	2.27E-03	1.09E-03
S5	-565.20	1.67E-03	3.82E-02	5.97E-05	6.29E-05
S6	-567.35	5.47E-03	1.48E-02	5.01E-04	4.99E-04

En los resultados obtenidos se puede revisar que se tienen errores de similar magnitud en comparación con el caso de la carga descentrada. Lo interesante es que S4 es la única que logra un error del potencial en la interfaz del orden de 10^{-3} (S1 y S6 están bastante cerca).

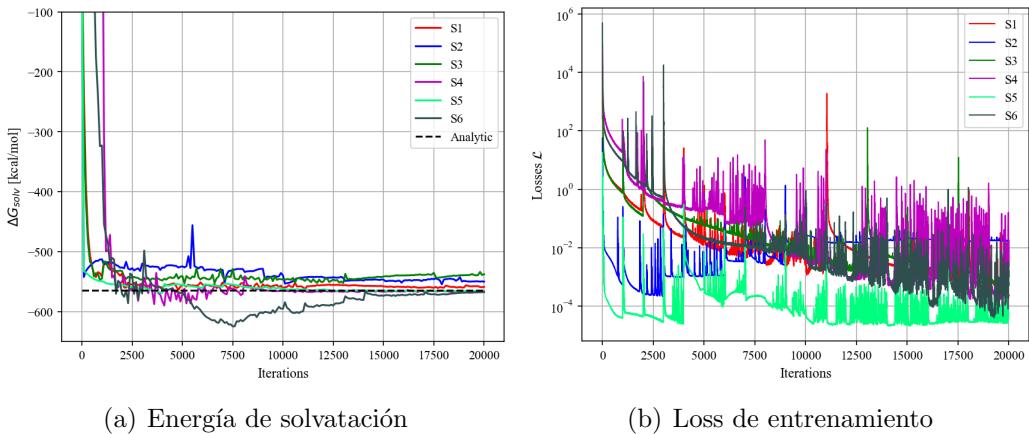


Figura 4.25: Energía de solvatación y loss de entrenamiento para 2 cargas positivas

Respecto a la evolución de la energía de solvatación, se nota que se tiene una mayor desviación respecto a la solución analítica. Esto predomina en S2, S3 y S6. Además, las configuraciones con una convergencia más rápida parecieran ser S4 y S5. Por último, para el loss de entrenamiento se puede revisar que la mayoría de los casos tienen una tendencia monótona decreciente, excepto S2 que se mantiene constante.

Para las gráficas del potencial de reacción, figuras 4.26 y 4.27, se obtiene el mismo fenómeno observado para la carga descentrada. S2 y S3 son incapaces de representar correctamente el potencial en el interior de la molécula. Por otra parte, se visualizan desviaciones en el centro de la molécula para S1, S4 y S6. Estas desviaciones no son visualizadas en S5, pero este caso tiene ciertas desviaciones en la interfaz.

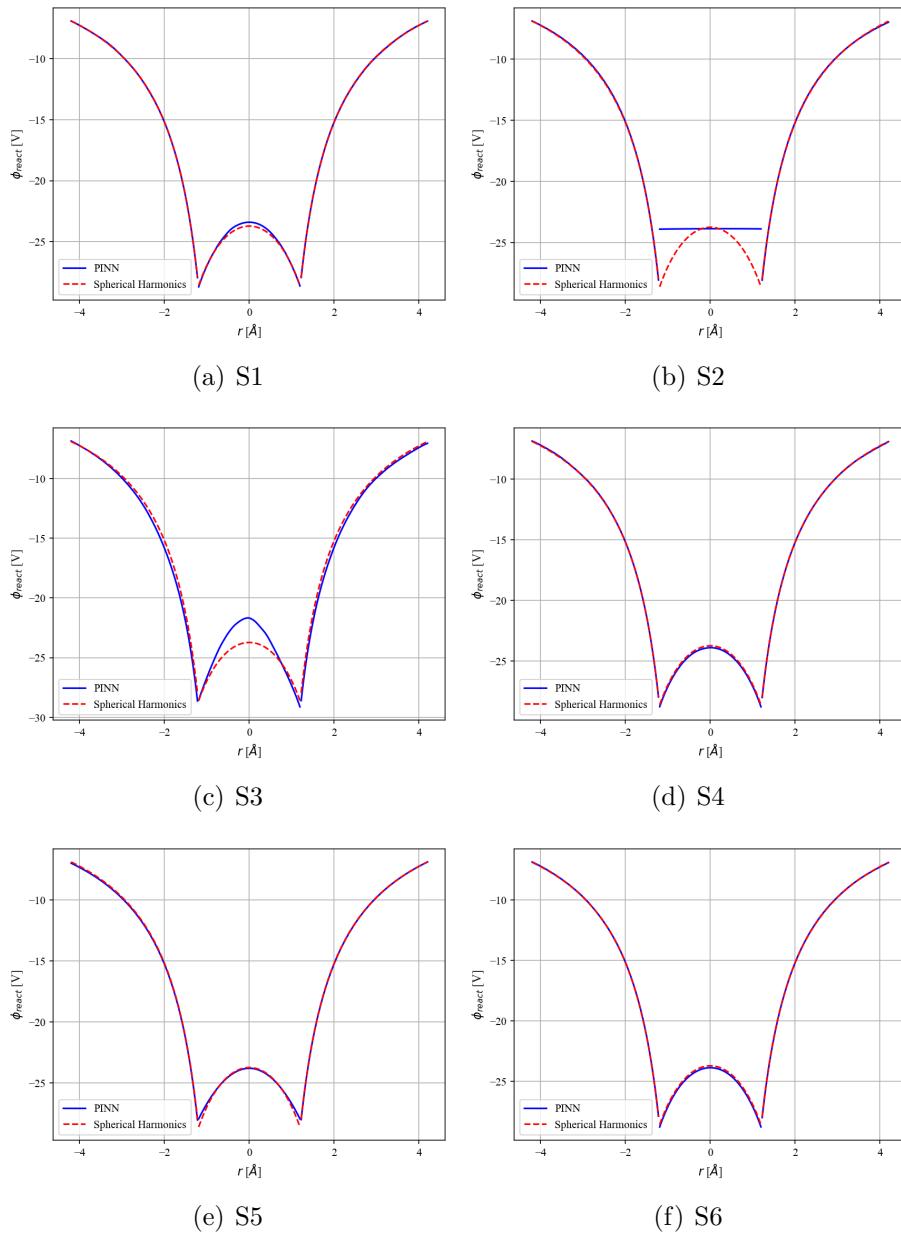


Figura 4.26: Potencial de reacción para 2 cargas positivas, dirección $\theta = 0$, $\phi = \pi/2$

4. Resultados y Análisis

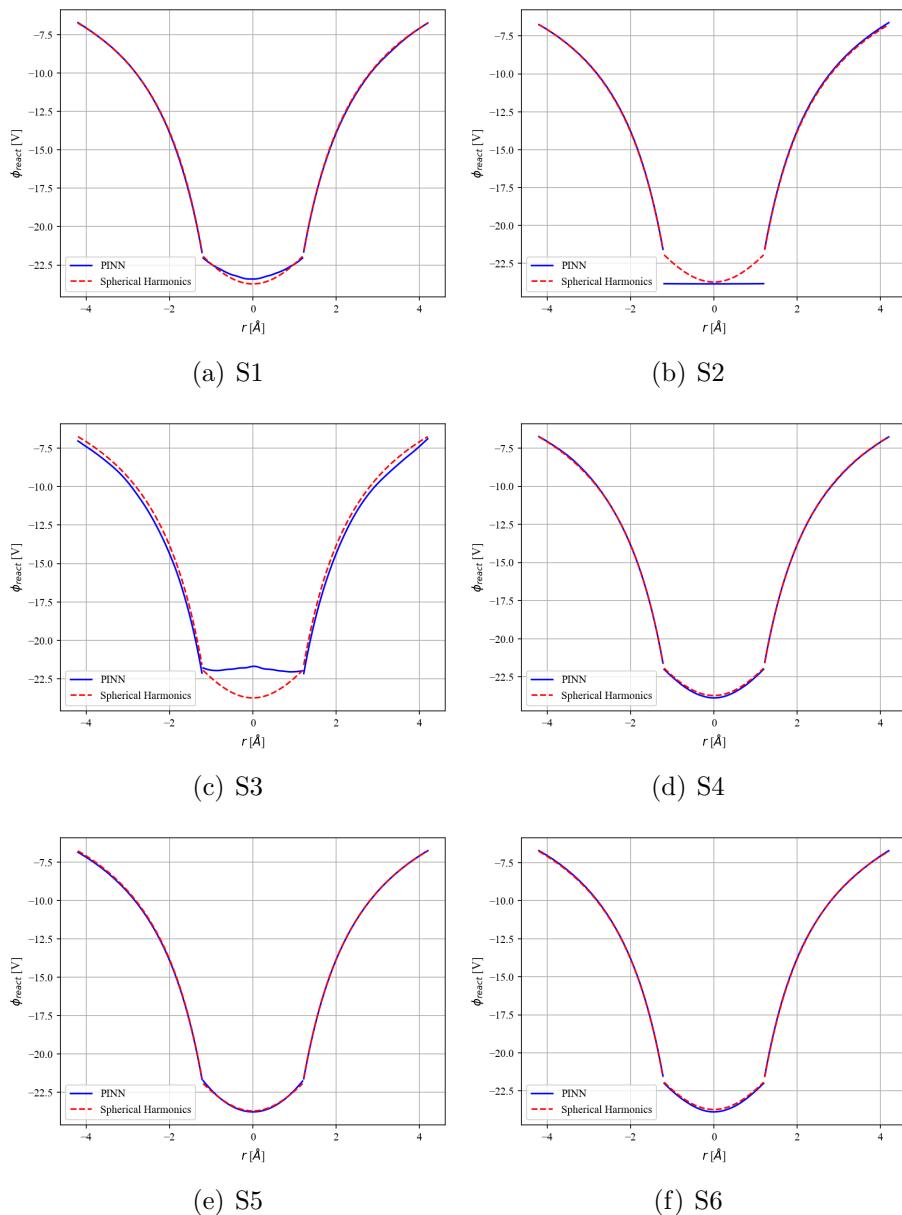


Figura 4.27: Potencial de reacción para 2 cargas positivas, dirección $\theta = \pi/2$, $\phi = \pi$

4.3.3. 2 Cargas de Signos Opuestos

El tercer caso del multipolo esférico es el uso de 2 cargas descentradas de signo opuesto. Con este caso, se revisará el efecto de la carga neta de la molécula en la predicción del potencial de reacción. Los resultados obtenidos se presentan en la tabla 4.18.

Tabla 4.18: Resultados para 2 cargas de signos opuestos

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
S1	-99.99	2.84E-02	3.40E-03	4.38E-04	1.82E-03
S2	-97.25	2.02E-04	3.76E-02	1.10E-02	1.08E-02
S3	-101.62	4.52E-02	1.57E-02	1.09E-04	7.54E-04
S4	-98.18	9.84E-03	1.77E-03	1.42E-03	2.72E-03
S5	-101.25	4.13E-02	5.23E-03	5.94E-03	6.03E-03
S6	-97.01	2.19E-03	2.52E-03	2.12E-03	3.07E-03

De la tabla anterior, se visualiza que S2 predice con mejor exactitud la energía de solvatación, pero con un error para el potencial en la interfaz y losses más altos que los otros casos. Además, se nota que solo S4 y S6 lograron errores en la energía de solvatación y el potencial en la interfaz del orden de 10^{-3} .

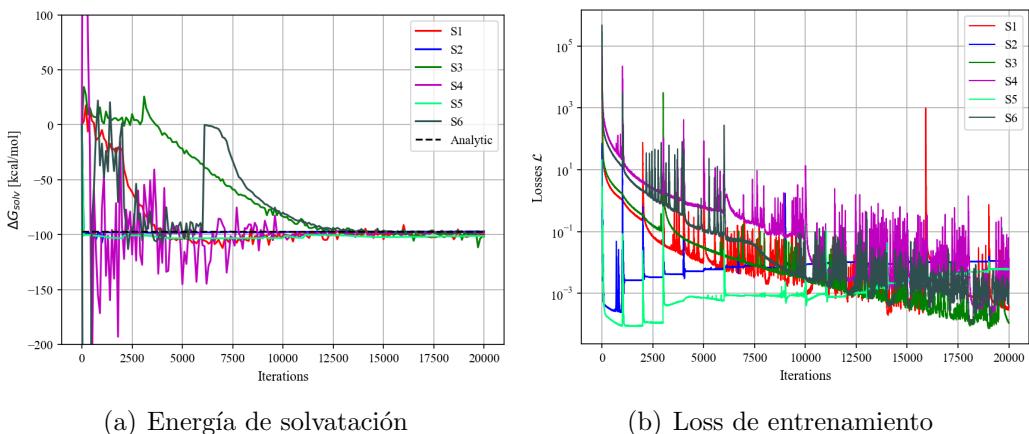


Figura 4.28: Energía de solvatación y loss de entrenamiento para 2 cargas de signos opuestos

En la figura 4.28 se puede revisar la evolución de la energía de solvatación y el loss de entrenamiento para los distintos casos de prueba. Es curioso notar que S3 y S6 (los casos que usan factorización de parámetros) recién predicen correctamente la energía de solvatación desde la iteración 12500. Sin embargo, su loss de entrenamiento se ve decreciente en todo momento. Otro resultado a destacar es la evolución del loss de S5. Consistentemente ha mostrado una rápida

4. Resultados y Análisis

bajada en las iteraciones tempranas, pero en este caso después de alcanzar el mínimo, empezó a aumentar de manera monótona.

Revisando la figura 4.29 se puede visualizar la predicción del potencial de reacción para los distintos casos estudiados. En estas gráficas se nota una muy buena representación por parte de S1 y S4 (los casos que solo usan la capa de Fourier).

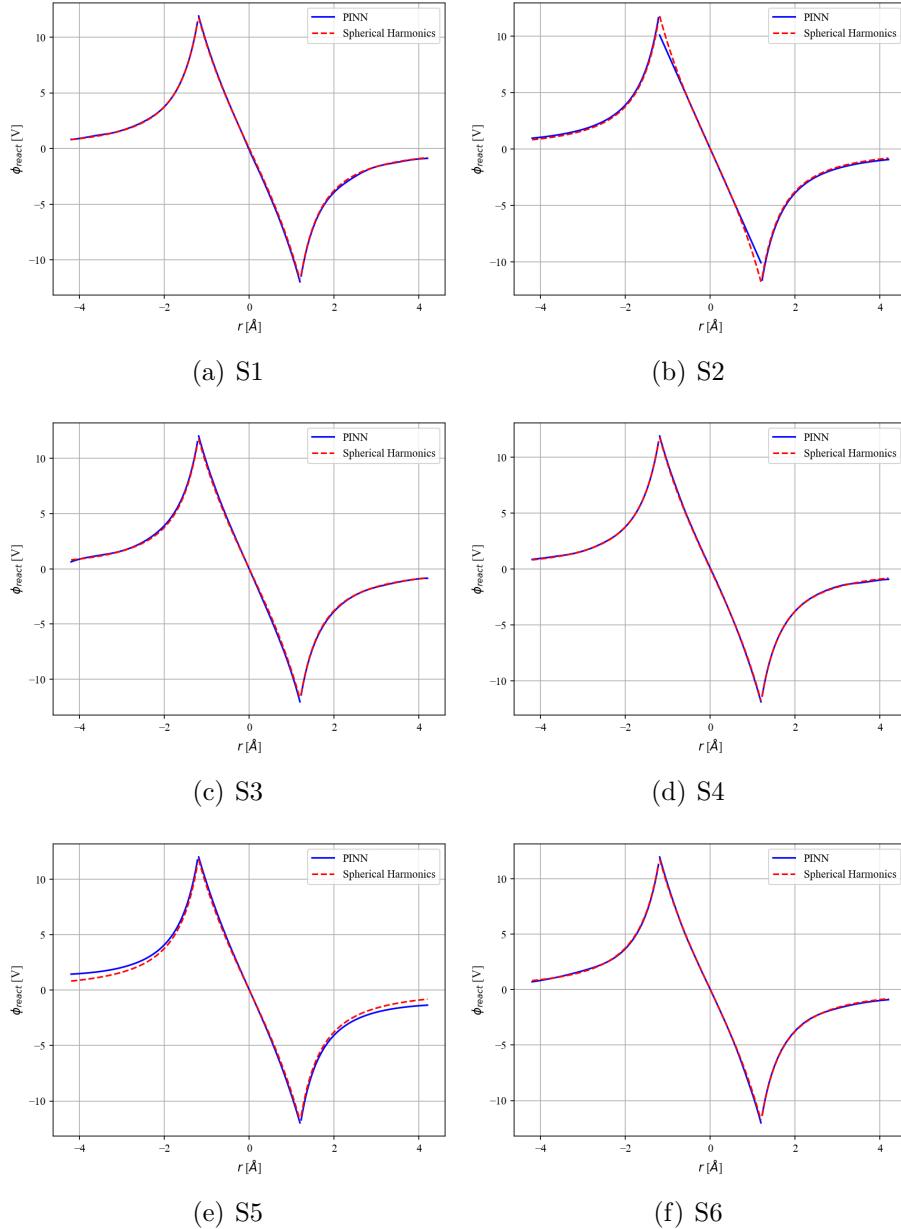


Figura 4.29: Potencial de reacción para 2 cargas opuestas, dirección $\theta = 0$, $\phi = \pi/2$

Además, sorprendentemente S3 logra una buena representación y S5 genera una gran desviación en la zona del solvente. Esto es opuesto a lo obtenido para los 2 casos del multipolo estudiados

anteriormente.

Revisando la dirección graficada en la figura 4.30, se nota que todos los casos generan oscilaciones en donde debería existir un valor constante. Sin embargo, estas oscilaciones son de amplitud del 1% aproximadamente respecto al valor máximo del potencial de reacción. Las amplitudes para los casos estudiados están entre los 0.2 [V] y 0.02 [V].

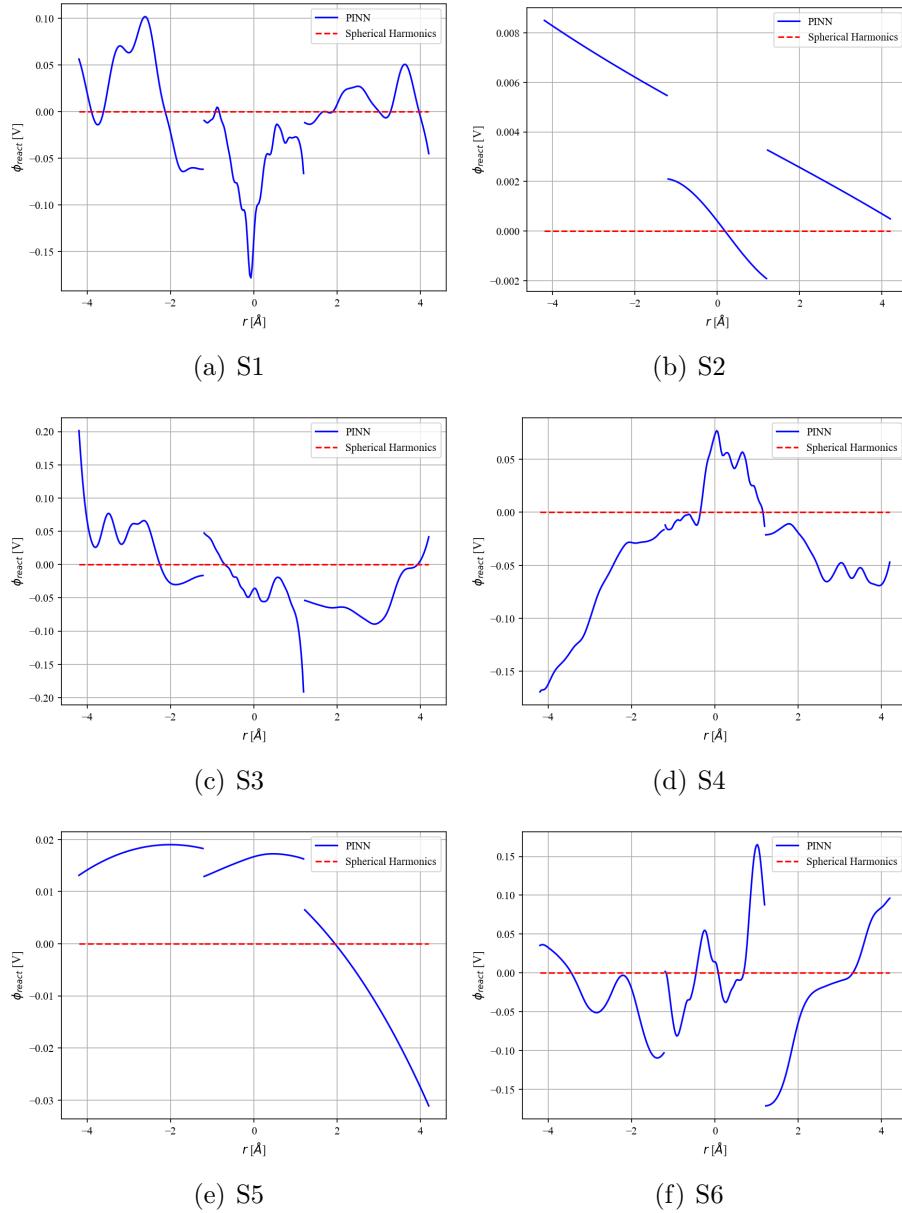


Figura 4.30: Potencial de reacción para 2 cargas opuestas, dirección $\theta = \pi/2$, $\phi = \pi$

4.3.4. Colección de Cargas

El último caso a estudiar para el multipolo esférico es la colección de 5 cargas dentro de la esfera. Esta validación revelará el efecto que tiene la cantidad de cargas en la precisión del método de PINNs.

Los resultados obtenidos se presentan en la tabla 4.19.

Tabla 4.19: Resultados para colección de cargas

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
S1	-557.32	1.56E-02	8.33E-03	2.65E-04	1.81E-03
S2	-559.52	1.96E-02	4.27E-02	5.38E-02	5.33E-02
S3	-565.52	3.05E-02	1.60E-02	3.25E-04	2.09E-03
S4	-552.83	7.40E-03	5.58E-03	3.08E-03	7.29E-03
S5	-565.90	3.12E-02	2.57E-02	6.57E-02	6.41E-02
S6	-556.08	1.33E-02	1.22E-02	1.62E-03	1.24E-02

De la tabla anterior, se nota que solo S4 logró una predicción de la energía de solvatación del orden de 10^{-3} . Además, S1 y S4 lograron un orden de 10^{-3} en la predicción del potencial en la interfaz. Por otra parte, revisando la figura 4.31, se puede visualizar como nuevamente S2 y S5 logran el loss más bajo en iteraciones tempranas, pero luego de este mínimo comienza a crecer de manera monótona. Esto puede implicar que el uso de la capa de Fourier ralentiza la convergencia en iteraciones tempranas, pero logra obtener menores errores al compararlo con los casos en donde no se utiliza.

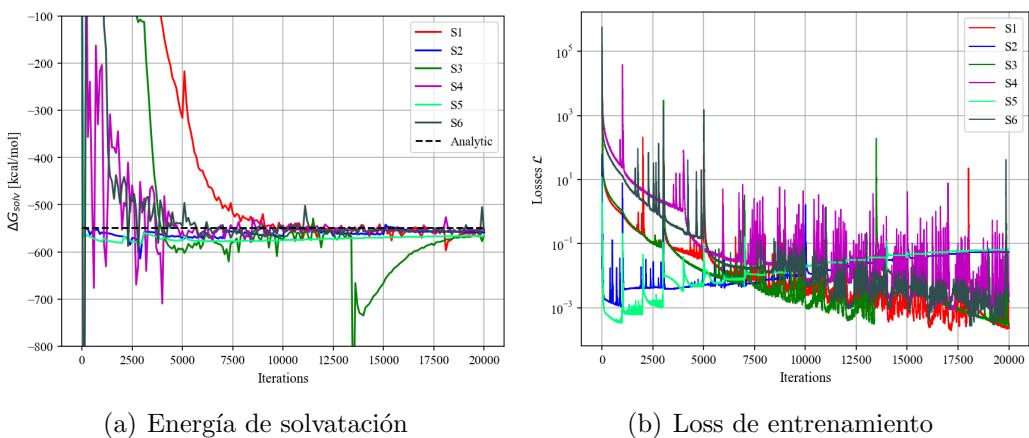


Figura 4.31: Energía de solvatación y loss de entrenamiento para colección de cargas

Todos los casos logran una convergencia al valor correcto de energía de solvatación, destacando que S6 generó una peak inesperado.

Revisando las figuras 4.32 y 4.33, se puede visualizar como S1 y S4 logran la mejor representación para el potencial de reacción. Ambos casos fueron consistentemente buenos en las distintas configuraciones de carga, y se mantuvieron como superiores para la colección de cargas.

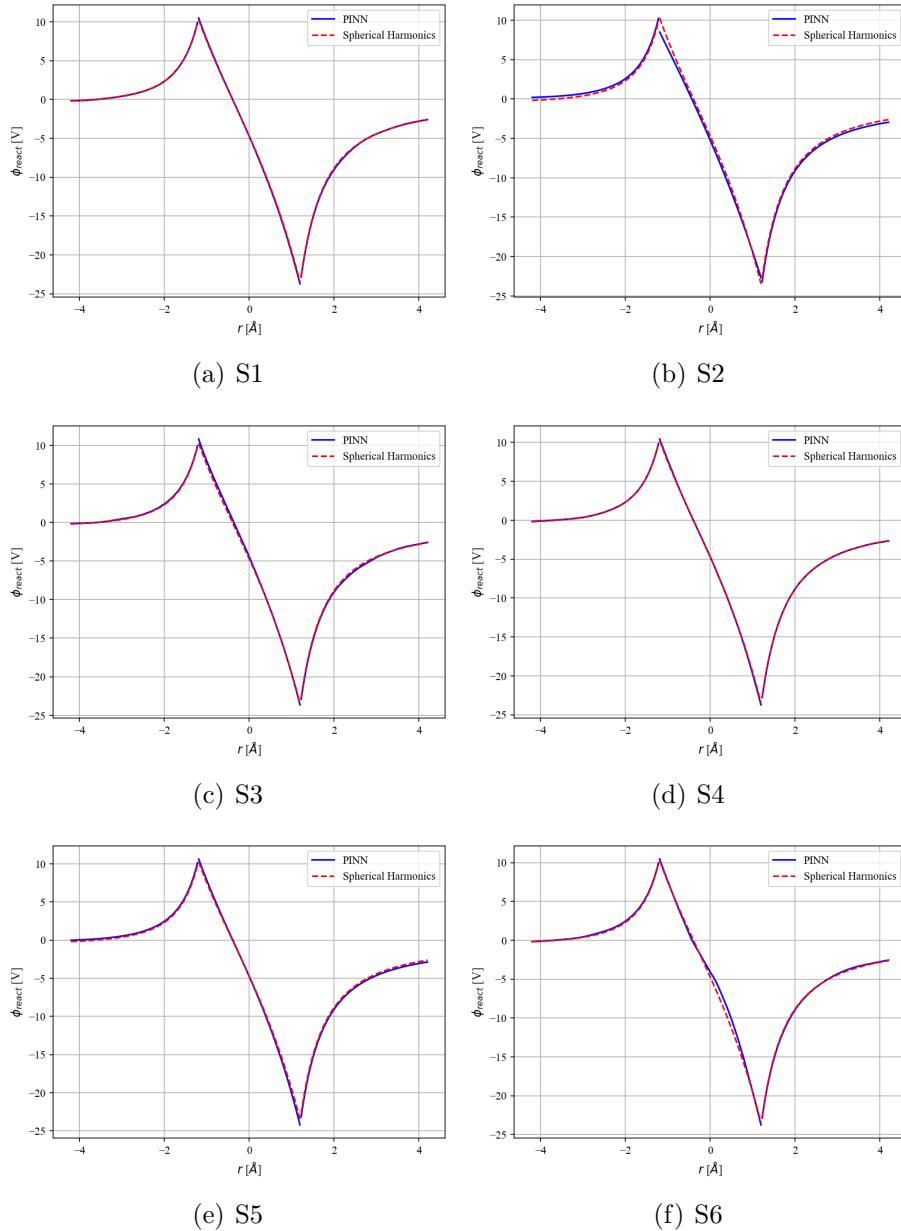


Figura 4.32: Potencial de reacción para colección de cargas, dirección $\theta = 0$, $\phi = \pi/2$

Sorprende que S5 y S6, este último sobre todo, tengan desviaciones respecto a la solución analítica. Lo anterior podría significar que al aumentar la cantidad de cargas, la capa de Fourier se hace cada vez más relevante, de modo que logra representar los gradientes que se generan por las cargas puntuales. Además, pareciera ser que la factorización de los pesos de la red no ayuda mucho en aumentar la precisión del método, generando oscilaciones no deseadas en la

4. Resultados y Análisis

convergencia de la energía de solvatación.

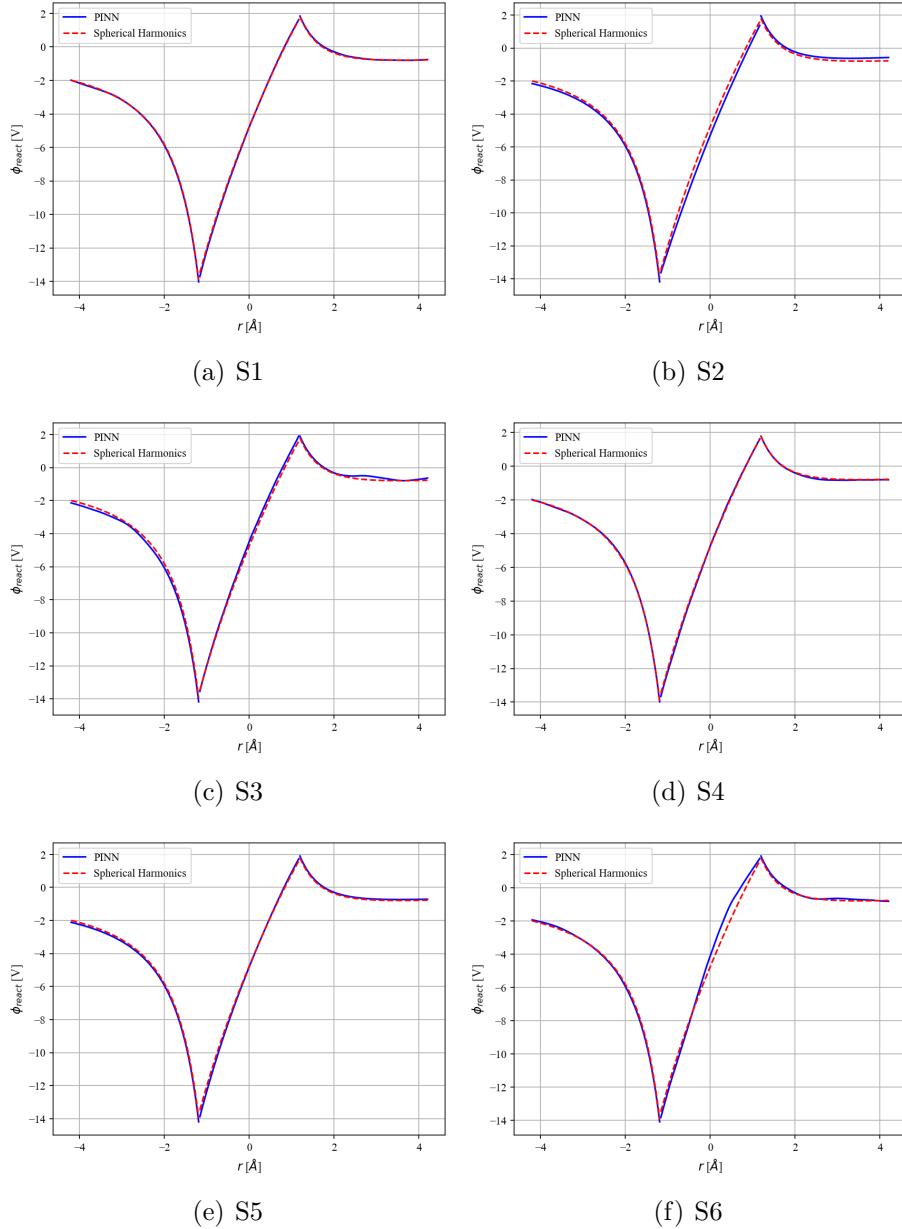


Figura 4.33: Potencial de reacción para colección de cargas, dirección $\theta = \pi/2$, $\phi = \pi/2$

4.4. Metanol

La primera molécula real que se revisó fue el metanol. Esta molécula tiene 6 cargas puntuales y se distribuyen como se muestra en la figura 4.34. Pese a que el metanol sea una molécula relativamente pequeña, sirve como una primera revisión del efecto de la geometría en la precisión del método.

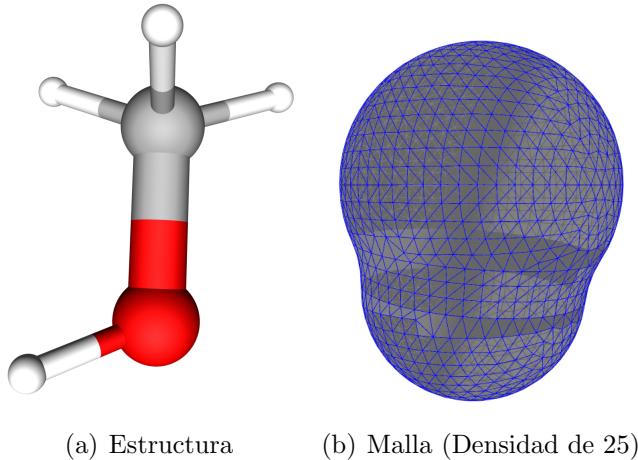


Figura 4.34: Estructura y malla para el metanol

Para todas las simulaciones, se utilizaron las siguientes dimensiones y/o propiedades:

- Constante dieléctrica soluto: $\epsilon_1 = 1 [\epsilon_0]$
- Constante dieléctrica solvente: $\epsilon_2 = 80 [\epsilon_0]$
- Inverso de la longitud de Debye: $\kappa = 0.125 [1/\text{\AA}]$
- Radio esfera frontera solvente: $R = 5.6 [\text{\AA}]$

Para el metanol se realizaron 3 simulaciones. Estas corresponden a las que se consideraron que funcionarían mejor, teniendo en cuenta los resultados obtenidos para el multipolo esférico. Estos casos se pueden revisar en la tabla 4.20.

Tabla 4.20: Casos de prueba para el metanol⁶

Simulación	Arquitectura	C. Fourier	WF
M1	MLP	x	
M4	ModMLP	x	
M6	ModMLP	x	x

⁶Se mantiene la misma numeración que el multipolo esférico para evitar confusión.

4. Resultados y Análisis

Todos estos casos fueron utilizados para resolver el esquema de regularización 2 de la ecuación de PB lineal. Respecto a los puntos de colocación, se utilizaron los mostrados en la tabla 4.21, incluyendo datos conocidos en el solvente dados por la función de Green de Yukawa. Los resultados se compararán con los obtenidos por el software de BEM (PBJ) utilizando la misma malla superficial.

Tabla 4.21: Puntos de colocación para el metanol

Nodos R1	Nodos R2	Nodos I	Nodos D2	Nodos K2
8048	17048	2538	1280	286

En la tabla 4.22 se pueden visualizar los resultados obtenidos. Notar que para M1 (arquitectura MLP) se tiene un error en la energía de solvatación del orden de 10^{-3} . Asimismo, M4 también logra un error del mismo orden en la energía de solvatación, pero también en el error L_2 para el potencial de reacción en la interfaz.

Tabla 4.22: Resultados para los casos de prueba del metanol

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
M1	-11.64	1.87E-03	2.54E-02	1.54E-05	2.47E-05
M4	-11.58	7.27E-03	8.76E-03	1.16E-04	1.22E-04
M6	-11.58	7.27E-03	1.53E-02	1.88E-05	1.78E-05

En la figura 4.35 se puede visualizar la evolución de la energía de solvatación y el loss de entrenamiento a medida que aumentan las iteraciones. Para la energía de solvatación, se nota como todos los casos convergen a la solución obtenida con BEM. Sin embargo, se nota que existen oscilaciones que no disminuyen con las iteraciones.

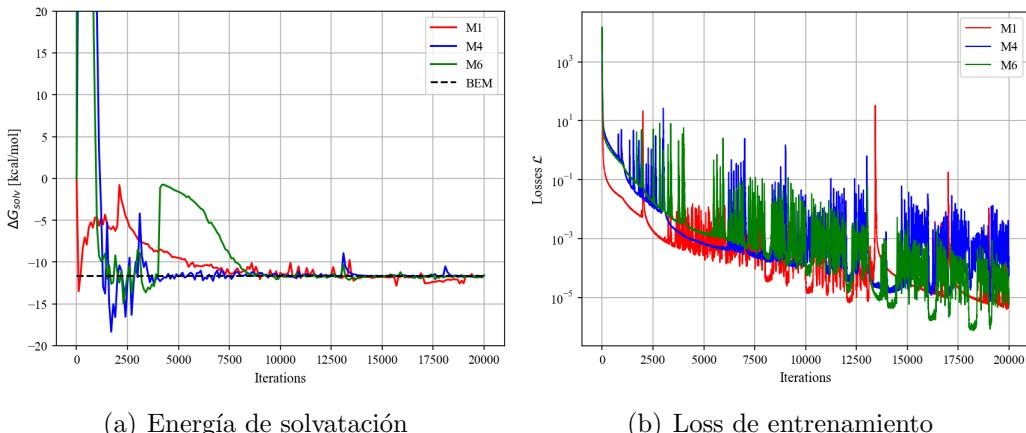


Figura 4.35: Energía de solvatación y loss de entrenamiento para los casos de prueba del metanol

Para el loss de entrenamiento, se nota como en los 3 casos se tienen comportamientos decrecientes, pero con gran amplitud de oscilaciones.

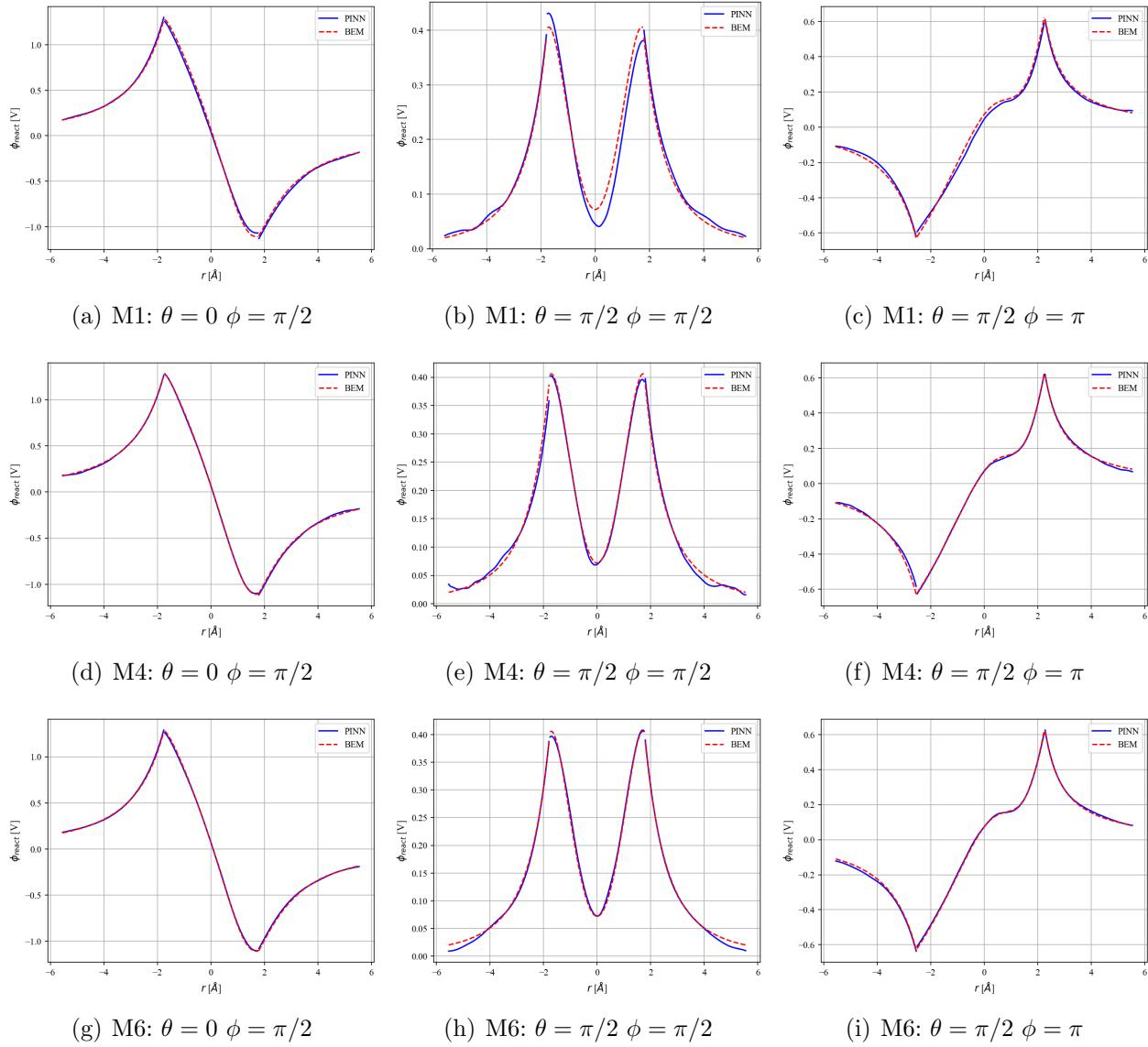


Figura 4.36: Potencial de reacción para el metanol en 3 dirección diferentes

La predicción del potencial en 3 direcciones distintas se puede visualizar en la figura 4.36. Se nota que en los 3 casos se tiene dificultad para representar la dirección dada por los ángulos $\theta = \pi/2, \phi = \pi/2$. Esta dirección coincide con la posición de 2 cargas de la molécula. De todas formas, este fenómeno no se evidenció en el multipolo esférico, donde las gráficas pasaban por las ubicaciones de las cargas en casi todos los casos.

En la figura 4.37 se puede visualizar la predicción del potencial de reacción en la superficie, en comparación con la solución obtenida con BEM. Notar que todos los casos muestran un

4. Resultados y Análisis

comportamiento similar, coincidiendo con la solución de referencia.

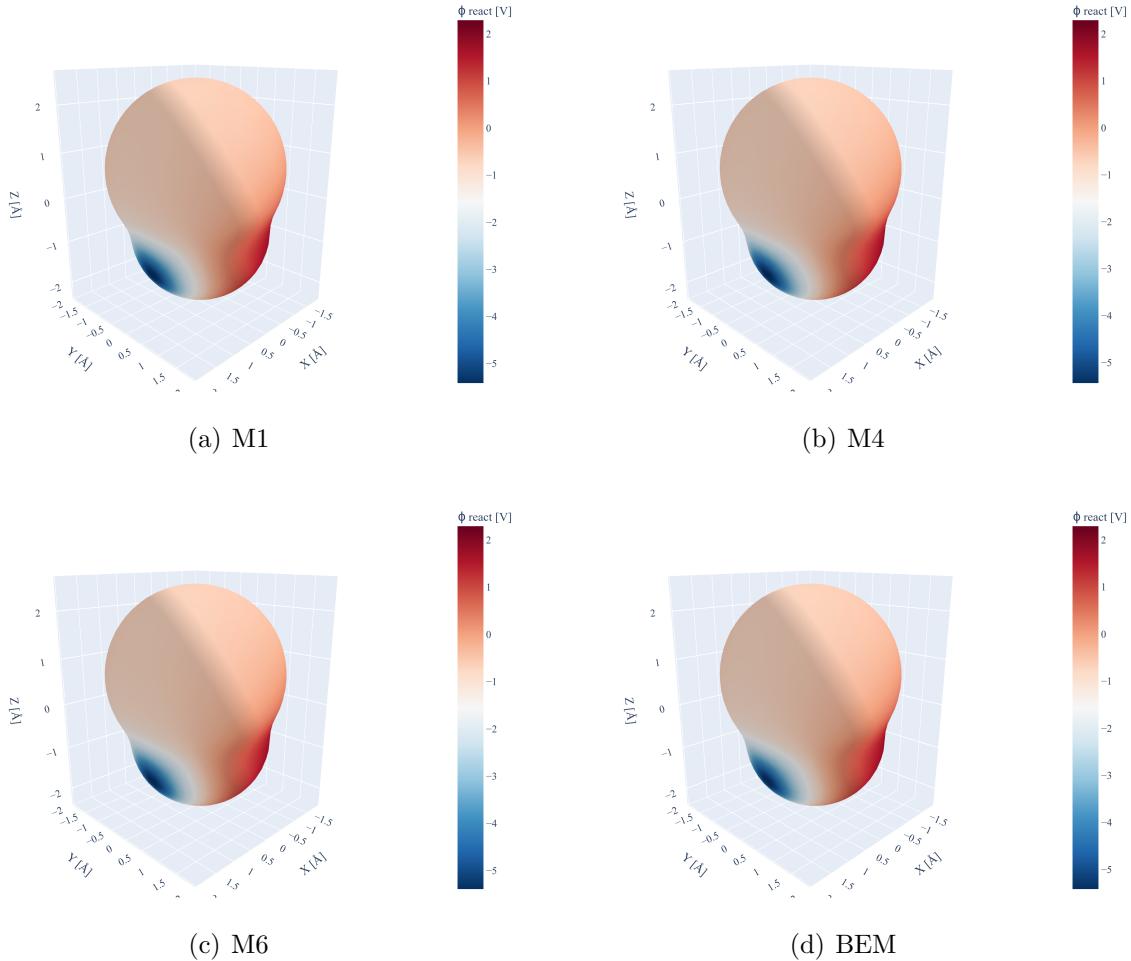


Figura 4.37: Potencial de reacción en la interfaz del metanol

Por último, en la figura 4.38 se muestra el error absoluto y relativo en la predicción del potencial de reacción en la superficie de la molécula. Se evidencia que los errores tienen una distribución similar, donde los valores más bajos se obtienen para el caso M4.

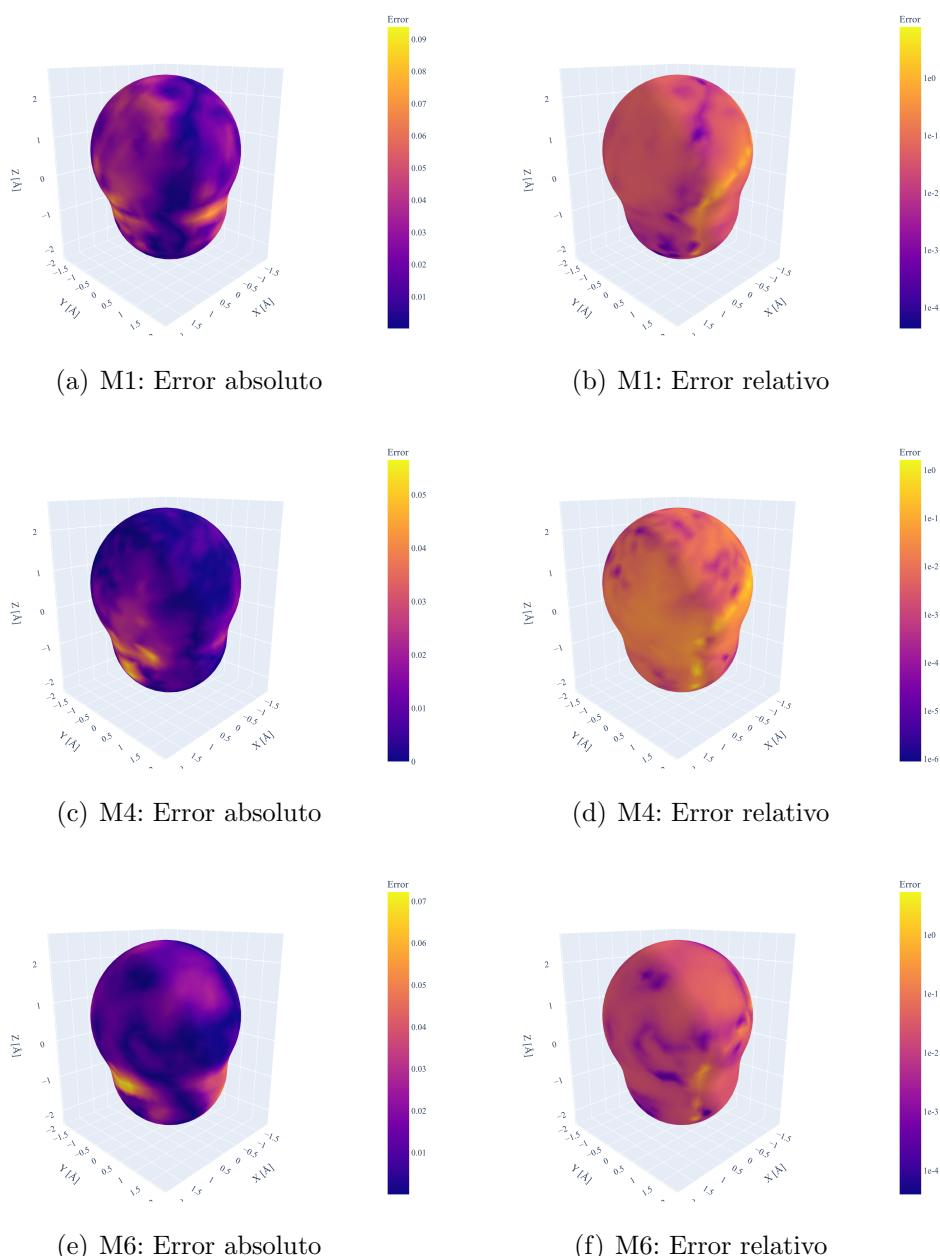


Figura 4.38: Error absoluto y relativo en la interfaz del metanol

4. Resultados y Análisis

Considerando los resultados mostrados, se decide repetir los mismos casos anteriores con la diferencia de no usar datos conocidos (K2). Esto revelará como el método varía al no usar datos conocidos en una molécula no esférica para estas 3 arquitecturas. En la tabla 4.23 se presentan estos resultados.

Tabla 4.23: Resultados al probar mismos casos del metanol sin datos puntuales

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
M1 sin K2	-11.63	2.55E-03	1.99E-02	3.13E-06	2.08E-06
M4 sin K2	-11.73	5.68E-03	5.44E-02	1.17E-03	1.16E-03
M6 sin K2	-11.73	6.28E-03	2.22E-02	2.09E-05	1.99E-05

Respecto a estos resultados, los 3 casos muestran errores para la energía de solvatación y para el potencial en la interfaz de órdenes de 10^{-3} y 10^{-2} respectivamente, siendo M1 el caso con menores errores. Además, estos valores tienen poca variación en comparación a los casos utilizando K2. Por otra parte, en la figura 4.39 se puede visualizar que la energía de solvatación tiene una convergencia más lenta, y los losses de entrenamiento tienen comportamientos bastante similares a los casos utilizando K2.

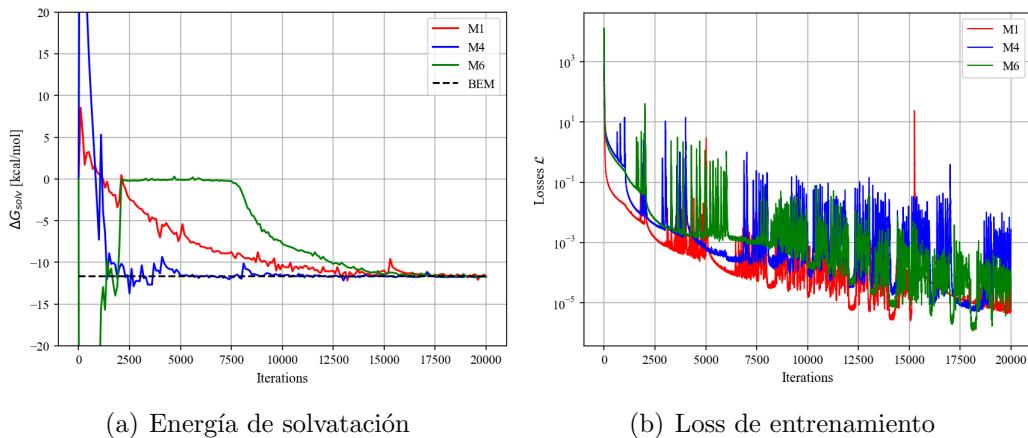


Figura 4.39: Energía de solvatación y loss de entrenamiento al no usar datos puntuales para el metanol

En la figura 4.40 se puede revisar el comportamiento del potencial de reacción en 3 direcciones. A simple vista pareciera ser que M1 no logra un gran cambio en la predicción del potencial al no utilizar datos conocidos, en comparación a M4 y M6. Además, se sigue teniendo una mala predicción para la dirección dada por los ángulos: $\theta = \pi/2$, $\phi = \pi/2$.

Por otra parte, se puede revisar como claramente los casos M4 y M6 empeoran al no utilizar datos conocidos. Esto quizás demuestra que la arquitectura ModMLP es más sensible al uso de

datos conocidos, en comparación a la arquitectura MLP, haciéndose dependiente de ellos. Esto puede generar un problema en moléculas más grandes donde la función de Green de Yukawa no sea una tan buena aproximación.

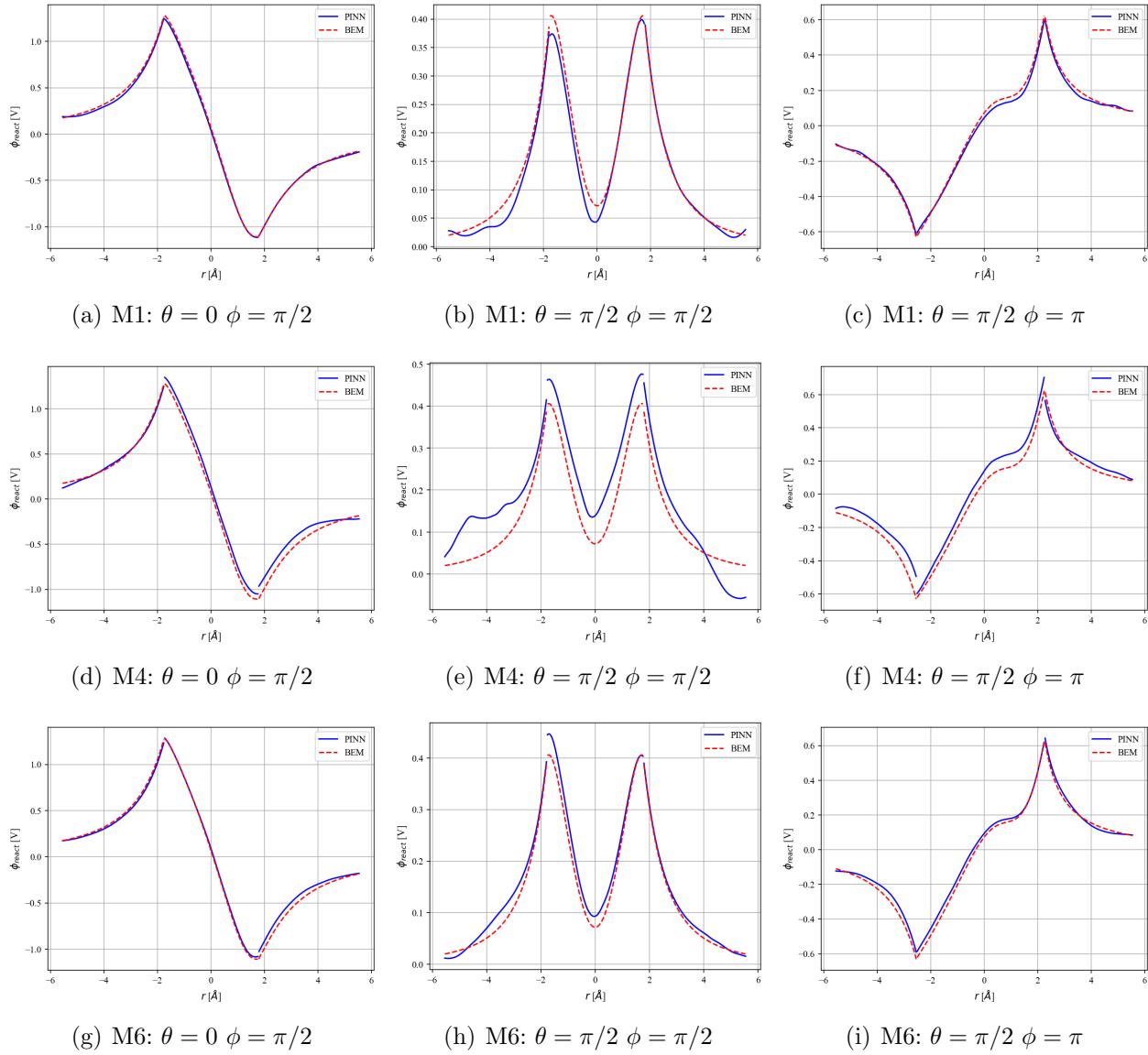


Figura 4.40: Potencial de reacción para el metanol en 3 dirección diferentes

De la misma forma que para el caso anterior, se grafica el potencial de reacción en la superficie de la molécula. Esto se puede visualizar en la figura 4.41. Los 3 casos muestran un comportamiento similar, en comparación a la solución obtenida con BEM.

4. Resultados y Análisis

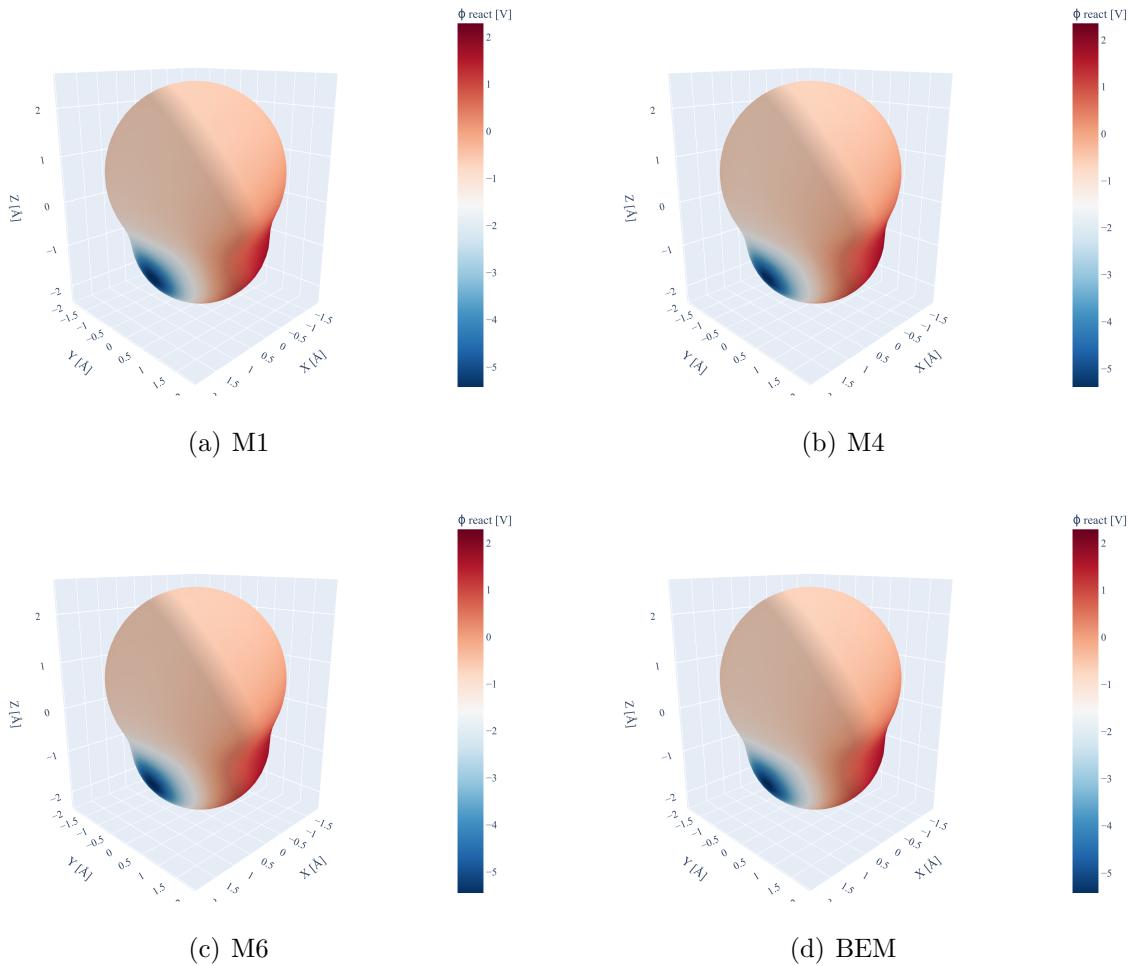


Figura 4.41: Potencial de reacción en la interfaz del metanol

Por último, en la figura 4.42 se puede revisar el error absoluto y el error relativo en la predicción del potencial de reacción en la interfaz de la molécula. Se nota que el comportamiento del error para M1 es casi idéntico, demostrando su robustez al uso de datos conocidos. Sin embargo, se visualiza un cambio en las simulaciones M4 y M6. Este cambio se evidencia por máximos distintos y en distintas zonas.

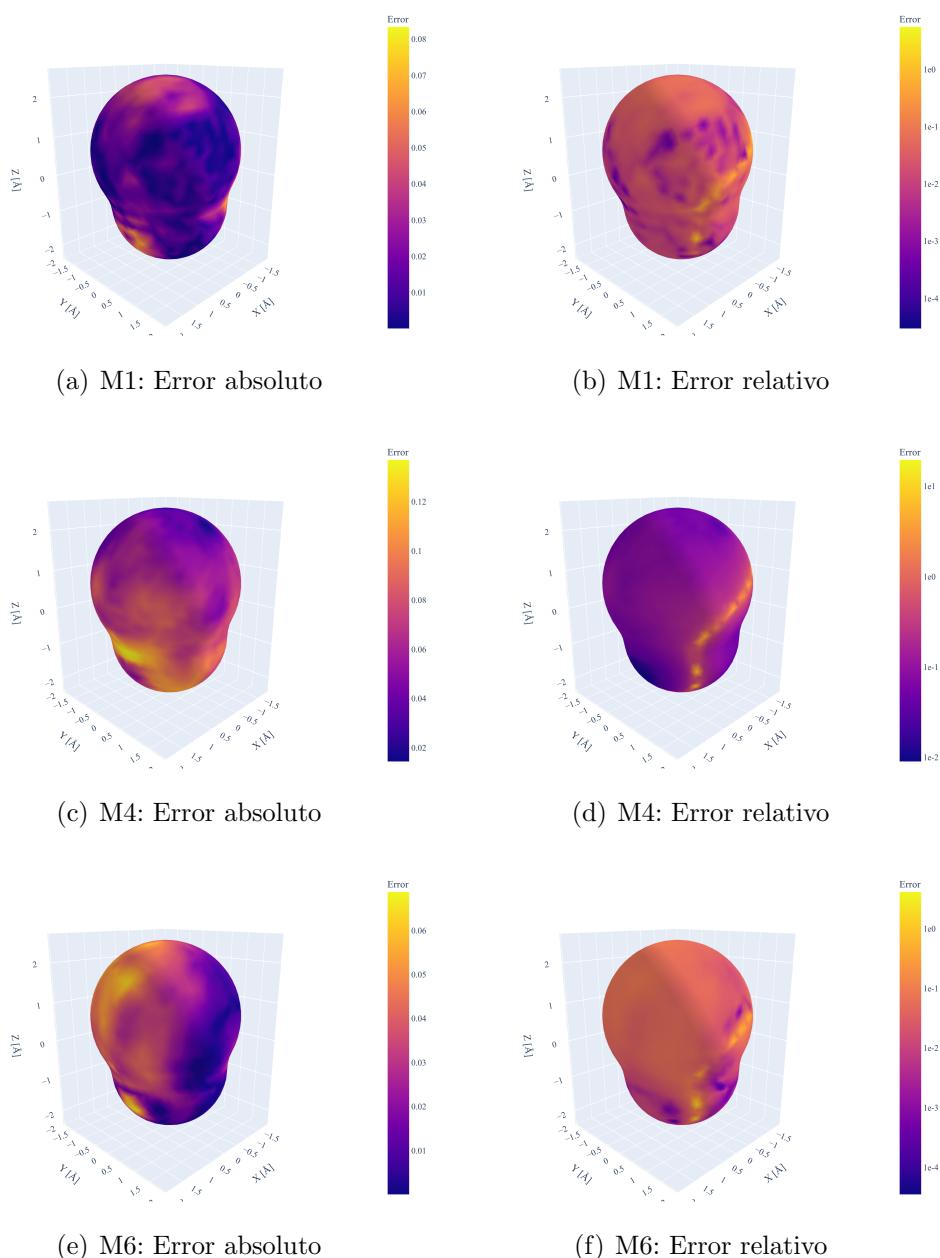


Figura 4.42: Error absoluto y relativo en la interfaz del metanol

4.5. Arginina

La segunda molécula real en la que se probó el método de PINNs para resolver la ecuación de PB es la arginina. Esta molécula posee 27 cargas en su estructura, y tiene una geometría más compleja que el metanol, tal como se visualiza en la figura 4.43.

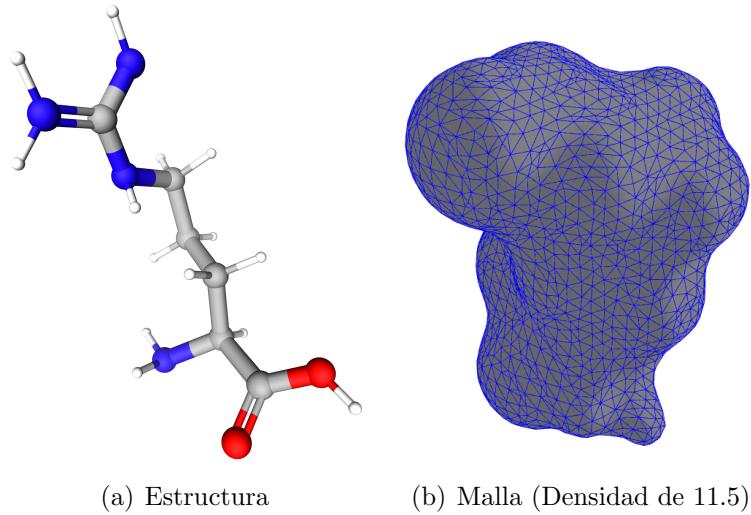


Figura 4.43: Estructura y malla para la arginina

Para todas las simulaciones de la arginina, se utilizaron las siguientes dimensiones y/o propiedades, excepto cuando se diga lo contrario:

- Constante dieléctrica soluto: $\epsilon_1 = 2 [\epsilon_0]$
- Constante dieléctrica solvente: $\epsilon_2 = 80 [\epsilon_0]$
- Inverso de la longitud de Debye: $\kappa = 0.125 [1/\text{\AA}]$
- Radio esfera frontera solvente: $R = 9.5 [\text{\AA}]$

En todos los casos a probados con esta molécula, se resolvió el esquema de regularización 2 de la ecuación de PB lineal. Respecto a los puntos de colocación, se utilizarán los mostrados en la tabla 4.24. Los resultados se compararán con los obtenidos con el software de BEM (PBJ) utilizando la misma malla superficial.

Tabla 4.24: Puntos de colocación para la arginina⁷

Nodos R1	Nodos R2	Nodos I	Nodos D2	Nodos K2
20412	46459	4728	5120	300

⁷En los casos que se utilizaron datos conocidos en el solvente, se obtuvieron a partir de la función de Green de Yukawa.

Para la arquitectura, solo se utilizó la arquitectura MLP (igual a la utilizada con M1 en las simulaciones del metanol). Esto es debido a la consistencia y robustez presentada en las pruebas pasadas. Para continuar con las pruebas del metanol, aplicadas a la arginina, en primera instancia se revisó el efecto de la adición de los datos conocidos en el solvente en la función de pérdida. Estos resultados se pueden revisar en la tabla 4.25.

Tabla 4.25: Resultados para la arginina, uso de datos puntuales

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Con K2	-129.77	5.91E-03	2.10E-02	4.49E-04	4.05E-04
Sin K2	-131.39	6.46E-03	1.06E-02	2.79E-05	2.38E-05

En la tabla 4.25 se puede revisar que el único impacto considerable al no usar datos conocidos es la reducción en un orden de magnitud en los losses de entrenamiento y validación, alcanzando un orden de 10^{-5} . Por otra parte, en la figura 4.44 se puede revisar la evolución de la energía de solvatación y el loss de entrenamiento. A partir de esto, ambos casos reflejan un comportamiento similar.

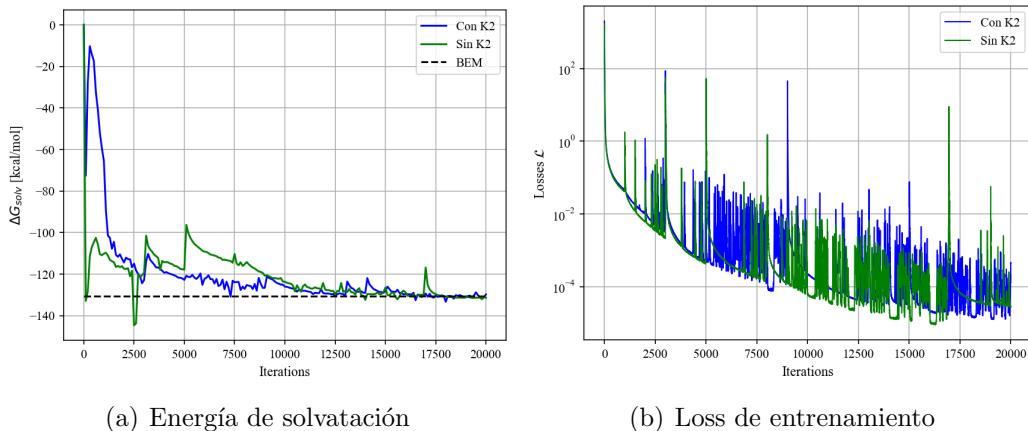


Figura 4.44: Energía de solvatación y loss de entrenamiento para los casos de prueba de la arginina

4. Resultados y Análisis

Pese a que los indicadores son similares, la figura 4.45 demuestra que el no usar datos conocidos, mejora sustancialmente la predicción del potencial de reacción. Notar que este caso logra ajustarse a la solución obtenida con BEM, en comparación con el uso de datos conocidos, la cual genera desviaciones y oscilaciones no deseadas.

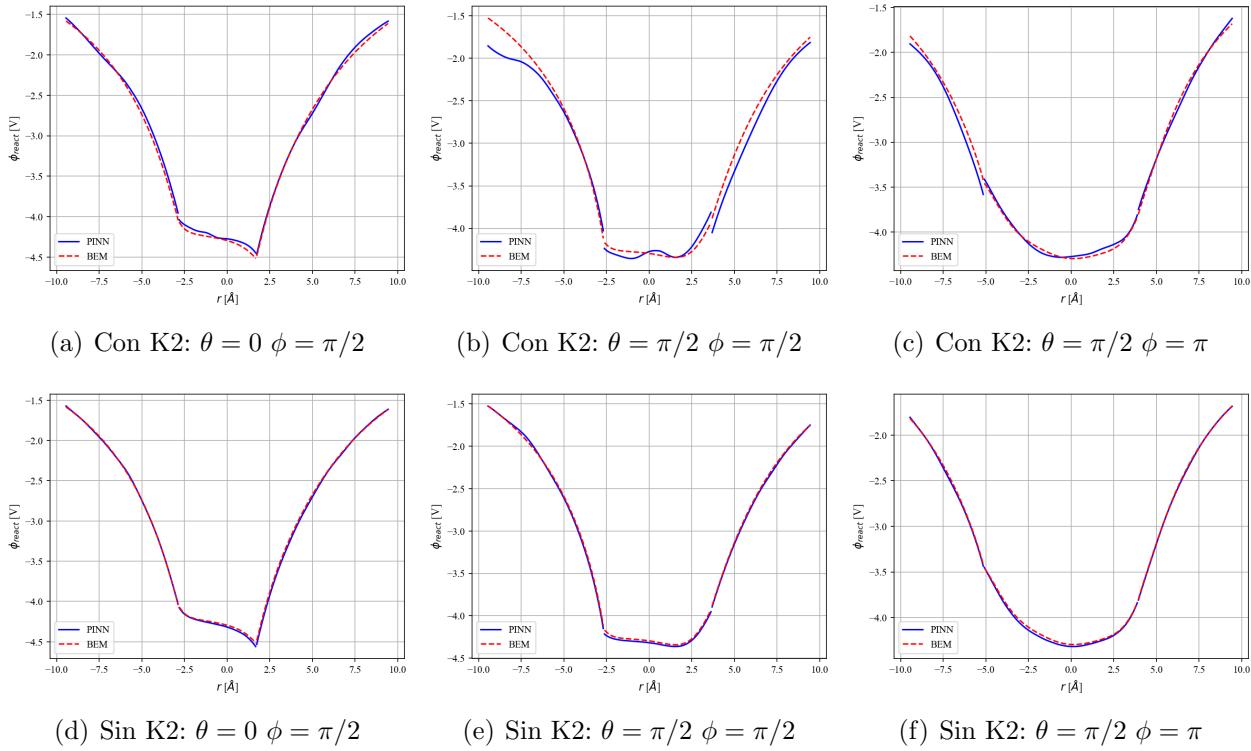


Figura 4.45: Potencial de reacción para la arginina en 3 direcciones diferentes

Este mismo comportamiento no se evidenció con la arquitectura MLP en el metanol, en el cual la solución no mejoró ni empeoró al no utilizar datos conocidos. En la arginina, este fenómeno se puede deber a que simplemente utilizar datos interrumpe la minimización de los residuales, necesarios para ajustar la curvatura del potencial. También, se puede deber a que la función de Green de Yukawa no es una buena aproximación para el potencial en el solvente para esta molécula. De todas formas, el hecho de no necesitar datos conocidos trae una ventaja al método, ya que con moléculas más grandes y complejas, cada vez es más difícil encontrar una buena aproximación.

Pese a las diferencias evidenciadas en las curvas del potencial en distintas direcciones, el potencial de reacción en la interfaz de la molécula es similar en ambos casos, al compararlo con la solución obtenida con BEM. Coincidén los máximos y mínimos en las ubicaciones deseadas. Esto se puede visualizar en la figura 4.46

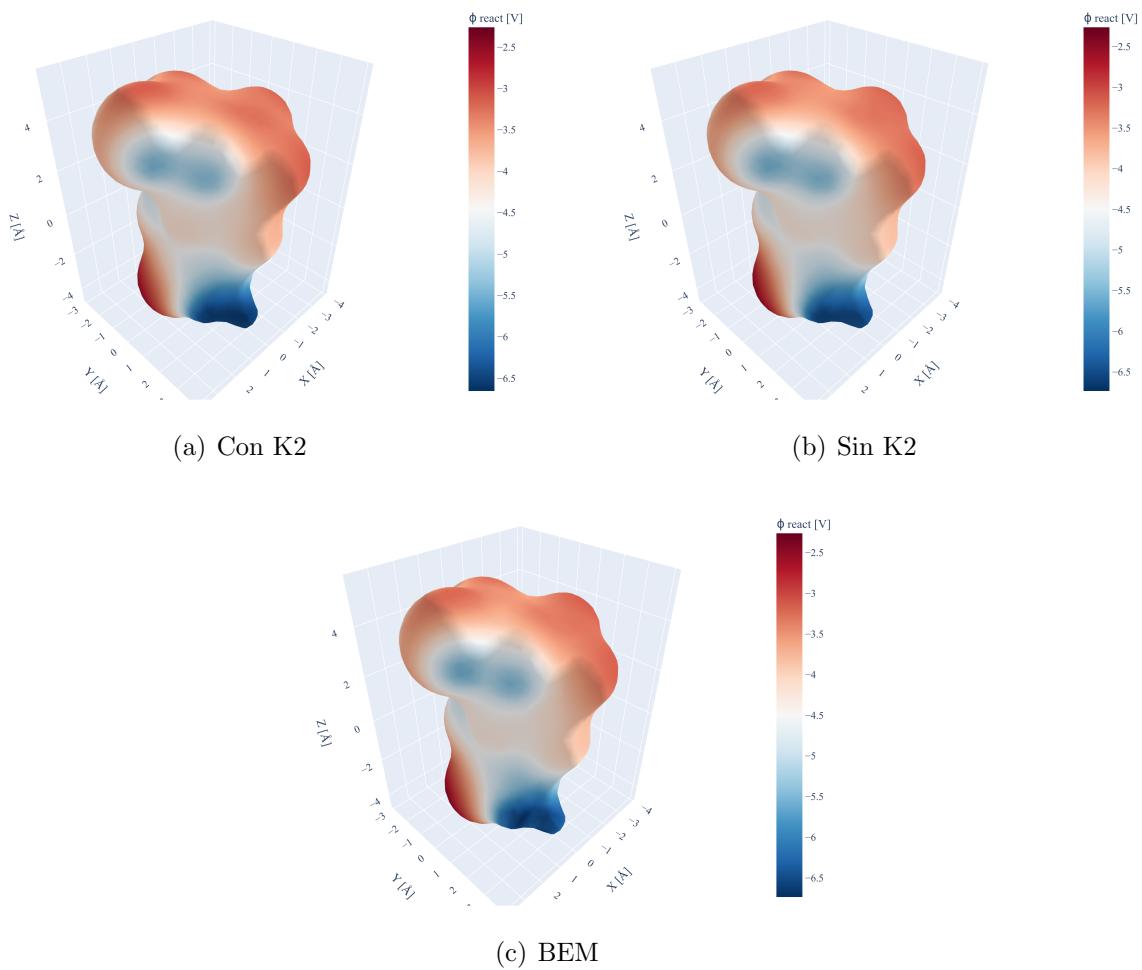


Figura 4.46: Potencial de reacción en la interfaz de la arginina

Por último, revisando la figura 4.47, se puede notar que el caso con datos conocidos tiene más zonas de alto error en la interfaz para el potencial de reacción, en comparación al caso sin datos conocidos.

4. Resultados y Análisis

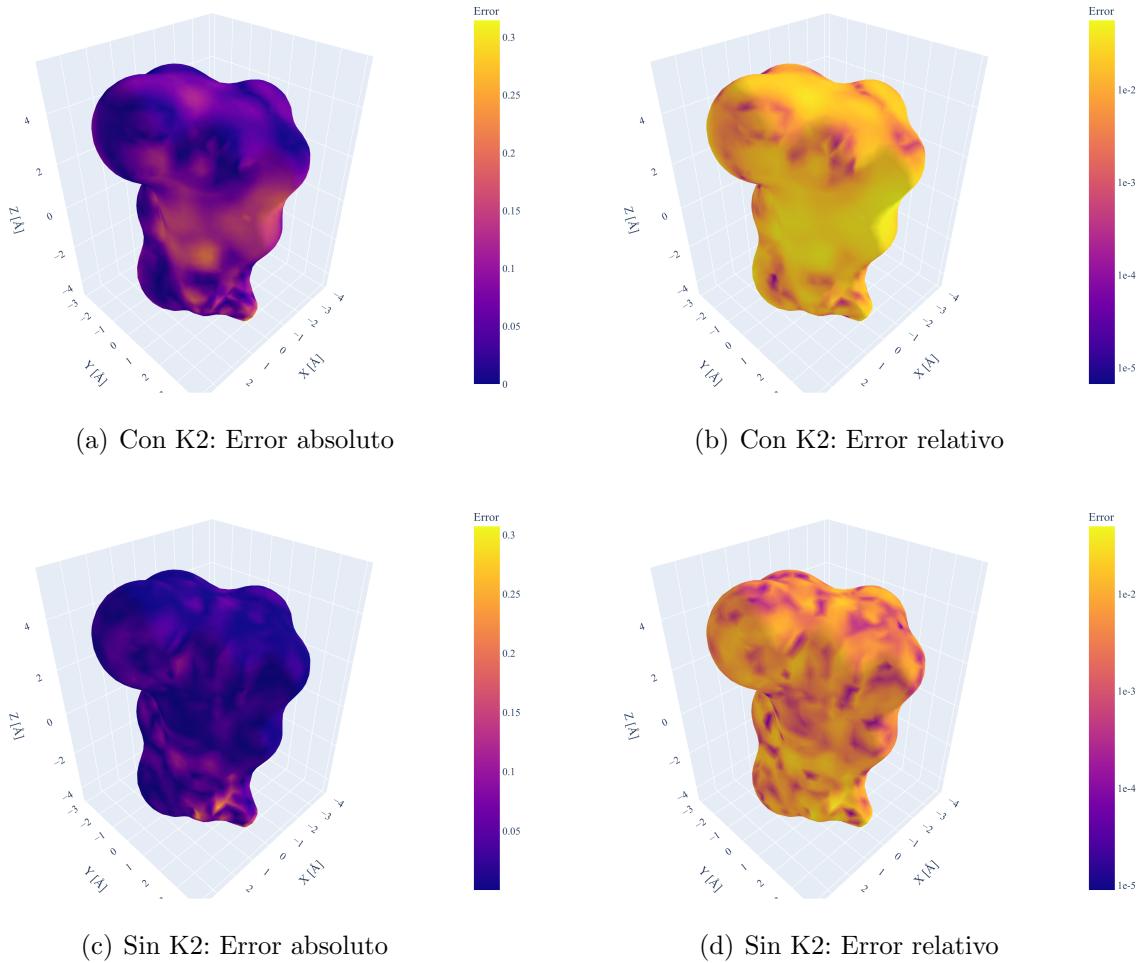


Figura 4.47: Error absoluto y relativo en la interfaz de la arginina

Para la misma malla utilizada, se encontró relevante añadir la capa del escalamiento del output, ya que se trabaja con una geometría más compleja y sin simetrías. En la tabla 4.26 se muestran 2 simulaciones, con y sin capa de escalamiento del output, sin utilizar datos conocidos.

Tabla 4.26: Resultados para la arginina, uso de la capa de escalamiento del output

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Sin Escal.	-131.39	6.46E-03	1.06E-02	2.79E-05	2.38E-05
Con Escal.	-131.54	7.55E-03	4.21E-03	2.24E-06	1.41E-06

En la tabla anterior se evidencia que el error en el potencial de reacción en la interfaz, y los losses de entrenamiento y validación, son menores en un orden de magnitud para el caso en que se utiliza la capa de escalamiento del output, lo que demuestra que es una buena adición al modelo para aumentar la precisión.

En la figura 4.48 se puede visualizar la evolución de la energía de solvatación y el loss de entrenamiento a medida que aumentan las iteraciones.

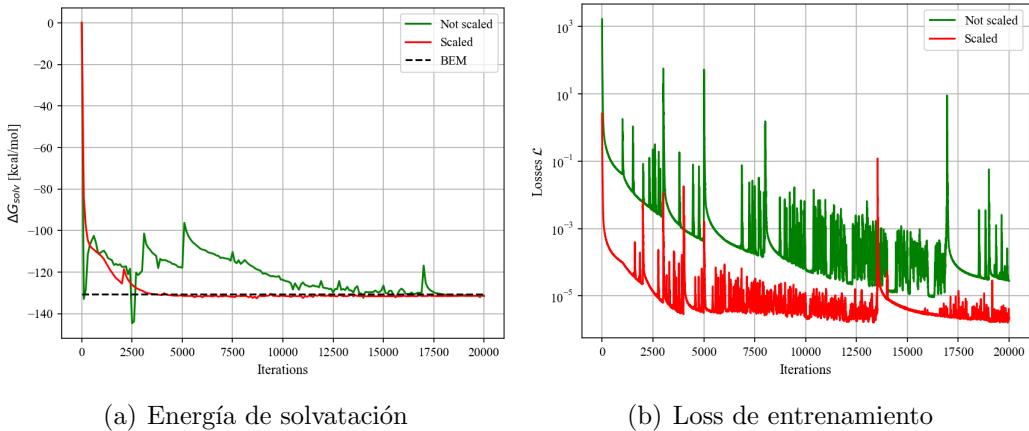


Figura 4.48: Energía de solvatación y loss de entrenamiento para los casos de prueba de la arginina

Se evidencia claramente que utilizar el escalamiento en el output mejora enormemente la convergencia en la energía de solvatación, alcanzando la solución dada por BEM en aproximadamente 3000 iteraciones, a diferencia del caso no escalado en que se alcanza esta estabilidad a las 18000 iteraciones con bastante ruido. Pese a esto, el error en la energía de solvatación para ambos modelos es relativamente el mismo (ver tabla 4.48).

Respecto al loss de entrenamiento, en todo el proceso de minimización el caso escalado logra un menor loss, lo que evidencia el efecto positivo en agregar esta capa. Además, provoca que la evolución del loss sea menos ruidosa.

Pareciera ser que la capa de escalamiento del output es útil para aumentar la convergencia y la precisión del modelo de PINNs, ya que logra “escalar” la predicción de la red hacia el orden correcto. Este efecto será vuelto a estudiar para moléculas más grandes en las secciones posteriores, lo que revelará aún más su importancia.

4.5.1. Estudio de Puntos de Colocación

Pese a que se realizó un estudio de puntos de colocación con el Ion de Born (sección 4.2.2), se decide realizar un nuevo estudio con la arginina. Esto es debido a que este estudio podría entregar nuevas conclusiones en comparación al Ion de Born, debido a que se tiene una geometría sin simetrías y que se podría clasificar como compleja. La idea será revisar los indicadores y las gráficas para 4 mallas distintas (utilizando muestreo aleatorio), variando el número de puntos de colocación⁸. Estas 4 mallas se pueden revisar en la tabla 4.27.

Tabla 4.27: Puntos de colocación para las distintas mallas de la arginina

Simulación	Nodos R1	Nodos R2	Nodos I	Nodos D2
Am1	3383	10413	282	1238
Am2	4418	10440	372	1238
Am3	6017	11596	624	1238
Am4	7568	15089	1318	1238

Recordar que para la obtención de los puntos de colocación, se generan 4 mallas, en donde los puntos de colocación coinciden con el número de elementos de cada malla. Esto implica que mientras más puntos de colocación se tengan en la superficie, más fina será esa malla superficial.

En la tabla 4.28 se pueden revisar los resultados obtenidos. Notar como todos los indicadores (errores y losses) mejoran al aumentar el número de puntos de colocación. Este resultado coincide con lo obtenido para el Ion de Born. Notar además que para estos resultados se tienen errores y losses mayores que los presentados en la tabla 4.25. Esto se debe a que para los resultados de la sección anterior se utilizaron mallas aún más finas.

Tabla 4.28: Resultados para estudio de malla de la arginina⁹

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Am1	-174.90	3.40E-01	8.91E-02	8.15E-04	3.36E-04
Am2	-145.99	1.18E-01	5.47E-02	3.07E-03	1.27E-04
Am3	-140.74	7.81E-02	3.20E-02	1.91E-04	2.81E-04
Am4	-134.72	3.20E-02	2.59E-02	8.63E-05	2.91E-05

Un resultado interesante se puede revisar en la figura 4.49. En esta figura se grafica el potencial de reacción en una dirección fijada. Notar que la solución de PINNs tiende a la solución de

⁸Se utilizó la misma arquitectura para ambas redes neuronales utilizadas en la sección anterior, pero sin el escalamiento del output para resaltar el efecto que generan los puntos de colocación.

⁹El error en la energía de solvatación se calcula respecto a la solución obtenida por una malla de BEM con una alta refinación (convergencia de malla). El error L_2 para el potencial se calcula respecto a la solución obtenida con BEM utilizando la misma malla superficial que PINNs.

BEM considerando la misma malla superficial. Esto principalmente se puede deber a que, al tratar con mallas tan gruesas, la geometría de la molécula es distinta, y por ende, no se está resolviendo el problema real. Además, es evidente como a medida que se utilizan más puntos de colocación, la solución se ajusta de mejor manera a la del código de BEM.

Este resultado refleja que es indispensable tener una buena malla superficial, de modo que represente correctamente la geometría molecular.

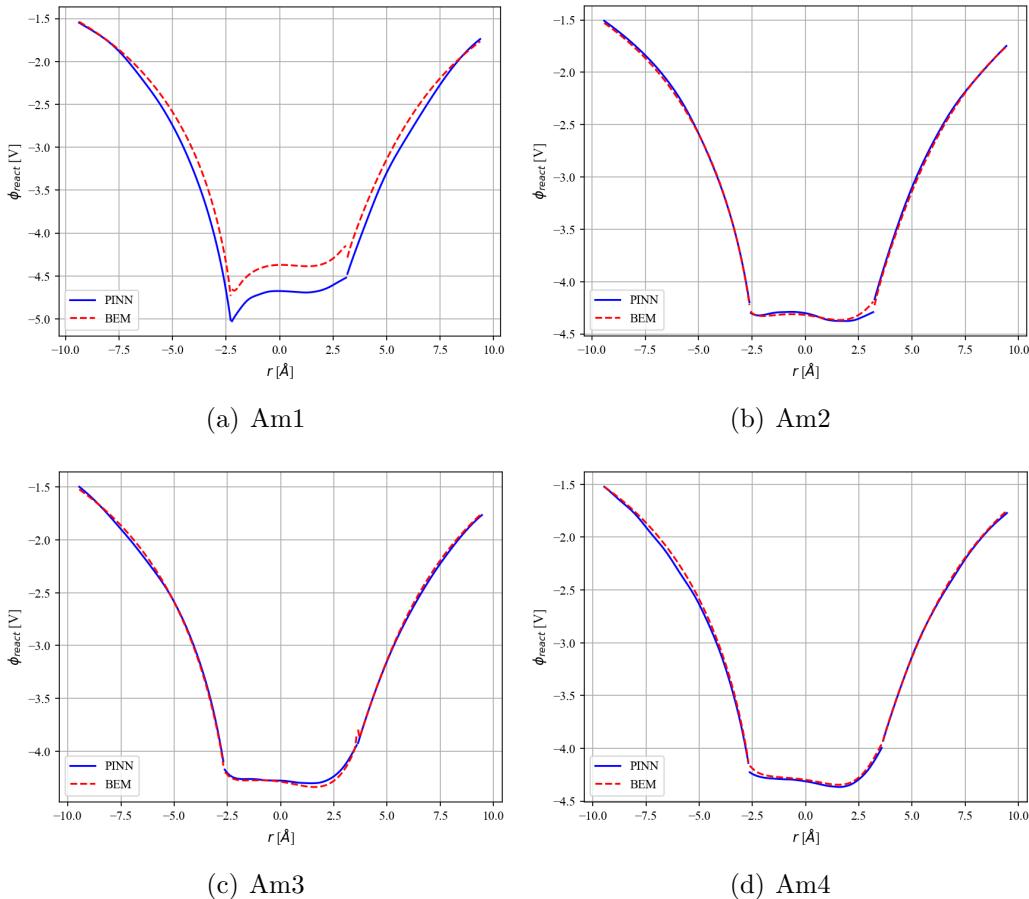


Figura 4.49: Potencial de reacción para la arginina usando distintas mallas, dirección $\theta = \pi/2$, $\phi = \pi/2$

El mismo comportamiento se evidencia en la figura 4.50, en donde se grafica la energía de solvatación. Notar como para las mallas más gruesas, la convergencia de la energía de solvatación tiene un comportamiento más ruidoso y con una mayor desviación. A medida que se refina la malla, la predicción se comienza a ajustar de manera esperada a la obtenida por BEM.

4. Resultados y Análisis

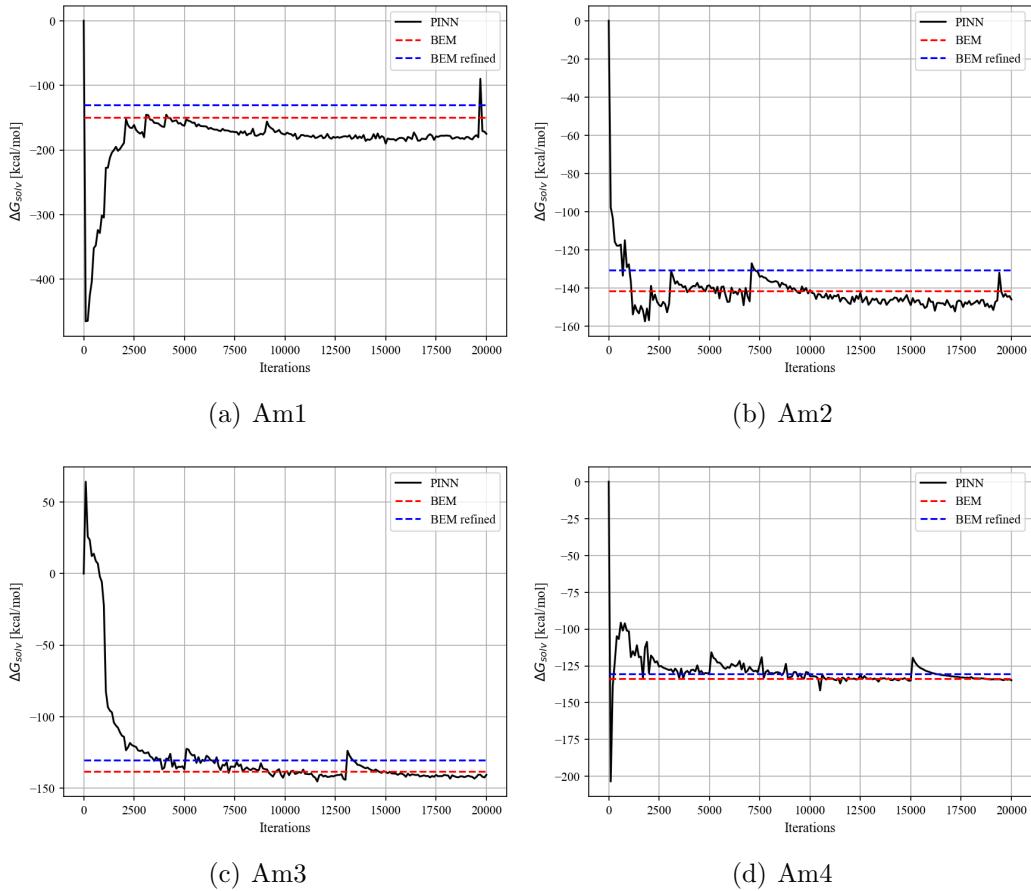


Figura 4.50: Evolución de la energía de solvatación de la arginina para las distintas mallas

Por último, se incluye en la figura 4.51 gráficos del potencial de reacción en la superficie de la molécula. En esta figura se nota claramente que las mallas gruesas no representan correctamente la geometría de la molécula. Sin embargo, se evidencia que los máximos y mínimos ocurren en las mismas zonas, a excepción del caso Am1 en donde no se evidencia el mínimo en la zona superior.

Para complementar el análisis, se utilizará la misma malla superficial que Am4 (la más fina estudiada), pero utilizando menos puntos de colocación en la zona del soluto y solvente. Esto se genera obteniendo una muestra de menor cantidad de elementos de las mallas volumétricas por cada iteración. Los nuevos puntos de colocación se pueden revisar en la tabla 4.29. El propósito de esta nueva prueba es revisar la convergencia del método para distintas cantidades de puntos de colocación en las zonas volumétricas, manteniendo la cantidad de puntos en la superficie.

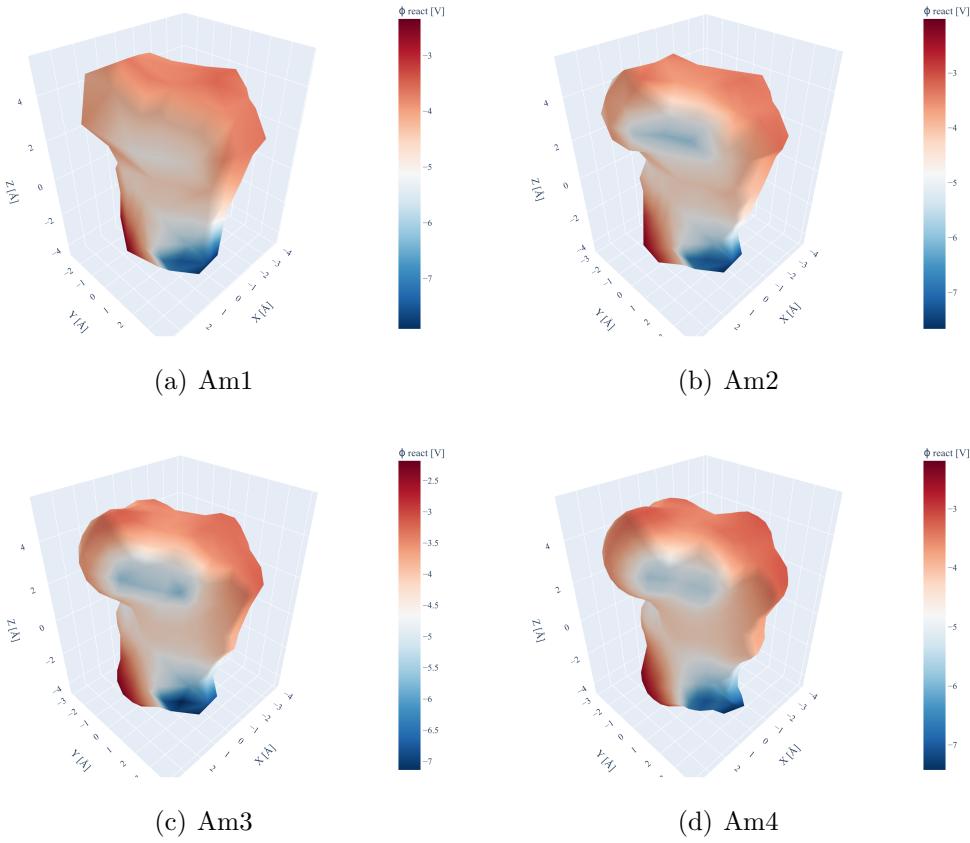


Figura 4.51: Potencial de reacción en la interfaz de la arginina para las distintas mallas

Tabla 4.29: Puntos de colocación utilizados para las nuevas mallas de la arginina

Simulación	Nodos R1	Nodos R2	Nodos I	Nodos D2
Am5	2282	4589	1318	1238
Am6	4494	9090	1318	1238

Además de comparar la cantidad de puntos de colocación en las zonas volumétricas, se mostrarán los resultados en 2 fases del entrenamiento: a 20000 iteraciones como en los otros casos, y a 35000 iteraciones.

Los resultados obtenidos fueron presentados en la tabla 4.30. Revisando los resultados, se nota que la mayoría de los indicadores se ven similares, incluso al compararlos también con el caso Am4. La única excepción es la variación en un orden de magnitud en los losses de entrenamiento y validación.

El gran cambio se visualiza en las gráficas del potencial de reacción en una dirección fijada, figura 4.52. Notar como reducir el número de puntos de colocación en el soluto y solvente empeora la predicción del potencial de reacción en comparación a Am4. Lo anterior se refleja

4. Resultados y Análisis

Tabla 4.30: Resultados para estudio de malla de la arginina

Simulación	Iteraciones	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Am5	20000	-131.952	1.07E-02	2.76E-02	2.82E-04	2.37E-04
Am5	35000	-133.641	2.37E-02	2.48E-02	1.68E-04	1.10E-04
Am6	20000	-134.575	3.08E-02	2.74E-02	1.38E-04	5.33E-05
Am6	35000	-135.220	3.58E-02	2.25E-02	2.39E-05	1.70E-05

con mayor medida en Am5, el cual utiliza menor cantidad de puntos de colocación que Am6. De todas formas, lo interesante ocurre al aumentar el número de iteraciones, ambas soluciones mejoran sustancialmente, convergiendo nuevamente a la predicción de BEM.

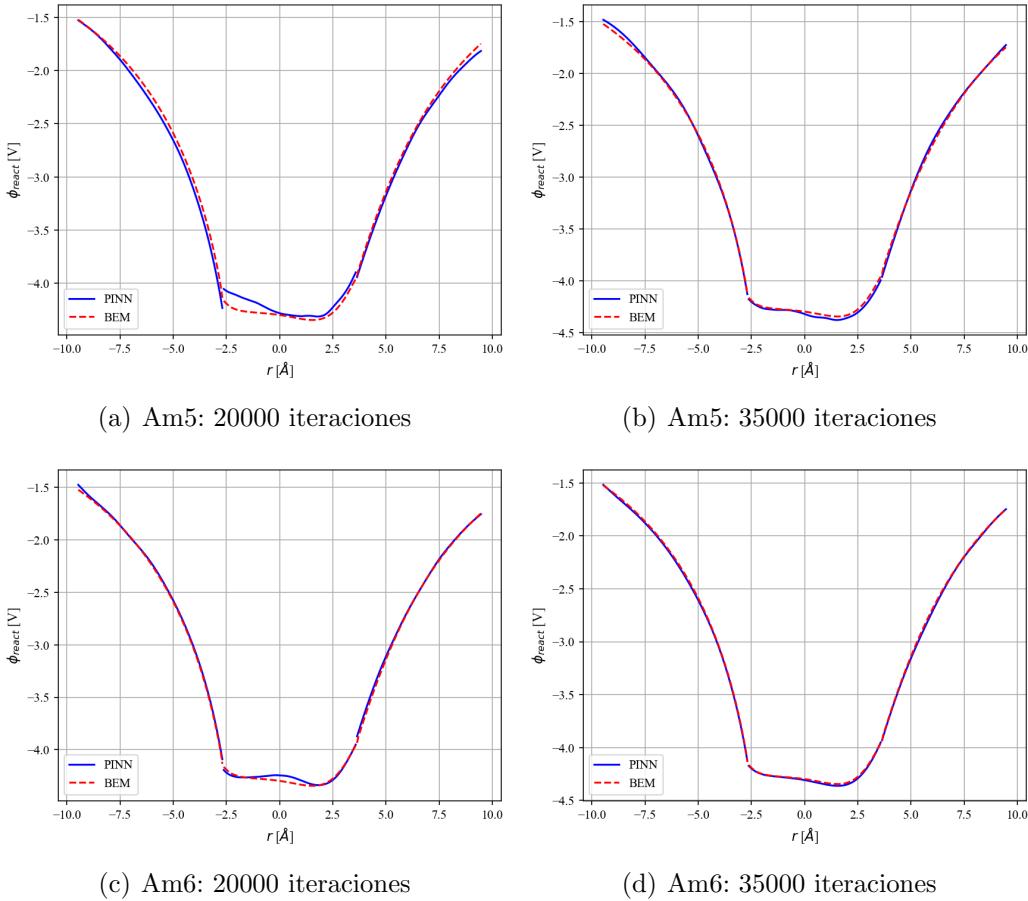


Figura 4.52: Potencial de reacción para la arginina, dirección $\theta = \pi/2$, $\phi = \pi/2$

De aquí se puede desprender que reducir el número de puntos de colocación obliga a aumentar el número de iteraciones para obtener una buena solución. Se estima que es menos costoso en términos computacionales utilizar más iteraciones con menor cantidad de puntos de colocación, que menos iteraciones con mayor cantidad de puntos de colocación. Además, mayor cantidad

de iteraciones con menos puntos en el solvente y/o soluto trae la ventaja de generar mayor evaluaciones en la interfaz de la molécula a lo largo del entrenamiento, ya que esa malla no se modifica.

4.5.2. Uso de Datos Experimentales

Para la arginina, se probó nuevamente el uso de datos experimentales ϕ_{ENS} en el proceso de entrenamiento. Se mostrarán los resultados para las 2 formas de incluir el cálculo ϕ_{ENS} en la función de pérdida, detalladas en la sección 3.2.4. El valor experimental de ϕ_{ENS} fue obtenido usando el software de BEM (PBJ). Debido a que el cálculo de ϕ_{ENS} necesita de un solvente de mayor tamaño, se utiliza una esfera para la frontera de radio $R = 12 \text{ \AA}$.

Tabla 4.31: Puntos de colocación para la arginina, uso de datos experimentales

Nodos R1	Nodos R2	Nodos I	Nodos D2	Nodos E2
12831	29010	4272	658	8667

Los resultados fueron presentados en la tabla 4.32. Rápidamente se puede visualizar que todos los indicadores empeoran al compararse con los resultados obtenidos en la tabla 4.25 (casos sin datos experimentales).

Tabla 4.32: Resultados para el uso de datos experimentales ϕ_{ENS} para la arginina

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Exponencial	-128.03	1.93E-02	3.48E-02	1.67E+00	1.19E-05
Promedio	-105.94	1.88E-01	1.96E-01	1.91E+02	4.90E-05

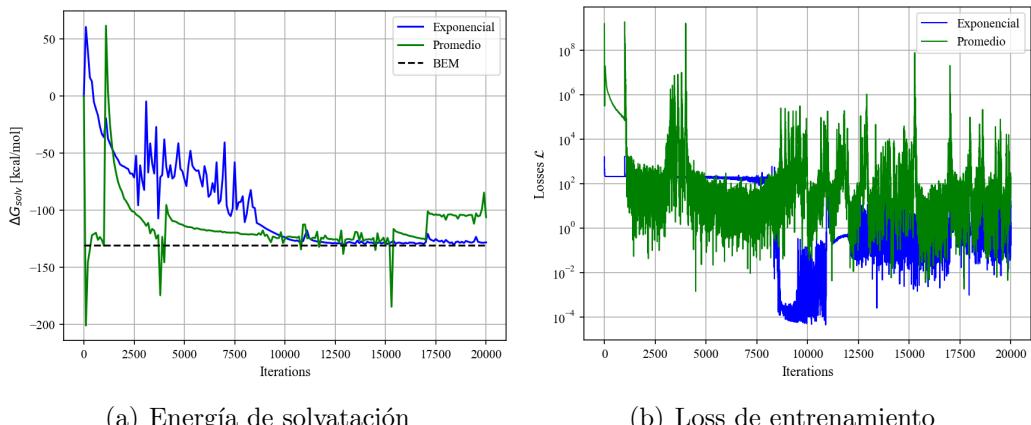


Figura 4.53: Energía de solvatación y loss de entrenamiento para uso de datos experimentales

4. Resultados y Análisis

Revisando la figura 4.53, se puede visualizar como el uso de la forma exponencial de calcular ϕ_{ENS} es totalmente superior a la forma de promedio. Incluso, esta última genera una predicción totalmente errónea de la energía de solvatación. Además, se visualiza que ambos métodos generan un loss de entrenamiento bastante ruidoso y con altos órdenes, similar a lo obtenido para el Ion de Born.

Adicionalmente, revisando la figura 4.54, se puede visualizar que el método exponencial nuevamente es superior al método del promedio para obtener ϕ_{ENS} , al incluirlo en el proceso de entrenamiento. Incluso, el método del promedio es incapaz de replicar el potencial de reacción en la zona del soluto. Sin embargo, el método exponencial empeora la solución en comparación al caso de no utilizarlo. Pese a que pueda replicar correctamente el potencial en el solvente, en el soluto se evidencian claras desviaciones. Esto se puede deber al loss ruidoso que genera añadir este término a la función de pérdida. Se infiere que la adición de este término a la función de pérdida interrumpe la minimización de los residuales de la ecuación de PB.

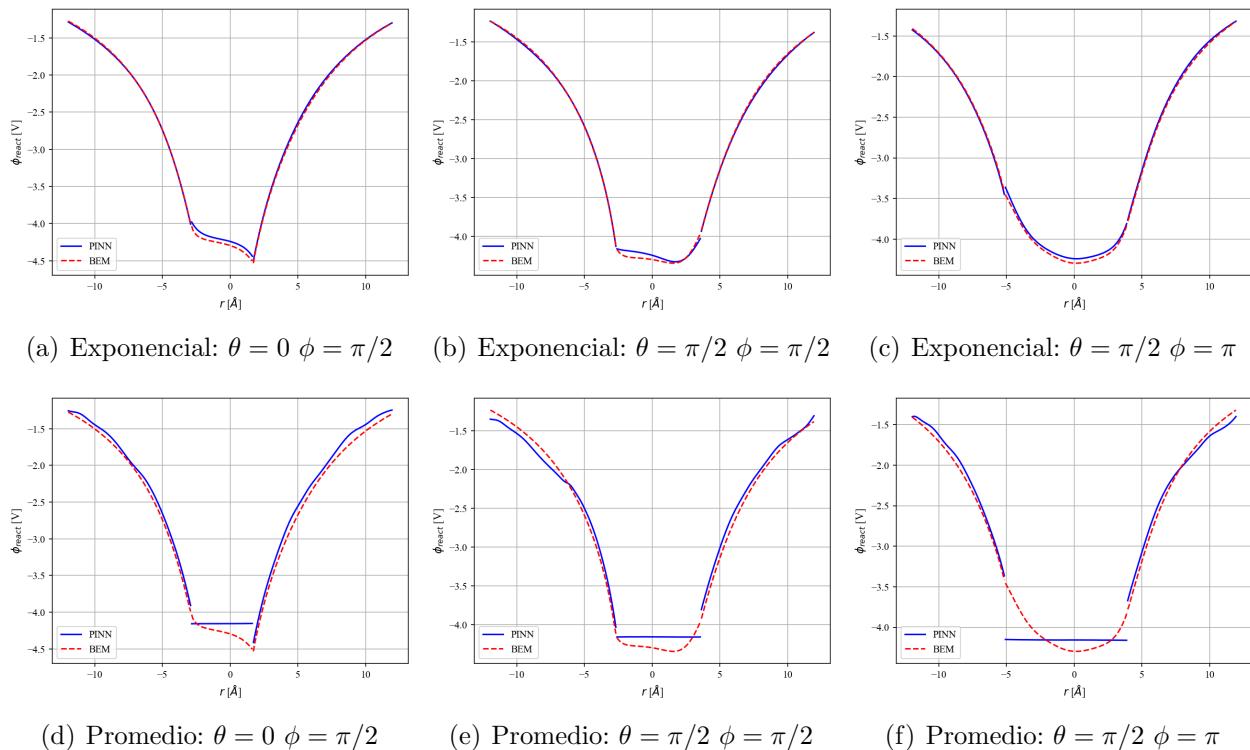


Figura 4.54: Potencial de reacción para la arginina, uso de datos experimentales

4.6. Ubicuitina

La tercera molécula real en la que se probó el método de PINNs para resolver la ecuación de PB es la ubicuitina. Esta molécula posee 1231 cargas en su estructura y tiene una geometría bastante compleja, tal como se visualiza en la figura 4.55.

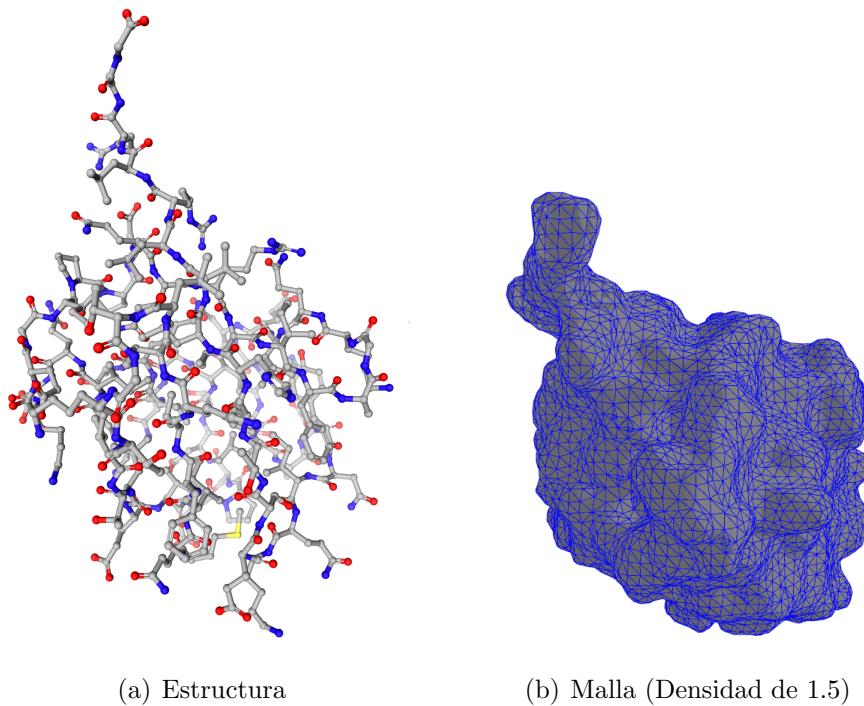


Figura 4.55: Estructura y malla para la ubicuitina

Para las simulaciones de la ubicuitina, se utilizaron las siguientes dimensiones y/o propiedades:

- Constante dieléctrica soluto: $\epsilon_1 = 2 [\epsilon_0]$
- Constante dieléctrica solvente: $\epsilon_2 = 80 [\epsilon_0]$
- Inverso de la longitud de Debye: $\kappa = 0.125 [1/\text{\AA}]$
- Radio esfera frontera solvente: $R = 28.7 [\text{\AA}]$

Para esta molécula se probaron 2 casos, revisando la utilización de la capa de escalamiento del output para la red neuronal. En ambos casos, se utilizaron los puntos de colocación que se presentan en la tabla 4.33. Además, se resolvió el esquema de regularización 2 de la ecuación de PB lineal y se utilizaron 40000 iteraciones para el proceso de entrenamiento. En estos casos se utilizaron 2 redes MLP con capa de escalamiento del input y de Fourier, función de activación entrenable, con 4 capas ocultas y 200 neuronas por capa. Los resultados se compararán con los obtenidos con el software de BEM (PBJ) utilizando la misma malla superficial.

4. Resultados y Análisis

Tabla 4.33: Puntos de colocación utilizados para la ubicuitina

Nodos R1	Nodos R2	Nodos I	Nodos D2
44109	65686	11404	2523

Los resultados se presentan en la tabla 4.34. Notar que el error en el potencial de reacción y los losses disminuyen al utilizar el escalamiento del output, coincidiendo con el estudio realizado para la arginina. Lo interesante ocurre al revisar el error en la energía de solvatación, dado que el caso no escalado alcanza un orden de magnitud menor que el caso escalado.

Tabla 4.34: Resultados para el uso de la capa del escalamiento del output para la ubicuitina

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Sin escalamiento	-585.53	2.08E-02	9.59E-02	4.49E-05	3.53E-05
Con escalamiento	-637.82	1.11E-01	3.99E-02	4.13E-06	3.62E-06

Revisando la figura 4.56a se puede identificar la causa de esa diferencia en el error de la energía de solvatación. Notar que el caso escalado converge a la predicción de BEM utilizando la misma malla superficial, a diferencia del caso no escalado el cual está cercano a la predicción de BEM refinado. Este comportamiento refleja enormemente el efecto y la importancia de escalar la red neuronal en esta aplicación. La evolución de la energía de solvatación para el caso no escalado es más ruidosa, y todavía no se ha estabilizado, lo que implica una convergencia bastante lenta, y por ende, podría llegar a confundir la aproximación hacia la solución de BEM refinado¹⁰.

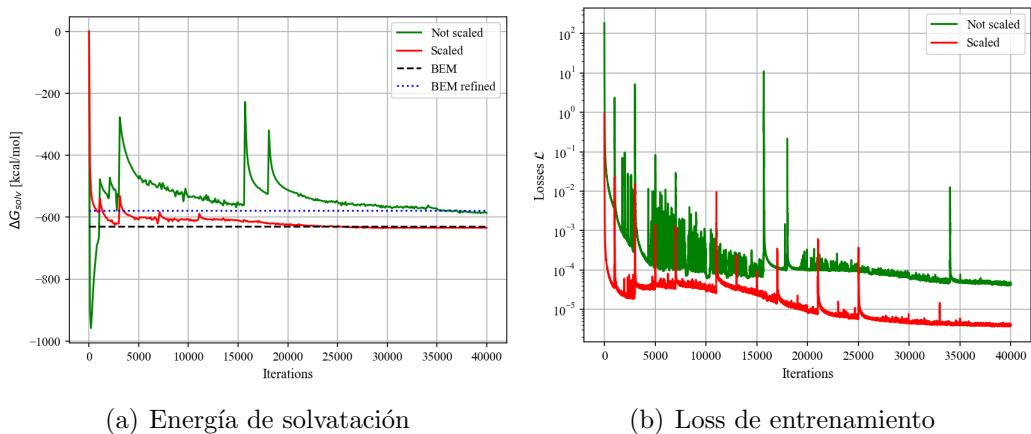


Figura 4.56: Energía de solvatación y loss de entrenamiento para los casos de la ubicuitina

¹⁰Recordar que el error en la energía de solvatación se calcula respecto a la solución de BEM refinado (aproximación de la solución real)

Por otra parte, se nota que la evolución en la energía de solvatación para el caso escalado es más suave y rápida, alcanzando la predicción de BEM con la misma malla superficial cerca de las 22000 iteraciones.

Además de la evolución de la energía de solvatación, es de relevancia revisar la evolución del loss de entrenamiento a medida que aumentan las iteraciones. En la figura 4.56b se puede visualizar como el caso escalado alcanza losses menores en todo el proceso de minimización. Adicionalmente, tiene un comportamiento no tan ruidoso, a diferencia del caso no escalado, lo cual puede ser beneficioso para replicar correctamente la solución de la ecuación.

En la figura 4.57 se puede revisar el potencial de reacción en 3 direcciones distintas. Notar que ambos casos tienen una desviación en el soluto, respecto a la predicción de BEM. Se estima que con un mayor número de iteraciones, estas desviaciones deberían ser minimizadas. Además, se nota que el caso escalado genera desviaciones menores, y una mejor aproximación cerca de la interfaz de la molécula.

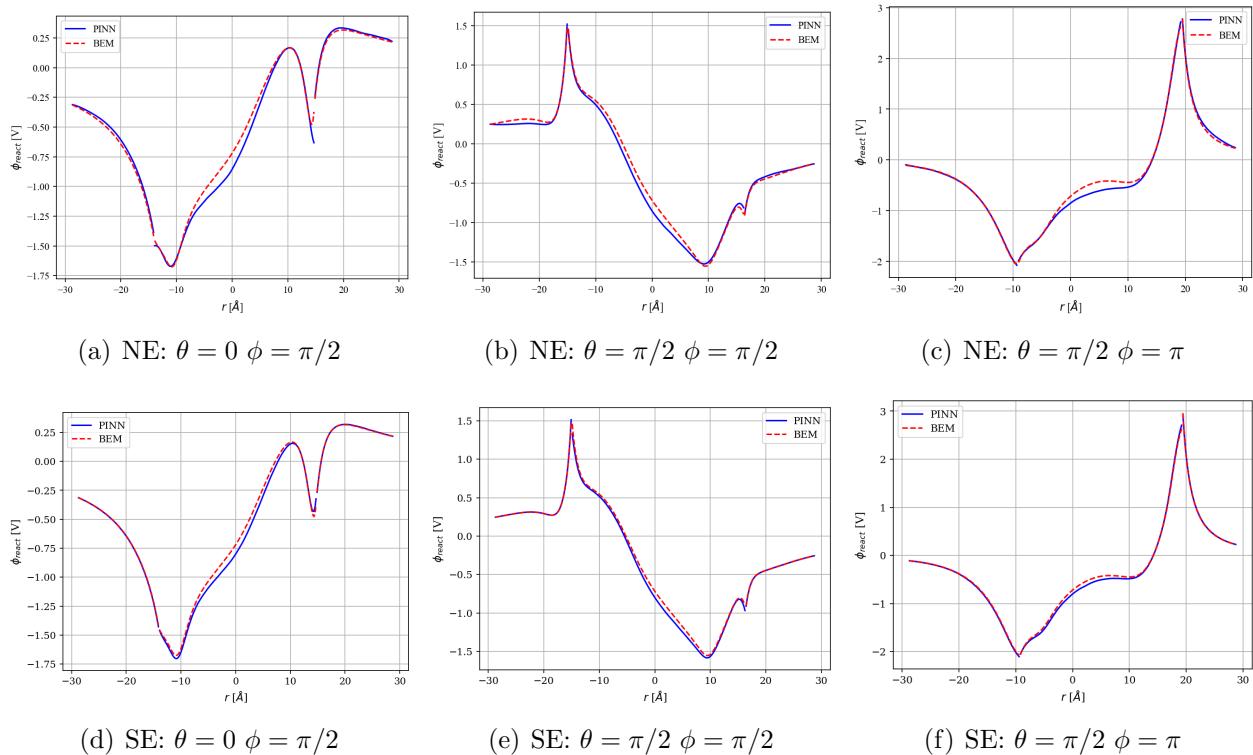


Figura 4.57: Potencial de reacción para la ubicuitina¹¹, dirección $\theta = 0$, $\phi = \pi/2$

¹¹NE y SE hacen referencia al caso no escalado y al caso escalado respectivamente.

4. Resultados y Análisis

Asimismo, en la figura 4.58 se puede revisar el potencial de reacción en la superficie de la ubicuitina. Ambos métodos tienen los máximos y mínimos en las mismas zonas, coincidiendo con la predicción de BEM. Sin embargo, se nota como el caso escalado tiende a las magnitudes correctas en los máximos y mínimos.

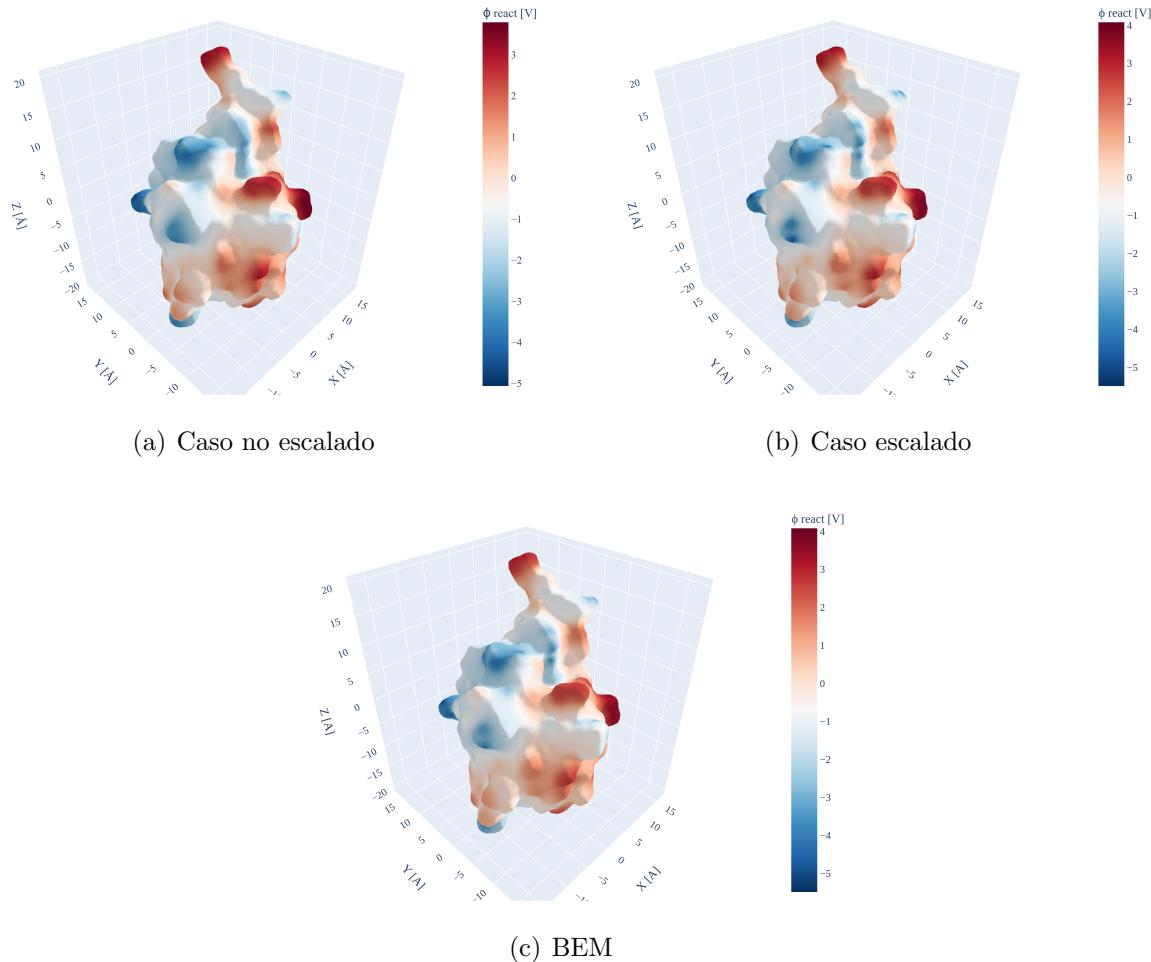


Figura 4.58: Potencial de reacción en la interfaz de la ubicuitina

Por último, en las figuras 4.59 y 4.60 se puede revisar el error absoluto y relativo en la predicción del potencial de reacción en la interfaz de la molécula. De estas figuras se desprende que el caso no escalado tiene errores mayores que el caso escalado. Además, tiene más zonas de alto error.

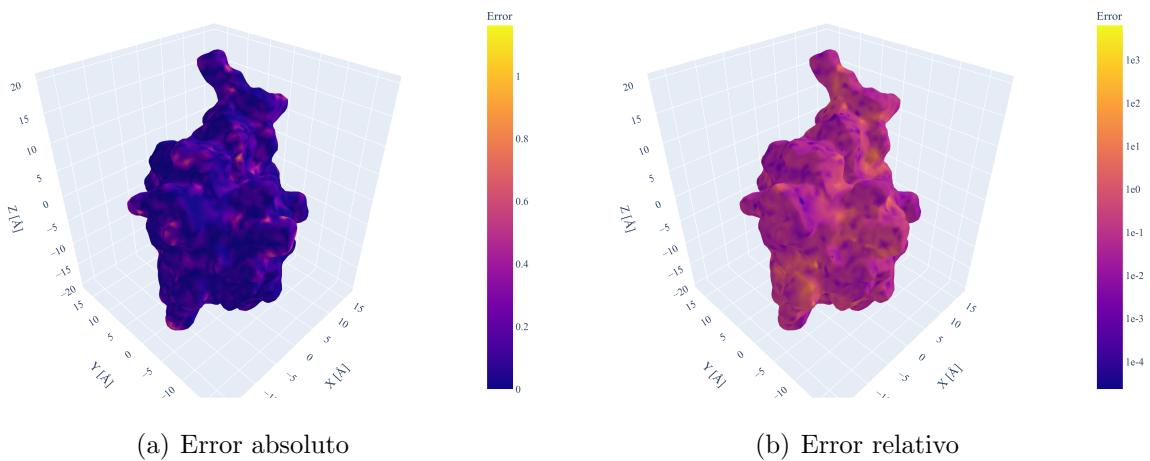


Figura 4.59: Error absoluto y relativo en la interfaz de la ubicuitina, caso no escalado

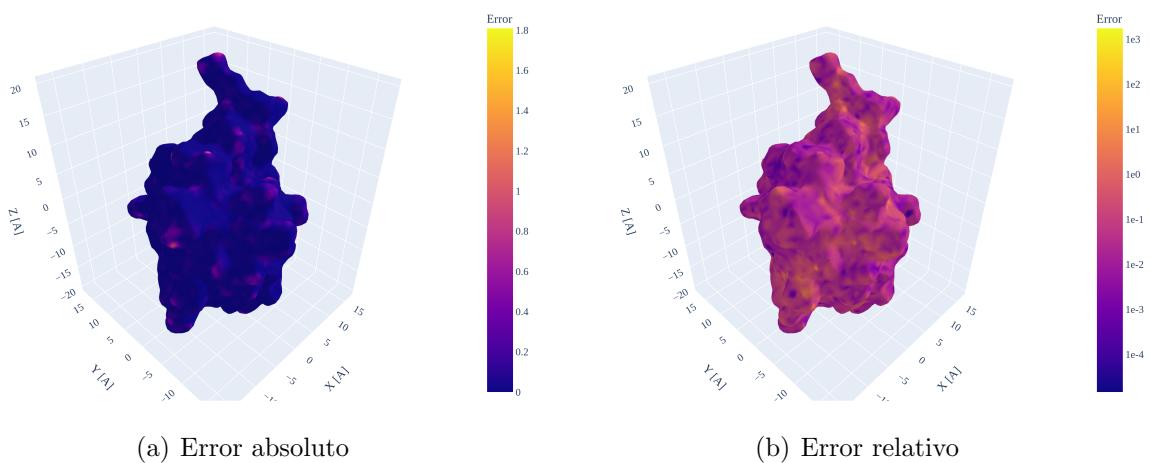


Figura 4.60: Error absoluto y relativo en la interfaz de la ubicuitina, caso escalado

4.7. ADN

La última molécula real en la que se probó el método de PINNs para resolver la ecuación de PB es una molécula de ADN, extraída del complejo Homeodominio Antennapedia. Esta molécula de ADN posee 951 cargas en su estructura, y tiene una geometría helicoidal, tal como se visualiza en la figura 4.61.

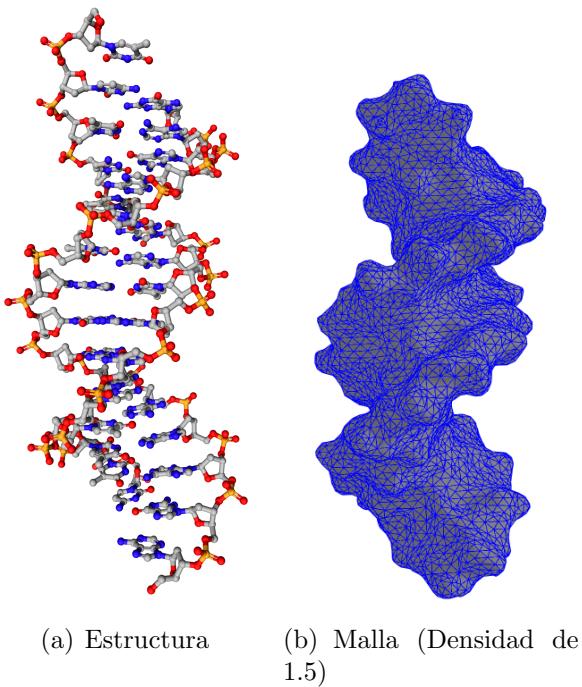


Figura 4.61: Estructura y malla para el ADN

Para las simulaciones del ADN, se utilizaron las siguientes dimensiones y/o propiedades:

- Constante dieléctrica soluto: $\epsilon_1 = 2 [\epsilon_0]$
- Constante dieléctrica solvente: $\epsilon_2 = 80 [\epsilon_0]$
- Inverso de la longitud de Debye: $\kappa = 0.125 [1/\text{\AA}]$
- Radio esfera frontera solvente: $R = 33.2 [\text{\AA}]$

La única prueba que se realizó con esta molécula fue de verificar que el método de PINNs es capaz de resolver la forma no lineal de la ecuación de Poisson Boltzmann, para revisar si puede reflejar las diferencias que existen con la solución de la ecuación lineal. Para esto, se resolvió el esquema de regularización 2 de la ecuación de PB lineal y no lineal, y se utilizaron 40000 iteraciones para el proceso de entrenamiento. El resultado para el caso lineal fue comparado con el software de BEM (PBJ) utilizando la misma malla superficial, y el caso no lineal con el

software de FDM (APBS). En estos casos se utilizaron 2 redes MLP con capas de escalamiento y de Fourier, función de activación entrenable, con 4 capas ocultas y 200 neuronas por capa.

Los puntos de colocación utilizados se pueden revisar en la tabla 4.35.

Tabla 4.35: Puntos de colocación utilizados para el ADN

Nodos R1	Nodos R2	Nodos I	Nodos D2
36515	65877	10580	3120

Los resultados obtenidos pueden ser revisados en la tabla 4.36. Notar que el caso lineal genera errores del orden de 10^{-2} para la predicción de la energía de solvatación y el potencial en la superficie. Sin embargo, en la figura 4.62 se puede visualizar que el método está convergiendo a la solución de BEM utilizando la misma malla. Esto puede dar indicios de que se necesitó una malla más fina para representar la geometría molecular.

Tabla 4.36: Resultados de ambos casos para el ADN¹²

Simulación	ΔG_{solv}^θ [kcal/mol]	Error rel. $\mathcal{E}_{G_{solv}}$	Error L_2 \mathcal{E}_ψ	Loss \mathcal{L} entrenamiento	Loss \mathcal{L}^v validación
Caso Lineal	-4763.17	2.15E-02	1.29E-02	7.70E-05	6.73E-05
Caso No Lineal	-4711.59	1.90E-03	2.10E-02	1.33E-04	1.12E-04

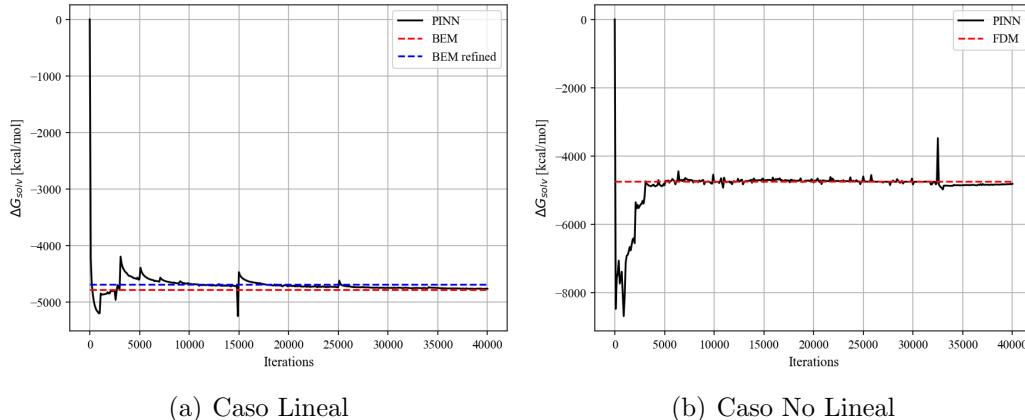


Figura 4.62: Energía de solvatación (componente lineal) del ADN para casos lineal y no lineal

Para la comparación del caso no lineal, se tuvo que utilizar el software de FDM (APBS). Este software no necesita una malla superficial debido al método que emplea en la resolución. Por esta razón, no se tiene una comparación adecuada entre el método de PINNs y FDM. Sin embargo,

¹²Para la energía de solvatación solo se consideró el componente lineal de la ecuación 2.45, debido a predicciones erróneas del componente no lineal producto de oscilaciones en el potencial.

4. Resultados y Análisis

en la figura 4.62 se puede visualizar que la predicción para el caso no lineal se aproxima a la obtenida con FDM.

En la figura 4.63 se puede revisar la predicción del potencial de reacción en 2 direcciones para el caso lineal. En estas gráficas se nota que la solución de PINNs se ajusta a la predicción de BEM para la molécula de ADN.

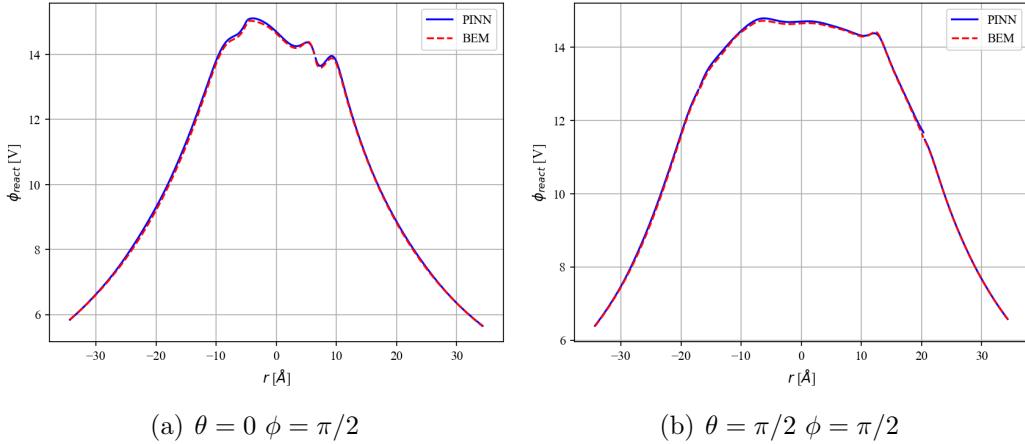


Figura 4.63: Potencial de reacción para el ADN, caso lineal

Asimismo, el potencial de reacción para el caso no lineal puede ser revisado en la figura 4.64. En estas gráficas se nota que resolver la ecuación no lineal de PB por medio de PINNs genera oscilaciones en la predicción del potencial en la zona del solvente. Además, este método no logra replicar la curvatura que existe en la solución en la zona del soluto.

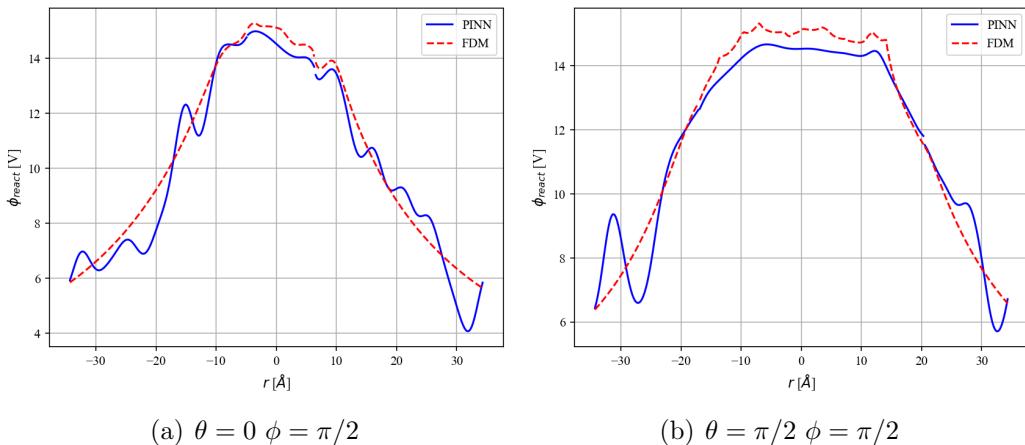


Figura 4.64: Potencial de reacción para el ADN, caso no lineal

El fenómeno visualizado para el caso no lineal se puede deber a las siguientes razones:

- Para el caso no lineal no se pudo agregar la función seno hiperbólico en la función de pérdida de la red neuronal, la cual aparece en el residual del solvente. Esto se debe a que en iteraciones tempranas, se tiene una predicción totalmente errónea del potencial en el solvente, el cual tiende a infinito al aplicarle el seno hiperbólico. Para solucionar este problema, se truncó la aproximación por series de Taylor, lo que puede inducir errores.
- Además, se utilizaron condiciones de borde lineales, dadas por la función de Green de Yukawa. Pese a que esta solución puede ser una buena aproximación lejos de la molécula, puede inducir errores en la predicción del potencial, intentando forzar la solución lineal. A futuro, se podría probar obtener una condición de borde similar a la de Yukawa, en donde se consideran las cargas inmersas en el solvente, pero usando la ecuación no lineal. Incluso, se podría utilizar otra red neuronal para resolver este caso y generar una aproximación para la condición de borde.
- La adimensionalización utilizada para el caso no lineal, planteada en la sección 3.5.3, puede generar un problema mal condicionado producto del factor $\beta^* \approx 10^{-4}$. Notar que este factor no afecta a la formulación lineal debido a que se simplifica.

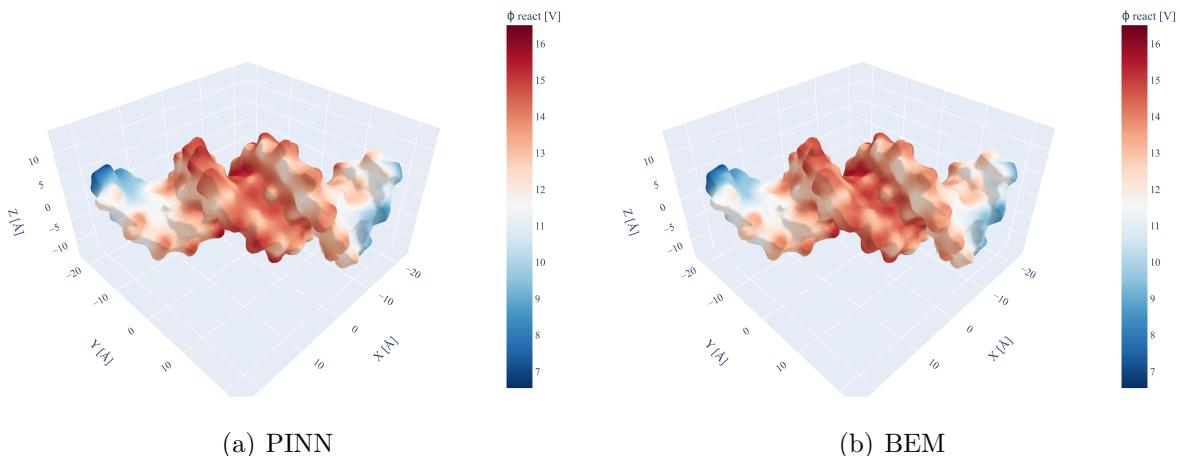


Figura 4.65: Potencial de reacción en la interfaz del ADN, caso lineal

Pese a las diferencias y oscilaciones presentes en el caso no lineal, los gráficos del potencial de reacción en la interfaz (figura 4.66) se ven bastante similares. Los máximos y mínimos ocurren en las mismas zonas.

Por último, en las figuras 4.67 y 4.68 se puede visualizar el error absoluto y relativo en la predicción del potencial de reacción, respecto a sus soluciones de referencia en cada caso. Notar que el caso no lineal genera mayores errores, con máximos en los centros y extremos de la helicoide.

4. Resultados y Análisis

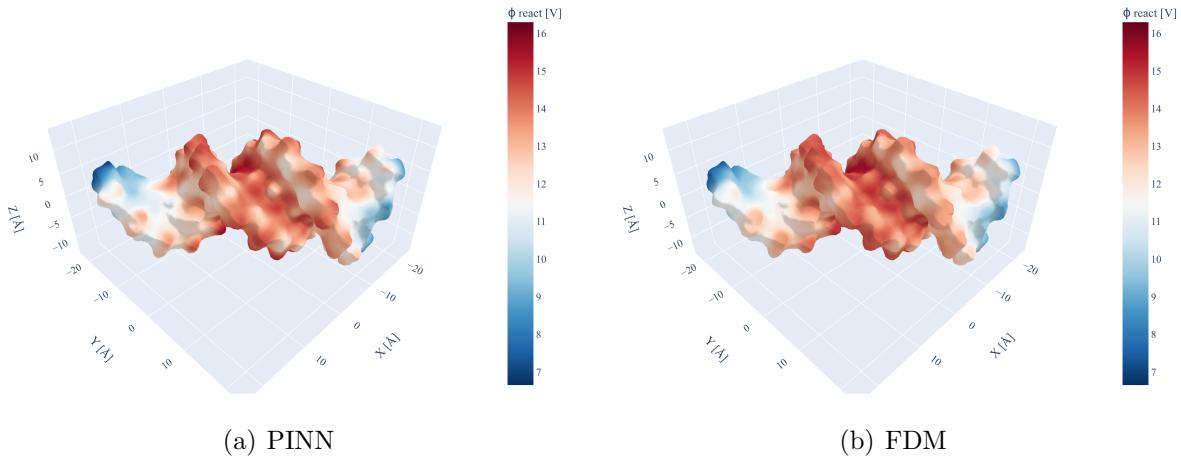


Figura 4.66: Potencial de reacción en la interfaz del ADN, caso no lineal

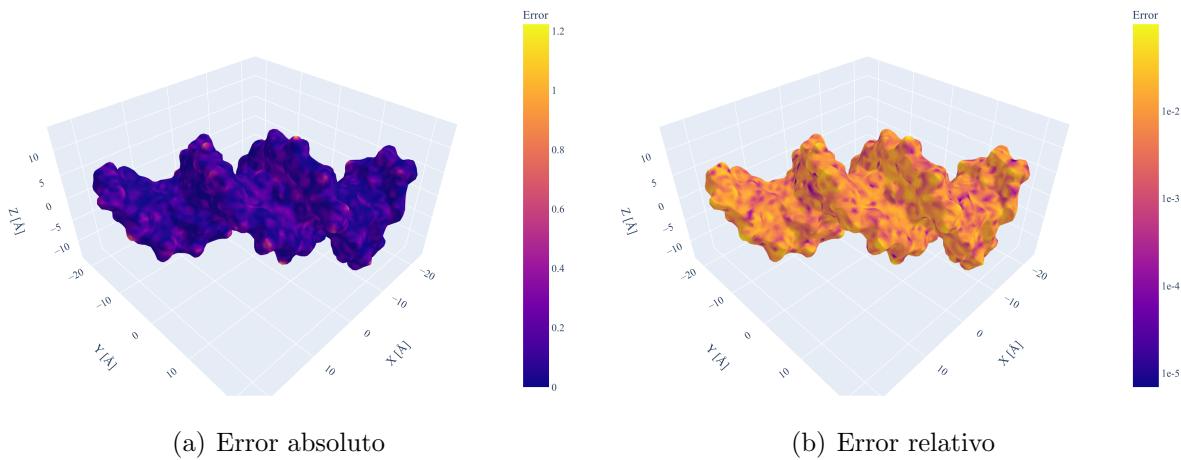


Figura 4.67: Error absoluto y relativo en la interfaz del ADN, caso lineal

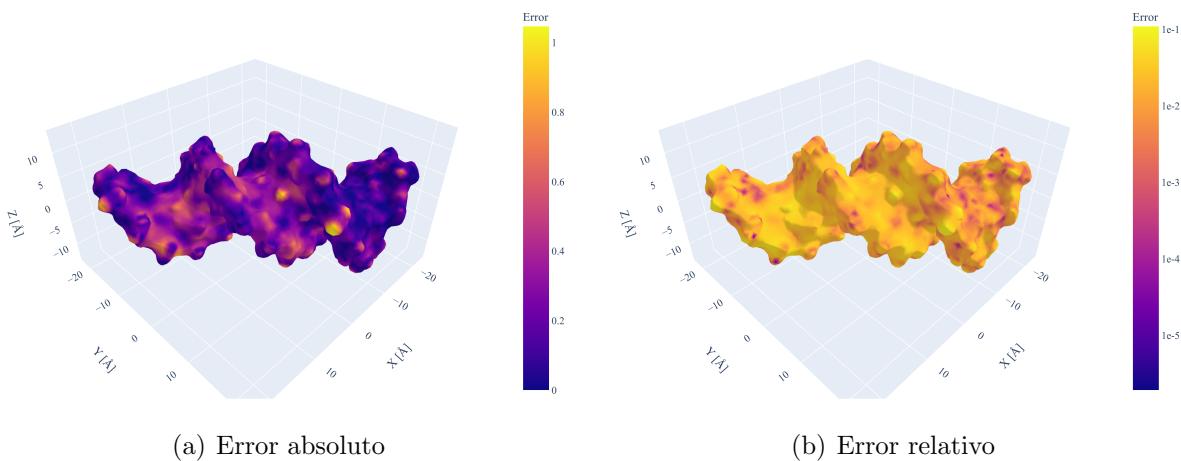


Figura 4.68: Error absoluto y relativo en la interfaz del ADN, caso no lineal

Capítulo 5

Conclusiones

En el presente trabajo se logró implementar y aplicar el método de PINNs para resolver la ecuación de Poisson-Boltzmann. La implementación detallada en este estudio permitió utilizar PINNs para resolver escenarios con distintas moléculas, partiendo desde el caso canónico dado por el Ion de Born, hasta alcanzar moléculas más complejas con más de 1000 cargas como la ubicuitina y el ADN. Todos los casos se compararon con las soluciones de referencia disponibles (soluciones analíticas, códigos de BEM o FDM). Para las distintas moléculas, se lograron errores de órdenes entre 10^{-2} y 10^{-3} tanto para el error relativo en la energía de solvatación, como para el error L_2 del potencial de reacción en la interfaz.

El análisis realizado sugiere que utilizar dos redes neuronales, una por subdominio, es ideal para resolver esta ecuación. Además, es esencial regularizar la ecuación en el soluto para evitar singularidades debido a las cargas puntuales, utilizar un muestreo aleatorio para los puntos de colocación y un algoritmo para ponderar los distintos términos de la función de pérdida. Se demostró que una arquitectura simple como el MLP puede replicar eficazmente la solución para el potencial en diversos escenarios, aunque es indispensable añadir capas adicionales, como las de escalamiento (tanto para el input como para el output) y la capa de Fourier.

A futuro, se debería realizar un análisis detallado de posibles mejoras a la arquitectura, incluyendo la evaluación de nuevas redes como las KANN [87], que presentan cambios significativos respecto a las arquitecturas basadas en MLP. También se recomienda estudiar técnicas de descomposición de dominios con PIELM [50] para evaluar si mejora la convergencia en la solución por parte de PINNs.

Respecto a lo obtenido, un hallazgo importante es que la energía de solvatación es un indicador bastante robusto, ya que PINNs puede predecir con precisión este valor incluso en iteraciones

5. Conclusiones

tempranas, aún cuando la forma del potencial no se haya estabilizado. Este hecho abre la posibilidad de utilizar PINNs para la predicción de esta energía en aplicaciones de biofísica.

Se evidenció que una mayor cantidad de puntos de colocación mejora la convergencia de PINNs, y que la cantidad de puntos en la interfaz es fundamental, ya que determina la geometría molecular. En los casos estudiados, la solución de PINNs coincidió con la solución obtenida por BEM utilizando la misma malla superficial, lo que convierte a PINNs en un método dependiente de la malla superficial de la molécula (al igual que BEM).

Uno de los desafíos por resolver es la resolución de la forma no lineal de la ecuación de Poisson-Boltzmann con PINNs. Es necesario revisar la condición de borde, la adimensionalización de la ecuación, y la inclusión del seno hiperbólico en la función de pérdida. Resolver esta forma no lineal significaría un gran avance en la electrostática molecular.

Además, se debe trabajar en la adición de nuevos términos para la función de pérdida. En este estudio, la utilización de datos experimentales como el *effective near surface potential* no mejoró la convergencia del método. Este aspecto necesita ser investigado en profundidad, ya que logra diferenciar el método PINNs de los métodos numéricos tradicionales en esta aplicación. También se debería considerar la inclusión de términos relacionados con modelaciones de dinámica molecular y/o información de la interfaz, aspectos que se pierden al considerar el modelo continuo.

Este trabajo representa un primer paso en la modelación de la electrostática molecular utilizando redes neuronales, demostrando la viabilidad y efectividad de PINNs en esta aplicación. Todo el código elaborado fue implementado en una librería llamada XPPBE, publicada en GitHub [9], permitiendo a la comunidad científica utilizarla para resolver la ecuación de Poisson-Boltzmann para distintas moléculas y escenarios. Esto abre la investigación a la contribución de terceros, fomentando el avance en este campo.

Bibliografía

- [1] B. Roux and T. Simonson, “Implicit solvent models,” *Biophysical chemistry*, vol. 78, no. 1-2, pp. 1–20, 1999.
- [2] N. A. Baker, “Poisson–Boltzmann methods for biomolecular electrostatics,” in *Methods in enzymology*. Elsevier, 2004, vol. 383, pp. 94–118.
- [3] F. Fogolari, A. Brigo, and H. Molinari, “The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology,” *Journal of Molecular Recognition*, vol. 15, no. 6, pp. 377–392, 2002.
- [4] C. D. Cooper Villagran, “Biomolecular electrostatics with continuum models: a boundary integral implementation and applications to biosensors,” Ph.D. dissertation, Boston University, 2015.
- [5] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [6] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.
- [7] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.
- [8] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, “Scientific machine learning through physics-informed neural networks: Where we are and what’s next,” *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022.
- [9] M. Achondo, “XPPBE: PINNs for PBE,” <https://github.com/MartinAchondo/XPPBE>, 2023.
- [10] J. N. Reddy, *An introduction to continuum mechanics*. Cambridge university press, 2013.

- [11] J. D. Jackson, *Classical electrodynamics*. John Wiley & Sons, 2012.
- [12] M. J. Holst *et al.*, “The Poisson-Boltzmann equation: Analysis and multilevel numerical solution,” *Applied Mathematics and CRPC, California Institute of Technology*, 1994.
- [13] F. Reif, “Fundamentals of statistical and thermal physics,” 1998.
- [14] W. C. Poon and D. Andelman, *Soft condensed matter physics in molecular and cell biology*. CRC Press, 2006.
- [15] G. Tomasello, I. Armenia, and G. Molla, “The Protein Imager: a full-featured online molecular viewer interface with server-side HQ-rendering capabilities,” *Bioinformatics*, vol. 36, no. 9, pp. 2909–2911, 2020.
- [16] A. Lee, W. Geng, and S. Zhao, “Regularization methods for the Poisson-Boltzmann equation: comparison and accuracy recovery,” *Journal of Computational Physics*, vol. 426, p. 109958, 2021.
- [17] M. Holst, J. A. Mccammon, Z. Yu, Y. Zhou, and Y. Zhu, “Adaptive finite element modeling techniques for the poisson-boltzmann equation,” *Communications in computational physics*, vol. 11, no. 1, pp. 179–214, 2012.
- [18] B. J. Yoon and A. Lenhoff, “A boundary element method for molecular electrostatics with electrolyte effects,” *Journal of Computational Chemistry*, vol. 11, no. 9, pp. 1080–1086, 1990.
- [19] C. D. Cooper, “A Boundary-Integral Approach for the Poisson–Boltzmann Equation with Polarizable Force Fields,” *Journal of Computational Chemistry*, vol. 40, no. 18, pp. 1680–1692, 2019.
- [20] C. D. Cooper and J. P. Bardhan, “A Simple Electrostatic Model for the Hard-Sphere Solute Component of Nonpolar Solvation,” *arXiv preprint arXiv:2005.13019*, 2020.
- [21] J. Che, J. Dzubiella, B. Li, and J. A. McCammon, “Electrostatic free energy and its variations in implicit solvent models,” *The Journal of Physical Chemistry B*, vol. 112, no. 10, pp. 3058–3069, 2008.
- [22] B. Yu, C. C. Pletka, B. M. Pettitt, and J. Iwahara, “De novo determination of near-surface electrostatic potentials by NMR,” *Proceedings of the National Academy of Sciences*, vol. 118, no. 25, p. e2104020118, 2021.
- [23] J. G. Kirkwood, “Theory of solutions of molecules containing widely separated charges

- with special application to zwitterions,” *The Journal of Chemical Physics*, vol. 2, no. 7, pp. 351–361, 1934.
- [24] W. Rocchia, “Poisson-Boltzmann equation boundary conditions for biological applications,” *Mathematical and computer modelling*, vol. 41, no. 10, pp. 1109–1118, 2005.
- [25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [26] A. L. Caterini and D. E. Chang, *Deep neural networks in a mathematical framework*. Springer, 2018.
- [27] K. Hornik, M. Stinchcombe, and H. White, “Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks,” *Neural networks*, vol. 3, no. 5, pp. 551–560, 1990.
- [28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [29] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, “On optimization methods for deep learning,” in *Proceedings of the 28th international conference on international conference on machine learning*, 2011, pp. 265–272.
- [30] P. K. Diederik, “Adam: A method for stochastic optimization,” (*No Title*), 2014.
- [31] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.
- [32] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning: a survey,” *Journal of machine learning research*, vol. 18, no. 153, pp. 1–43, 2018.
- [33] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, “An Expert’s Guide to Training Physics-informed Neural Networks,” *arXiv preprint arXiv:2308.08468*, 2023.
- [34] J. Blechschmidt and O. G. Ernst, “Three ways to solve partial differential equations with neural networks—A review,” *GAMM-Mitteilungen*, vol. 44, no. 2, p. e202100006, 2021.
- [35] Y. Sun, Q. Sun, and K. Qin, “Physics-Based Deep Learning for Flow Problems,” *Energies*, vol. 14, no. 22, p. 7760, 2021.
- [36] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, “Physics-informed neural networks (PINNs) for fluid mechanics: A review,” *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–

1738, 2021.

- [37] D. W. Abueidda, Q. Lu, and S. Koric, “Meshless physics-informed deep learning method for three-dimensional solid mechanics,” *International Journal for Numerical Methods in Engineering*, vol. 122, no. 23, pp. 7182–7201, 2021.
- [38] L. Cheng, E. A. Illarramendi, G. Bogopolsky, M. Bauerheim, and B. Cuenot, “Using neural networks to solve the 2D Poisson equation for electric field computation in plasma fluid simulations,” *arXiv preprint arXiv:2109.13076*, 2021.
- [39] Y. Shin, J. Darbon, and G. E. Karniadakis, “On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs,” *arXiv preprint arXiv:2004.01806*, 2020.
- [40] S. Wu, A. Zhu, Y. Tang, and B. Lu, “Convergence of physics-informed neural networks applied to linear second-order elliptic interface problems,” *arXiv preprint arXiv:2203.03407*, 2022.
- [41] T. De Ryck, A. D. Jagtap, and S. Mishra, “Error estimates for physics-informed neural networks approximating the Navier–Stokes equations,” *IMA Journal of Numerical Analysis*, vol. 44, no. 1, pp. 83–119, 2024.
- [42] S. Wang, X. Yu, and P. Perdikaris, “When and why PINNs fail to train: A neural tangent kernel perspective,” *Journal of Computational Physics*, vol. 449, p. 110768, 2022.
- [43] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, “Characterizing possible failure modes in physics-informed neural networks,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 548–26 560, 2021.
- [44] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*. Springer Science & Business Media, 2006, vol. 37.
- [45] K. F. Riley, M. P. Hobson, and S. J. Bence, “Mathematical methods for physics and engineering,” 1999.
- [46] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, “Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems,” *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020.
- [47] A. D. Jagtap and G. E. Karniadakis, “Extended Physics-informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition based Deep Learning Fra-

- mework for Nonlinear Partial Differential Equations.” in *AAAI spring symposium: MLPS*, vol. 10, 2021.
- [48] Z. Hu, A. D. Jagtap, G. E. Karniadakis, and K. Kawaguchi, “Augmented Physics-Informed Neural Networks (APINNs): A gating network-based soft domain decomposition methodology,” *Engineering Applications of Artificial Intelligence*, vol. 126, p. 107183, 2023.
- [49] V. Dwivedi, N. Parashar, and B. Srinivasan, “Distributed physics informed neural network for data-efficient solution to partial differential equations,” *arXiv preprint arXiv:1907.08967*, 2019.
- [50] V. Dwivedi and B. Srinivasan, “Physics informed extreme learning machine (pielm)—a rapid method for the numerical solution of partial differential equations,” *Neurocomputing*, vol. 391, pp. 96–118, 2020.
- [51] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, “Variational physics-informed neural networks for solving partial differential equations,” *arXiv preprint arXiv:1912.00873*, 2019.
- [52] B. Yu *et al.*, “The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems,” *Communications in Mathematics and Statistics*, vol. 6, no. 1, pp. 1–12, 2018.
- [53] J. Sun, Y. Liu, Y. Wang, Z. Yao, and X. Zheng, “BINN: A deep learning approach for computational mechanics problems based on boundary integral equations,” *Computer Methods in Applied Mechanics and Engineering*, vol. 410, p. 116012, 2023.
- [54] G. Lin, P. Hu, F. Chen, X. Chen, J. Chen, J. Wang, and Z. Shi, “BINet: learning to solve partial differential equations with boundary integral networks,” *arXiv preprint arXiv:2110.00352*, 2021.
- [55] L. Lu, P. Jin, and G. E. Karniadakis, “Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators,” *arXiv preprint arXiv:1910.03193*, 2019.
- [56] S. Li and X. Feng, “Dynamic Weight Strategy of Physics-Informed Neural Networks for the 2D Navier–Stokes Equations,” *Entropy*, vol. 24, no. 9, p. 1254, 2022.
- [57] W. Li, X. Xiang, and Y. Xu, “Deep domain decomposition method: Elliptic problems,” in *Mathematical and Scientific Machine Learning*. PMLR, 2020, pp. 269–286.
- [58] X. Jiang, Z. Wang, W. Bao, and Y. Xu, “Generalization of PINNs for elliptic interface problems,” *Applied Mathematics Letters*, p. 109175, 2024.

- [59] C. He, X. Hu, and L. Mu, “A mesh-free method using piecewise deep neural network for elliptic interface problems,” *Journal of Computational and Applied Mathematics*, vol. 412, p. 114358, 2022.
- [60] S. Wu and B. Lu, “INN: Interfaced neural networks as an accessible meshless approach for solving interface PDE problems,” *Journal of Computational Physics*, vol. 470, p. 111588, 2022.
- [61] J. Ying, J. Liu, J. Chen, S. Cao, M. Hou, and Y. Chen, “Multi-scale fusion network: A new deep learning structure for elliptic interface problems,” *Applied Mathematical Modelling*, vol. 114, pp. 252–269, 2023.
- [62] Y.-H. Tseng, T.-S. Lin, W.-F. Hu, and M.-C. Lai, “A cusp-capturing PINN for elliptic interface problems,” *Journal of Computational Physics*, vol. 491, p. 112359, 2023.
- [63] Z. Liu, W. Cai, and Z.-Q. J. Xu, “Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains,” *arXiv preprint arXiv:2007.11207*, 2020.
- [64] S. Wu, A. Zhu, Y. Tang, and B. Lu, “Solving parametric elliptic interface problems via interfaced operator network,” *arXiv preprint arXiv:2308.14537*, 2023.
- [65] X. Huang, H. Liu, B. Shi, Z. Wang, K. Yang, Y. Li, M. Wang, H. Chu, J. Zhou, F. Yu *et al.*, “A Universal PINNs Method for Solving Partial Differential Equations with a Point Source,” in *IJCAI*, 2022, pp. 3839–3846.
- [66] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu, “A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115671, 2023.
- [67] M. A. Nabian, R. J. Gladstone, and H. Meidani, “Efficient training of physics-informed neural networks via importance sampling,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 8, pp. 962–977, 2021.
- [68] S. Wang, H. Wang, J. H. Seidman, and P. Perdikaris, “Random weight factorization improves the training of continuous neural representations,” *arXiv preprint arXiv:2210.01274*, 2022.
- [69] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, “Fourier features let networks learn high frequency functions in low dimensional domains,” *Advances in neural information processing systems*, vol. 33, pp. 7537–7547, 2020.

- [70] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, “Adaptive activation functions accelerate convergence in deep and physics-informed neural networks,” *Journal of Computational Physics*, vol. 404, p. 109136, 2020.
- [71] S. D. Search, C. D. Cooper, and E. Van’t Wout, “Towards optimal boundary integral formulations of the Poisson–Boltzmann equation for molecular electrostatics,” *Journal of Computational Chemistry*, vol. 43, no. 10, pp. 674–691, 2022.
- [72] E. Jurrus, D. Engel, K. Star, K. Monson, J. Brandi, L. E. Felberg, D. H. Brookes, L. Wilson, J. Chen, K. Liles *et al.*, “Improvements to the APBS biomolecular solvation software suite,” *Protein Science*, vol. 27, no. 1, pp. 112–128, 2018.
- [73] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.
- [74] T. Developers, “TensorFlow,” *Zenodo*, 2022.
- [75] T. J. Dolinsky, P. Czodrowski, H. Li, J. E. Nielsen, J. H. Jensen, G. Klebe, and N. A. Baker, “PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations,” *Nucleic acids research*, vol. 35, no. suppl_2, pp. W522–W525, 2007.
- [76] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, “The protein data bank,” *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.
- [77] M. F. Sanner, A. J. Olson, and J.-C. Spehner, “Reduced surface: an efficient way to compute molecular surfaces,” *Biopolymers*, vol. 38, no. 3, pp. 305–320, 1996.
- [78] S. Decherchi and W. Rocchia, “A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale,” *PloS one*, vol. 8, no. 4, p. e59744, 2013.
- [79] D.-H. et al., “trimesh,” <https://trimesh.org/>, 12 2019, version 3.2.0.
- [80] C. T. Lee, J. G. Laughlin, J. B. Moody, R. E. Amaro, J. A. McCammon, M. J. Holst, and P. Rangamani, “An Open Source Mesh Generation Platform for Biophysical Modeling Using Realistic Cellular Geometries,” *Biophysical Journal*, Jan. 2020.
- [81] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, “SciPy 1.0: fundamental algorithms for scientific computing in Python,” *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

- [82] T. Betcke and M. Scroggs, “Bempp-cl: A fast Python based just-in-time compiling boundary element library,” *Journal of Open Source Software*, vol. 6, no. 59, pp. 2879–2879, 2021.
- [83] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [84] Plotly Technologies Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: <https://plot.ly>
- [85] S. Vijay-Kumar, C. E. Bugg, and W. J. Cook, “Structure of ubiquitin refined at 1.8 Å-resolution,” *Journal of molecular biology*, vol. 194, no. 3, pp. 531–544, 1987.
- [86] E. Fraenkel and C. O. Pabo, “Comparison of X-ray and NMR structures for the Antennapedia homeodomain–DNA complex,” *Nature structural biology*, vol. 5, no. 8, pp. 692–697, 1998.
- [87] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, “Kan: Kolmogorov-arnold networks,” *arXiv preprint arXiv:2404.19756*, 2024.

Anexos

A. Información Estructural de Moléculas

La información estructural de las moléculas modeladas (metanol, arginina, ubicuitina y ADN) fue obtenida del *Protein Data Bank* [76, 85, 86]. Los archivos .pdb fueron convertidos a archivos .pqr utilizando el software de pdb2pqr [75]. Por otra parte, los archivos .pqr para el Ion de Born y el multipolo esférico fueron de elaboración propia.

Se presentarán los archivos .pqr utilizados para el Ion de Born y el multipolo esférico. Para los archivos de las demás moléculas, referirse a [76].

A.1. Archivos PQR

Tabla A.1: Archivo .pqr para el Ion de Born

ATOM	1	I	SPH	1	0.00	0.00	0.00	1.00	1.00
------	---	---	-----	---	------	------	------	------	------

Tabla A.2: Archivo .pqr para la esfera con carga descentrada

ATOM	1	I	SPH	1	0.00	0.00	0.00	0.00	1.20
ATOM	2	I	SPH	1	0.45	0.00	0.00	1.00	0.00

Tabla A.3: Archivo .pqr para la esfera con 2 cargas positivas

ATOM	1	I	SPH	1	0.00	0.00	0.00	0.00	1.20
ATOM	2	I	SPH	1	0.50	0.00	0.00	1.00	0.00
ATOM	3	I	SPH	1	-0.50	0.00	0.00	1.00	0.00

Tabla A.4: Archivo .pqr para la esfera con 2 cargas de signos opuestos

ATOM	1	I	SPH	1	0.00	0.00	0.00	0.00	1.20
ATOM	2	I	SPH	1	0.50	0.00	0.00	1.00	0.00
ATOM	3	I	SPH	1	-0.50	0.00	0.00	-1.00	0.00

Tabla A.5: Archivo .pqr para la esfera con colección de 5 cargas

ATOM	1	I	SPH	1	0.00	0.00	0.00	1.00	1.20
ATOM	2	I	SPH	1	0.50	0.20	0.00	2.00	0.00
ATOM	3	I	SPH	1	-0.50	0.00	0.10	-1.00	0.00
ATOM	4	I	SPH	1	0.00	0.20	0.40	1.40	0.00
ATOM	5	I	SPH	1	0.00	-0.30	-0.10	-3.00	0.00

B. Librería XPPBE

En esta sección se mostrará un ejemplo simple de uso de XPPBE, más un ejemplo de los parámetros a definir por el usuario. Para revisar el código completo, referirse al repositorio [9].

B.1. Ejemplo de Uso de XPPBE

Se muestra un ejemplo simple de uso. Toda la configuración del modelo debe estar definida en el archivo `.yaml`, utilizado por la clase `Simulation`. Además, se debe especificar el directorio donde se encuentran los archivos `.pqr` o `.pdb`. En este ejemplo, se grafica el potencial y la energía de solvatación.

Código B.1: Ejemplo de uso de la librería XPPBE más post-procesamiento

```

1 # Example of the xppbe library
2
3 from xppbe import Simulation
4
5 # Paths to the configuration and molecule files
6 yaml_path = 'path/to/.yaml/file'
7 molecule_dir = 'path/to/.pqr/or/.pdb/directory'
8
9 # Creation of the simulation object
10 simulation = Simulation(yaml_path, molecule_dir)
11
12 # Setup of the algorithm
13 simulation.create_simulation()
14 simulation.adapt_model()
15
16 # Solve the model
17 simulation.solve_model()
18
19 # Postprocessing
20 post = simulation.postprocessing()
21
22 # Plot of loss history for different domains
23 post.plot_loss_history(domain=1);
24 post.plot_loss_history(domain=2);
25
26 # Plot of solvation energy evolution
27 post.plot_G_solv_history();
28
29 # Different plots of the reaction field
30 post.plot_phi_line(value='react');
31 post.plot_phi_contour(value='react');
32 post.plot_interface_3D(value='react');
```

B.2. Ejemplo del Archivo Requerido por XPPBE

En esta sección se incluirá un ejemplo del archivo .yaml con toda la información requerida por el software de XPPBE para resolver la ecuación de Poisson-Boltzmann utilizando PINNs.

Código B.2: Archivo .yaml de ejemplo

```
1 # Simulation settings
2
3 # Equation to solve
4 equation: regularized_scheme_2
5
6 # PBE model
7 pbe_model: linear
8 phi_units: qe_eps0_angs
9
10 # Domain properties
11 domain_properties:
12     molecule: arg
13     epsilon_1: 2
14     epsilon_2: 80
15     kappa: 0.125
16     T: 300
17
18 # Mesh properties
19 mesh_properties:
20     vol_max_interior: 0.06
21     vol_max_exterior: 0.6
22     density_mol: 10
23     density_border: 0.5
24     mesh_generator: nanoshaper
25     dR_exterior: 4
26
27 # Sampling method
28 sample_method: random_sample
29
30 # Losses to add
31 losses:
32     - R1
33     - R2
34     - D2
35     - Iu
36     - Id
37
38 # Weights adapting algorithm inputs
39 adapt_weights: true
40 adapt_w_iter: 1000
41 alpha_w: 0.7
```

```
42
43 # Architecture
44 num_network: 2
45
46 hyperparameters_in:
47     architecture_Net: FCNN
48     num_hidden_layers: 4
49     num_neurons_per_layer: 200
50     activation: tanh
51     adaptive_activation: true
52     fourier_features: true
53     weight_factorization: false
54     scale_input: true
55     scale_output: true
56
57 hyperparameters_out:
58     architecture_Net: FCNN
59     num_hidden_layers: 4
60     num_neurons_per_layer: 200
61     activation: tanh
62     adaptive_activation: true
63     fourier_features: true
64     weight_factorization: false
65     scale_input: true
66     scale_output: true
67
68 # Optimizer properties
69 optimizer: Adam
70 lr:
71     method: exponential_decay
72     initial_learning_rate: 0.001
73     decay_steps: 2000
74     decay_rate: 0.9
75     staircase: true
76
77 # Solve parameters
78 N_iters: 20000
79 iters_save_model: 5000
```

B.3. Instalación de XPPBE

Para instalar el software de XPPBE, basta correr el siguiente script en el terminal.

Código B.3: Ejemplo de instalación de XPPBE

```
1 # Clone the repository to your local machine
2 git clone https://github.com/MartinAchondo/XPPBE
3
4 # Navigate to the project directory
5 cd XPPBE
6
7
8 # Create a virtual environment
9 conda create --name xppbe python=3.9
10
11 # Activate the virtual environment
12 conda activate xppbe
13
14 # Install the project
15 pip install .
```

Luego de la instalación, la librería puede ser utilizada tal como lo muestra la sección B.1.