UNIVERSIDAD MICHOACANA
DE SAN NICOLÁS DE HIDALGO

DIVISIÓN DE ESTUDIOS DE POSGRADO

FACULTAD DE INGENIERÍA QUÍMICA

Desarrollo de aplicación web FermApp® TecNM/ITMorelia-UMICH-V1 para el análisis de procesos fermentativos

TESIS presentada por:

Juan Manuel Gutiérrez García

A la División de Estudios de Posgrado de la Facultad

de Ingeniería Química como requisito parcial

para autorizar el desarrollo de la tesis para obtener el grado de:

MAESTRO EN CIENCIAS

EN

INGENIERÍA QUÍMICA

Directora de tesis:

D. C. Ma. del Carmen Chávez Parga

Codirector de tesis:

D. C. Juan Carlos González Hernández

Morelia, Mich. **Agosto 2024**

Índice

Índice v

Índice de figuras viii

Índice de tablas ix

1. Introducción 2

1.1. Antecedentes 3

1.2. Justificación 5

1.3. Planteamiento del Problema 6

1.4. Hipótesis 7

1.5. Objetivos 8

1.5.1. General 8

1.5.2. Específicos 8

2. Marco Teórico 9

2.1. Procesos fermentativos 9

2.2. Modelos de biorreactor para la optimización de procesos de fermentación 12

2.2.1. Modelos mecanisticos para la optimización de procesos de fermentación . 12

2.2.2. Modelos de aprendizaje automático para la optimización de procesos de fermentación 16

2.3. Optimización de problemas no lineales, no convexos y multimodales mediante técnicas globales de optimización 21

2.3.1. Algoritmos genéticos en la optimización de procesos 26

2.4. Necesidad de usar programas computacionales para la optimización y simulación de procesos de fermentaciones 37

2.4.1. Programación en Python 38

3. Metodología **39**

3.1. Estimación de parámetros cinéticos mediante la implementación de algoritmos genéticos 39

3.2. Optimización del flujo de alimentación mediante la implementación de algoritmos genéticos 41

3.3. Construcción de la red neuronal artificial para simular procesos dinámicos de fermentación 43

3.4. Construcción de plataforma web para simulación y análisis de procesos fermentativos 45

4. Resultados **48**

4.1. Estimación de parámetros cinéticos mediante la implementación de algoritmos genéticos 48

4.1.1. Descripción del problema de optimización para la estimación de parámetros cinéticos 48

4.1.2. Planteamiento del problema de optimización para la estimación de parámetros cinéticos 50

4.1.3. Resultados de la implementación para la estimación de parámetros cinéticos mediante algoritmos genéticos 56

4.2. Optimización del flujo de alimentación mediante la implementación de algoritmos genéticos 60

4.2.1. Descripción del problema de optimización del flujo de alimentación en el biorreactor tipo lote-alimentado 60

4.2.2. Planteamiento del problema de optimización del flujo de alimentación . . 61

4.2.3. Resultados de la implementación de algoritmos genéticos para la optimización del flujo de alimentación 66

4.3. Funcionamiento de la red neuronal artificial para la simulación de procesos dinámicos de fermentación 72
4.3.1. Generación de trayectorias de entrenamiento y validación 73
4.3.2. Estructura de la red neuronal 76
4.3.3. Función de pérdida 77
4.3.4. Algoritmo de entrenamiento 78
4.3.5. Configuración de la red 78
4.3.6. Resultados obtenidos con las diferentes configuraciones 81
4.4. Desarrollo de aplicación web para automatizar el análisis y la optimización de procesos de fermentación a nivel biorreactor 85
4.4.1. Diseño de la base de datos 85
4.4.2. Diseño de la interfaz de usuario 91
4.4.3. Capturas de pantalla de la aplicación web 93
4.5. Casos de estudio para el análisis y la optimización de procesos fermentativos mediante FermApp® 98

5. Conclusiones **102**

6. Perspectivas **104**

Referencias **105**

A. ANEXO A. Metodología del desarrollo de la aplicación web FermApp® **111**
A.1. Recopilación y análisis de requerimientos 111
A.1.1. Almacenamiento de los datos experimentales 111
A.1.2. Visualización de datos 112
A.1.3. Análisis de datos 113
A.1.4. Registro y gestión de cuentas de usuario 114

A.2. Planificación 115
A.2.1. Selección de pila de tecnología 115
A.2.2. Diseño de la arquitectura del sistema 117
A.2.3. Plazos 118
A.2.4. Funciones y responsabilidades del equipo 119
A.2.5. Consideraciones de seguridad 120
A.2.6. Planificación de escalabilidad 120
A.3. Diseño del *frontend* 120
A.4. Desarrollo de la aplicación web 122
A.4.1. Configuración del entorno 122
A.4.2. Estructuración de la base de datos 123
A.4.3. Desarrollo del *backend* 123
A.4.4. Desarrollo del *frontend* 123
A.4.5. Integración 123
A.4.6. Pruebas 124
A.5. Pruebas 124
A.5.1. Pruebas unitarias 124
A.5.2. Pruebas de integración 125
A.5.3. Pruebas funcionales 125
A.5.4. Pruebas de usabilidad 125
A.5.5. Pruebas de rendimiento 125
A.5.6. Pruebas de seguridad 126
A.5.7. Pruebas de compatibilidad 126
A.6. Despliegue a producción 126
A.6.1. Selección del Plan de Alojamiento 127

A.6.2. Configuración de la cuenta de alojamiento 127
A.6.3. Configuración de la base de datos 127
A.6.4. Carga del sitio web 127
A.6.5. Instalación de dependencias 127
A.6.6. Configuración del entorno virtual 128
A.6.7. Configuración del modo de producción para Django 128
A.6.8. Lanzamiento de la aplicación 128
A.6.9. Configuración de implementación continua 128
A.6.10. Supervisión de aplicaciones 128

B. ANEXO B. Constancias y Participaciones **130**

Índice de figuras

1. Tipos de operación de fermentación en biorreactor. Las Imagenes fueron creadas por el autor utilizando BioRender. 10
2. Red neuronal con una sola capa escondida. 18
3. Funciones de activación. 19
4. Ejemplo de función convexa. 21
5. Ejemplo de función multimodal. 22
6. Representación de un algoritmo metaheurístico como una caja con entradas y salidas. 23
7. Dos criterios en conflicto en el diseño de una metaheurística: exploración contra explotación. 25
8. Esquema de algoritmo de selección proporcional en los AGs 29
9. Esquema de algoritmo de selección de torneo de los AGs. 30
10. Esquema de algoritmo de cruce de un solo punto en los AGs. 32
11. Esquema de algoritmo de cruce uniforme en los AGs. 33
12. Esquema de algoritmo de mutación de bit a bit en los AGs. 34
13. Diagrama de flujo del funcionamiento de un AG. 36
14. Esquema de la metodología para la estimación de parámetros cinéticos. 40
15. Esquema de la metodología para la optimización del flujo de alimentación. . . . 42
16. Esquema de la metodología para la construcción de la red neuroanl artificial. . . 45
17. Esquema de la metodología para el desarrollo de la aplicación web. 47
18. Esquema del biorreactor tipo lote para la optimización paramétrica 49
19. Gráfica de los datos generados para llevar a cabo la estimación de los parámetros cinéticos y evaluar la eficiencia del algoritmo de optimización. 50

20. Valores de los parámetros cinéticos del modelo de Monod que fueron estimados y sus intervalos de confianza. 58

21. Valores de los parámetros cinéticos del modelo de inhibición que fueron estimados y sus intervalos de confianza. 59

22. Esquema de un biorreactor tipo lote-alimentado para suministrar sustrato de forma gradual durante la fermentación. 60

23. Producción de biomasa en función del flujo de alimentación. Se muestran como líneas verticales los valores óptimos obtenidos en cada una de las estimaciones. . 69

24. Perfiles de alimentación óptimo de las cinco estimaciones. 70

26. Red neuronal para simular un proceso dinámico. 73

27. Trayectorias de entrenamiento con intervalos de 1 hora. 75

28. Trayectorias de validación con intervalos de tiempo de 1 hora. 76

29. Función *softplus* usada en las capas escondidas de las redes neuronales. 77

30. Escalador de tiempo directo. 79

31. Escalador de tiempo residual. 80

32. Escalador de tiempo numérico. 81

33. Evolución de la función de pérdida con las tres configuraciones de la red propuestas. 83

34. Trayectoria de validación. 83

35. Esquema del diseño de la base de datos de la aplicación. 91

36. Diseño de la aplicación web. En el esquema es posible apreciar todas las secciones de la aplicación web, así como su ubicación en la interfaz de usuario. 93

37. Interfaz Principal de la Aplicación Web FermApp®. 94

38. Interfaz de Estimación de Parámetros con Modelo Monod en FermApp®. 95

39. Interfaz de Análisis de Regresión Lineal en FermApp®. 96

40. Datos experimentales de la producción de compuestos por *Gibberella fujikuroi* y simulación del modelo de Monod con parámetros óptimos. 99

41. Datos experimentales de fermentaciones de xilitol y simulación del modelo de Monod con parámetros óptimos. 100

Índice de tablas

1. Condiciones iniciales y valores de los parámetros cinéticos usados para generar los datos de procesos de fermentación en biorreactor de operación tipo lote. . . . 50
2. Parámetros del algoritmo genético empleados para la estimación de los parámetros cinéticos. 56
3. Parámetros cinéticos que se optimizaron y sus rangos de búsqueda. 56
4. Valores de la función objetivo en las 5 estimación con ambos modelos. 57
5. Valores estadísticos de los parámetros cinéticos estimados usando el modelo de Monod. LI y LS sol los límites inferior y superior del intervalo de confianza, respectivamente. 57
6. Valores estadísticos de los parámetros cinéticos estimados usando el modelo de inhibición por sustrato. LI y LS sol los límites inferior y superior del intervalo de confianza, respectivamente. 58
7. Parámetros de operación usados en la optimización del flujo de alimentación. . . 67
8. Valores de los parámetros del algoritmo genético empleados en la optimización del flujo de alimentación. 67
9. Valores del flujo y perfiles óptimos, así como biomasa producida. 68
10. Valores de la función de pérdida tras finalizar el entrenamiento de la red. 82

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas aquellas personas que han hecho posible la realización de esta tesis y han contribuido a mi desarrollo profesional y personal durante el periodo de mi maestría.

En primer lugar, mis más profundos agradecimientos a mis padres, cuyo amor, apoyo y orientación han sido mi constante motivación y guía a lo largo de este camino. Su incondicional apoyo ha sido fundamental en cada paso que he dado.

Un especial reconocimiento a mis asesores, el Doctor Juan Carlos Gonzalez Hernández y la Doctora María del Carmen Chávez Parga, por su invaluable asesoría, paciencia y dedicación. Sus enseñanzas, consejos y apoyo han sido esenciales para la realización de este trabajo.

Mi gratitud a mis sinodales, la Doctora Elisa Domínguez Hüttinger, y los doctores José María Ponce Ortega y José Apolinar Cortés, por su tiempo, sus valiosas observaciones y comentarios que enriquecieron significativamente este trabajo.

No puedo dejar de mencionar y agradecer al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) por el apoyo económico brindado a través de la beca que me fue otorgada, lo que me permitió dedicarme de lleno a mis estudios de maestría.

Finalmente, agradezco a todos aquellos que, de una forma u otra, han sido parte de este viaje académico y personal. Cada uno de ustedes ha dejado una huella imborrable en mi vida.

Resumen

Esta tesis investiga rigurosamente la aplicación de algoritmos genéticos (GA) y redes neuronales artificiales (ANN) en el análsis de procesos de fermentación dentro de biorreactores discontinuos, contextualizados en la era transformadora de la Industria 4.0. Subraya el papel fundamental de la digitalización y la automatización en el avance de las prácticas de ingeniería química.

Un logro fundamental de esta investigación es la implementación exitosa de GA para la estimación rápida y precisa de parámetros cinéticos en fermentaciones en biorreactores discontinuos. Una parte integral de este avance es el desarrollo y despliegue a producción de una aplicación web, diseñada para automatizar estos procesos analíticos. La primera versión de esta aplicación, ya operativa, incorpora funcionalidades de estimación de parámetros y ha sido validada mediante datos simulados y experimentales. Esto marca un hito importante en la tarea de hacer que los procesos analíticos complejos sean más accesibles y fáciles de usar en el ámbito de la ingeniería química.

Al reconocer las limitaciones computacionales del modelo actual, particularmente en el manejo de intervalos de discretización más altos para la optimización del flujo de alimentación, la tesis propone un cambio estratégico hacia las ANN. Se prevé que esta transición permitirá un perfilado más eficiente del flujo de alimentación, mejorando potencialmente la producción de biomasa. Esta previsto que las versiones futuras de la aplicación web integren estos modelos ANN avanzados y análisis de optimización de la tasa de flujo de entrada, con énfasis en la aplicación de datos experimentales reales para la validación.

La tesis también evalúa la efectividad de varias configuraciones de ANN, con un enfoque en escaladores numéricos y residuales de tiempo. El escalador de tiempo residual se identifica como la opción preferida para futuras aplicaciones, considerando su reducido tiempo de entrenamiento.

Además de sus logros computacionales, la tesis profundiza en el desarrollo de una estructura de base de datos robusta para la aplicación web. Esta base de datos está meticulosamente diseñada para gestionar de manera eficiente un amplio espectro de datos experimentales. La interfaz de usuario de la aplicación se caracteriza por su diseño intuitivo, lo que garantiza facili-

dad de uso. La implementación gradual de la aplicación es estratégica, comienza con funciones de estimación de parámetros y evoluciona para incluir funcionalidades más complejas como simulaciones basadas en ANN y optimización del flujo de alimentación en futuras iteraciones.

Palabras clave: *Algoritmos Genéticos, Redes Neuronales Artificiales, Fermentación en Biorreactores Discontinuos, Industria 4.0, Optimización del Flujo de Alimentación*

Abstract

This thesis rigorously investigates the application of genetic algorithms (GAs) and artificial neural networks (ANNs) in the analysis of fermentation processes within batch bioreactors, contextualized in the transformative era of Industry 4.0. It underscores the pivotal role of digitalization and automation in advancing chemical engineering practices.

A core achievement of this research is the successful implementation of GAs for the swift and accurate estimation of kinetic parameters in batch bioreactor fermentations. Integral to this advancement is the development and deployment of a web application, designed to automate these analytic processes. The first version of this application, already operational, incorporates functionalities for parameter estimation and has been validated using simulated and experimental data. This marks a significant milestone in rendering complex analytical processes more accessible and user-friendly in the realm of chemical engineering.

Acknowledging computational limitations in the current model, particularly in handling higher discretization intervals for feed flow optimization, the thesis proposes a strategic shift to ANNs. This transition is anticipated to enable more efficient profiling for feed flow, potentially enhancing biomass production. The future versions of the web application are planned to integrate these advanced ANN models and inflow rate optimization analyses, with an emphasis on applying real experimental data for validation.

The thesis also evaluates the effectiveness of various ANN configurations, with a focus on time residual and numerical scalers. The time residual scaler is identified as the preferred choice for future applications, considering its reduced training time.

In addition to its computational achievements, the thesis elaborates on the development of a robust database structure for the web application. This database is meticulously designed to manage a broad spectrum of experimental data efficiently. The user interface of the application is characterized by its intuitive design, ensuring ease of use and engagement. The phased rollout of the application is strategic, beginning with parameter estimation features, and evolving to include more complex functionalities like ANN-based simulations and feed flow optimization in future iterations.

1. Introducción

Desde 2011, se ha reconocido que la industria está atravesando una transición hacia la Cuarta Revolución Industrial, o Industria 4.0. Este concepto, introducido por el gobierno alemán, destaca la importancia de la computarización y automatización en los procesos de producción (Mowbray y cols., 2021). En este contexto, la ingeniería química está evolucionando hacia una digitalización y automatización completas, lo que incrementa la disponibilidad de datos y la necesidad de tomar decisiones basadas en esta información (Schweidtmann y cols., 2021). Por tanto, es crucial desarrollar herramientas que faciliten esta transición hacia una Industria 4.0 más integrada y eficiente en el sector químico.

Los modelos mecanicistas han sido fundamentales en la automatización de procesos químicos, permitiendo simular diversas condiciones operativas y establecer estrategias de control (Contois, 1959; Andrews, 1968; Lee y cols., 1983; Bailey, 1998; Chen y cols., 2010; Mowbray y cols., 2021). Su aplicación mejora la comprensión de los procesos, aunque su desarrollo puede ser computacionalmente intensivo y demandar tiempo y conocimientos especializados (Villadsen y cols., 2011). Paralelamente, el campo del aprendizaje automático ha emergido como un enfoque poderoso para analizar datos y descubrir patrones en sistemas complejos (Mowbray y cols., 2021). Su aplicación en la ingeniería química y bioquímica ha crecido rápidamente debido a su eficiencia, flexibilidad y facilidad de uso (Rodriguez-Granrose y cols., 2021; Trinh y cols., 2021; Thebelt y cols., 2022; Samal y cols., 2022; Moon y cols., 2022).

Sin embargo, la implementación efectiva de modelos mecanicistas y de aprendizaje automático en la simulación y optimización de procesos aún enfrenta barreras significativas, especialmente en el contexto de la fermentación. Herramientas computacionales como MATLAB®, SuperPro Designer® y Aspen Plus® son ampliamente utilizadas, pero su acceso limitado y la falta de soporte para el aprendizaje automático simplificado representan desafíos notables. Este escenario subraya la necesidad de herramientas más accesibles y adaptadas, como librerías específicas, interfaces gráficas de usuario, repositorios de código abierto y bases de datos que puedan automatizar y facilitar el uso de estos modelos avanzados. El desarrollo de tales herramientas no solo avanzará la optimización de procesos de fermentación, sino que también democratizará el acceso a tecnologías de vanguardia en la ingeniería química.

1.1. Antecedentes

El desarrollo de programas para la optimización de procesos fermentativos está poco extendido, sin embargo, existen programas tanto de distribución comercial como programas de distribución libre que llevan a cabo algunas de estas tareas. Por ejemplo, Ribas-García y cols. (2014) desarrollaron el software `FERMENTA` para la simulación de procesos fermentativos, este software permite la simulación de modos de fermentación en continuo y discontinuo. Las expresiones cinéticas pueden ser seleccionadas por el usuario entre un conjunto de más de 30 posibilidades, los parámetros en el modelo elegido pueden ser determinados mediante los datos experimentales.

Phan-thien (2011) desarrollaron el software `BioMass` mediante el uso del lenguaje de programación `Visual Basic`. En el programa el usuario puede elegir y especificar el tipo de alimentación al biorreactor. El programa permite simular un reactor tipo lote o continuo en la fase transitoria y estable. Dentro de los modelos cinéticos contiene modelos para inhibición competitiva y no competitiva. Además, es posible simular procesos de fermentación con múltiples sustratos. Se empleó el programa para analizar los datos de un proceso de generación de hidrógeno a partir de glucosa empleando la bacteria *Thermotoga neapolitana*.

Silva y Inácio (2015) desarrollaron un programa haciendo uso del lenguaje de programación `MATLAB`. El programa se empleó para simular y optimizar un proceso de fermentación y sacarificación simultaneo (proceso mediante el cual un polisacárido se transforma en azúcar fermentable), el microorganismo que se empleó en las simulaciones fue la levadura *Saccharomyces cerevisiae*. Se simularon escenarios de operación tipo lote y continuo, encontrando los mayores rendimientos en el segundo modo de operación.

En un trabajo similar Oliveira y cols. (2017) presentaron el paquete de software `AnaBioPlus`, el cuál fue desarrollado para el análisis de procesos en biorreactor. El paquete consta de dos programas: `OptimusFerm` y `SimulaFerm`. `OptimusFerm` permite el análisis de conjuntos de datos experimentales de cultivo por lotes para estimar parámetros de modelos. `SimulaFerm` realiza simulaciones obteniendo valores de crecimiento celular, formación de productos y consumo de sustrato para cultivos discontinuos, alimentados y continuos. Treinta y dos modelos cinéticos de crecimiento celular están disponibles en `OptimusFerm` y `SimulaFerm`.

Hemmerich y cols. (2021) desarrollaron una librería de `Python` que permite la simulación

y optimización de procesos fermentativos mediante ecuaciones diferenciales ordinarias que se crean usando un programación orientada a objetos. Entre las funcionalidades de la librería están la estimación de parámetros cinéticos y la simulación de procesos de fermentación tipo lote-alimentado. Además, el código que se utilizó para el desarrollo de la librería se encuentra disponible en GitHub junto con ejemplos de como usarla.

A pesar de la existencia de programas para el modelado y optimización de procesos fermentativos, estos no siempre están disponibles para el público o, en caso de estarlo, son de difícil acceso. Además, a menudo el usuario no tiene acceso al código utilizado para su desarrollo, lo que limita la comprensión de las funciones y algoritmos empleados en la simulación y optimización. Incluso cuando el código está disponible, su uso de lenguajes de programación poco comunes dificulta su comprensión y utilización por parte de los usuarios.

1.2. Justificación

La industria de la fermentación desempeña un papel crucial en varios sectores, incluidos los alimentarios, farmacéuticos y biotecnológicos. La eficiencia y precisión en los procesos de fermentación son fundamentales para garantizar la calidad del producto, la rentabilidad y la sostenibilidad de la producción. Sin embargo, la industria actualmente se enfrenta a desafíos significativos en la estimación de parámetros cinéticos y la optimización de estos procesos, debido en gran parte a la dependencia de métodos manuales y convencionales. Estos métodos no solo son propensos a errores y menos eficientes, sino que también limitan la capacidad para adaptarse a condiciones cambiantes y escalar procesos de manera efectiva.

El desarrollo de una aplicación web que automatice y facilite el uso de modelos mecanicistas y de aprendizaje automático aborda directamente estos desafíos. Esta plataforma permitirá una estimación más precisa y rápida de parámetros cinéticos, una optimización más efectiva de los procesos y una mejor capacidad de respuesta a las variables en juego. La automatización y la analítica avanzada tienen el potencial de transformar la industria, llevando a mejoras en la eficiencia operativa, reducciones en el consumo de recursos y energía, y una mayor coherencia y calidad del producto final.

Además, al ser una aplicación web, la plataforma ofrece accesibilidad y facilidad de uso, lo que es crucial para su adopción en una industria con una amplia gama de usuarios con diferentes niveles de experiencia técnica. La combinación de una interfaz gráfica intuitiva con un poderoso *backend* técnico garantizará que la plataforma no solo sea potente en términos de capacidad analítica, sino también accesible y fácil de usar para ingenieros y científicos.

1.3. Planteamiento del Problema

En la industria de la fermentación, la estimación precisa de parámetros cinéticos y la optimización efectiva de los procesos son esenciales para la eficiencia de producción y la calidad del producto. Sin embargo, estos procesos a menudo se ven obstaculizados por métodos tradicionales que dependen de la intervención manual y de enfoques menos precisos, lo que conduce a ineficiencias operativas y limitaciones en la escalabilidad. Además, la implementación de modelos mecanicistas y de aprendizaje automático, aunque prometedora, se enfrenta a desafíos significativos debido a la falta de herramientas automatizadas y accesibles que faciliten su aplicación en entornos industriales. Esta brecha en la tecnología actual limita la capacidad de los profesionales para realizar análisis detallados y optimizar los procesos de fermentación de manera efectiva. Por lo tanto, surge la necesidad de una solución que no solo automatice y simplifique la utilización de estos modelos avanzados, sino que también haga el proceso más accesible y eficiente para ingenieros y científicos en la industria.

1.4. Hipótesis

La implementación de una aplicación web que automatiza la utilización de modelos mecanicistas y de aprendizaje automático para la estimación de parámetros cinéticos y la optimización de procesos de fermentación resultará en una mejora significativa en la precisión y eficiencia de estos procesos. Esta plataforma, al integrar herramientas especializadas, una interfaz de usuario intuitiva y un sistema de gestión de datos efectivo, facilitará a los ingenieros y científicos en la industria de la fermentación la realización de análisis complejos, la toma de decisiones basada en datos y la optimización de procesos, superando las limitaciones de los métodos tradicionales y manuales.

1.5. Objetivos

1.5.1. General

Desarrollar y desplegar una aplicación web avanzada que automatice la utilización de modelos mecanicistas y de aprendizaje automático para la estimación precisa de parámetros cinéticos y la optimización de procesos de fermentación.

1.5.2. Específicos

1. Crear y validar un conjunto de funciones y algoritmos específicos para la optimización de procesos de fermentación, utilizando `Python`.
2. Establecer una interfaz de usuario amigable, responsiva y funcional que permita a los usuarios interactuar fácilmente con las herramientas de optimización.
3. Establecer una base de datos robusta y escalable con `PostgreSQL` para almacenar, gestionar y analizar datos relacionados con los procesos de fermentación.
4. Evaluar la plataforma integrada mediante la estimación de parámetros cinéticos a partir de datos reales de fermentación, combinando las librerías desarrolladas en `Python`, la IGU creada con `React JS` y `Bootstrap`, y el sistema de gestión de datos en `PostgreSQL`.

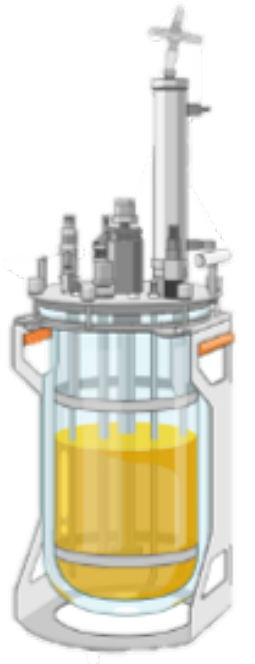
2. Marco Teórico

2.1. Procesos fermentativos

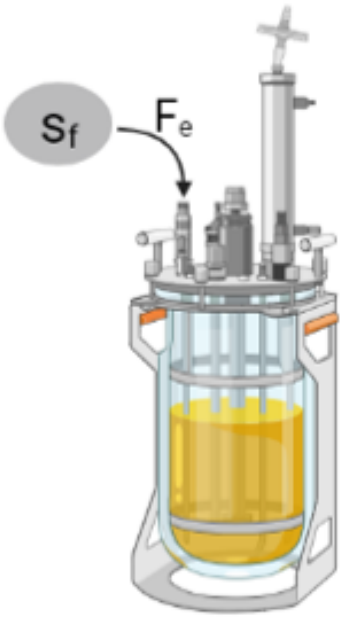
Los procesos de fermentación utilizan sistema biológicos, como microorganismos y enzimas, para convertir materia prima con alto contenido de azúcares en productos de valor comercial, como biocombustibles. La fermentación suele llevarse a cabo en un recipiente llamado biorreactor, en el cual se propician las condiciones óptimas para el crecimiento de los microorganismos. La fermentación se inicia inoculando al microorganismo con el medio de cultivo en el bioreactor. Una vez que termina la fermentación, el producto crudo se puede usar directamente o se puede procesar para aislar entidades moleculares específicas de él (Soetaert y Vandamme, 2010).

La fermentación se suele realizar mediante tres diferentes tipos de operación (Lim y Shin, 2013). En la Figura 1 se muestra un esquema de estos tres tipos de operación. Las operaciones tipo lote y tipo lote-alimentado son comunes en la industria, mientras que la operación tipo continuo no se suele utilizar dado que resulta complicada de controlar. Los tres tipos de operación guardan ciertas características en común, algunas de las más importantes son:

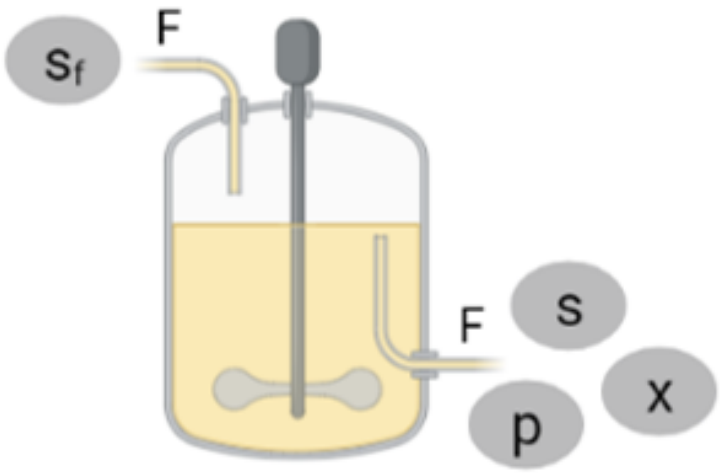
- El medio de cultivo, el cual contiene los sustratos que serán consumido durante la fermentación, se introduce en condiciones estériles al biorreactor previo al inicio de la fermentación.
- Para que inicie la fermentación la cepa o cepas de microorganismos a usar se inoculan en condiciones estériles.
- La temperatura permanece constante durante la fermentación. La temperatura elegida es la temperatura de óptimo crecimiento del microorganismo, o bien, la temperatura óptima para la generación de un producto de su metabolismo.



(a) Operación tipo lote



(b) Operación tipo lote-alimentado



(c) Operación tipo continuo

Figura 1.- Tipos de operación de fermentación en biorreactor. Las Imagenes fueron creadas por el autor utilizando BioRender.

La principal diferencia entre los tres tipos de operación radica en los flujos de alimentación y de salida del biorreactor. El flujo de alimentación contiene el medio de cultivo con los sustratos que los microorganismos consumen, mientras que el flujo de salida contiene sustratos sin consumir, biomasa y productos del metabolismo de los microorganismos.

Operación tipo lote

En la operación tipo lote no existen flujos de alimentación ni de salida del biorreactor. El sustrato que es consumido por el microorganismo es el que se introduce al inicio de la fermentación al biorreactor. Este tipo de operación se utiliza cuando se pueden mantener elevadas concentraciones de sustrato en el biorreactor sin afectar el crecimiento de los microorganismos. Además, dado que no hay flujos, es el tipo de operación más simple de realizar. Es bastante común usar este tipo de operación en la producción de alcohol y otros productos de bajo valor comercial (Cinar y cols., 2003).

Operación tipo lote-alimentado

En este tipo de operación existe un flujo de alimentación al biorreactor. Este flujo de alimentación contiene el sustrato que los microorganismos requieren para su crecimiento. El flujo puede permanecer constante o variar durante la fermentación. Este tipo de operación se utiliza cuando los microorganismos utilizados presentan algún tipo de inhibición hacia los sustratos. Con el fin de evitar una concentración elevada de sustrato que afecte el crecimiento de los microorganismos, el sustrato se introduce de manera gradual al biorreactor. Dado que es posible controlar el flujo de alimentación durante la operación del biorreactor, una pregunta de interés es la de cómo variar este flujo de manera que se maximice la producción de biomasa y/o producto. Si se establece un flujo de alimentación demasiado bajo, la cantidad de sustrato suministrada a los microorganismos puede ser insuficiente para favorecer su crecimiento; por otro lado, un flujo demasiado elevado puede dar lugar a que exista una alta concentración de sustrato en el biorreactor, lo cual resulta tóxico para los microorganismos que presentan inhibición. Este tipo de operación se suele utilizar en la generación de biofármacos, donde el producto es de alto valor comercial y existe un gran interés en maximizar su producción (Cinar y cols., 2003).

Operación tipo continuo

En la operación tipo continuo se tiene un flujo de alimentación al biorreactor y un flujo de salida. Al inicio de la fermentación este tipo de biorreactor se opera de manera similar a un biorreactor tipo lote-alimentado, es decir, con un flujo de alimentación y sin un flujo de salida. Una vez que se alcanza cierta concentración de biomasa, se empiezan a retirar los productos de la fermentación a través del flujo de salida. El flujo de alimentación y de salida se igualan de manera que el volumen permanece constante. En este tipo de operación el objetivo es

mantener condiciones estacionarias durante la fermentación de manera que se estén obteniendo los productos de la fermentación a través del flujo de salida de manera controlada. Es común utilizar este tipo de operación en el tratamiento de aguas residuales, donde continuamente se está introducción agua contaminada con desechos orgánicos al biorreactor y se obtiene agua menos contaminada en el flujo de salida (Lim y Shin, 2013).

2.2. Modelos de biorreactor para la optimización de procesos de fermentación

Un paso fundamental en la optimización de procesos de fermentación es establecer un modelo que describa los flujos de masa, las reacciones y el efecto de condiciones de operación sobre la fermentación. Existen varios tipos de modelos y clasificaciones de los mismos. Aquí se muestra un criterio de clasificación que distingue entre modelos macanisticos y modelos empíricos.

2.2.1. Modelos mecanisticos para la optimización de procesos de fermentación

Los modelos de fermentación mecanisticos están compuestos por un conjunto de ecuaciones diferenciales que describen el cambio de concentración de biomasa, sustrato, y producto con respecto al tiempo. Estas ecuaciones diferenciales se obtienen al realizar balances de masa sobre el biorreactor (Mears y cols., 2017). En los balances se consideran los flujos de entrada y salida del biorreactor, así como las reacciones que se presentan en el interior de este. El balance sobre el biorreactor dependerá en gran medida del tipo de operación que se utiliza. Aunque existen tres diferentes tipos de operación: lote, lote-alimentado, y continuo, en la mayoría de las aplicaciones industriales se suele utilizar solamente las operaciones tipo lote y lote alimentado, y estas son los que se consideran en el presente trabajo. Algunas de las suposiciones que se realizan al momento de plantear los modelos de biorreactor son: la composición del medio en el interior del biorreactor es homogénea, todos los microorganismos son iguales, existe solo un sustrato limitante, y los parámetros cinéticos permanecen constantes durante la fermentación.

Cuando la operación del biorreactor es tipo lote, no existen flujos en la entrada o en la salida del biorreactor y, por lo tanto, el volumen permanece constante. En ese caso, el modelo del biorreactor que describe la fermentación está compuesto por las siguientes ecuaciones.

$$\frac{dx}{dt} = r_x - k_d x \tag{1}$$
$$\frac{ds}{dt} = -r_s \tag{2}$$
$$\frac{dp}{dt} = r_p - k_p p \tag{3}$$

donde: x representa la concentración de biomasa (g/L), s representa la concentración de sustrato (g/L), y p representa la concentración de producto (g/L). Del lado derecho de la ecuación (1), el término r_x y el término $k_d x$ representan la velocidad de crecimiento y la velocidad de muerte del microorganismo (g/L h), respectivamente. El término r_s en la ecuación (2) representa la velocidad de consumo de sustrato (g/L h). El término r_p y el término $k_p p$ en la ecuación (3) representan la velocidad de generación y degradación de producto (g/L h), respectivamente.

Cuando la operación del biorreactor es tipo lote-alimentado, existe un flujo de alimentación, lo cual ocasiona un incremento en el volumen. En ese caso, el modelo del biorreactor que describe el proceso es.

$$\frac{dV}{dt} = F_e \tag{4}$$
$$\frac{dx}{dt} = r_x - k_d x - \frac{F_e}{V} x \tag{5}$$
$$\frac{ds}{dt} = -r_s + \frac{F_e}{V}(s_f - s) \tag{6}$$
$$\frac{dp}{dt} = r_p - k_p p - \frac{F_e}{V} p \tag{7}$$

donde: V representa el volumen del biorreactor (L), F_e representa el flujo de alimentación al biorreactor (L/h), y s_f representa la concentración de sustrato en el flujo de alimentación (g/L).

Modelos cinéticos de crecimiento microbiano

Para describir la velocidad de crecimiento del microorganismo, r_x, se han propuesto varios modelos de crecimiento microbiano. De todos ellos, el modelo de Monod es el más usado. El modelo de Monod se basa en el modelo de Michaelis – Menten para describir la velocidad de una reacción enzimática y se escribe como (Monod, 1949).

$$r_x = \mu(s)x \tag{8}$$

$$\mu(s) = \mu_{max}\frac{s}{k_s + s} \tag{9}$$

donde: $\mu(s)$ representa la velocidad específica de crecimiento del microorganismo, el parámetro μ_{max} representa la velocidad específica máxima de crecimiento del microorganismo (1/h). El parámetro k_s representa la afinidad del microorganismo por el sustrato (g/L).

Las velocidades de consumo de sustrato, r_s, y la velocidad de generación de producto, r_p , se pueden escribir como funciones de la velocidad de crecimiento de microorganismo y la biomasa como sigue (Wang y cols., 2008).

$$r_s = \frac{1}{Y_{xs}}r_x \tag{10}$$

$$r_p = = \alpha r_x + \beta x \tag{11}$$

donde: el parámetro Y_{xs}, rendimiento biomasa - sustrato, representa la cantidad de biomasa generada a partir de la masa de sustrato. En la ecuación (11), el primer término representa la generación de producto asociada al crecimiento, y el segundo término representa la generación de producto no asociada al crecimiento. Por lo tanto, el parámetro α es una medida de la cantidad de producto que se genera debido al crecimiento de los microorganismos y el parámetro β es una medida de la cantidad de producto que se genera debido a la presencia del microorganismo.

Los valores de los parámetros cinéticos presentes en los modelos del biorreactor brindan información sobre la capacidad de los microorganismos para transformar los sustratos en los productos y biomasa que se desean generar. Por esta razón, el valor de estos parámetros se puede utilizar como criterio para seleccionar al microorganismo más adecuado para un proceso de fermentación.

El valor de la velocidad específica máxima de crecimiento, μ_{max}, permite conocer qué tan rápido crecen los microorganismos durante la fase de crecimiento exponencial. Los valores de este parámetro que se han reportado en la literatura suelen ir desde 0.01 g/L hasta 2 g/L

(Sheng y Sheu, 2000; Amenaghawon y cols., 2012; De Ovalle y cols., 2018). Un valor alto de este parámetro indica que el microorganismo experimenta un crecimiento acelerado. En ese caso, se usará un periodo de fermentación breve si lo que se desea es producir biomasa, dado que esta llegará a su máxima concentración en poco tiempo. Por otro lado, un valor bajo de este parámetro implicará mayores tiempos de fermentación para que el microorganismo llegue a su concentración máxima de biomasa. Muchas veces el tiempo que dura la fermentación implica un elevado costo de operación, por lo cuál se buscará utilizar un microorganismo que posea una velocidad máxima de crecimiento grande.

Los valores de la constante de afinidad al sustrato, k_s, suelen variar más en comparación con el resto de parámetros cinéticos, con valores que van desde los 0.2 hasta los 300 g/L (Sheng y Sheu, 2000; Zapata M y cols., 2005; Phisalaphong y cols., 2006; Amenaghawon y cols., 2012; Ariyajaroenwong y cols., 2016). Un valor de la constante de afinidad bajo implica que el microorganismo posee la habilidad para consumir el sustrato de forma eficiente. Esto puede deberse a que produce cantidades suficientes de las enzimas implicadas en la degradación y absorción de este sustrato. Un valor alto implicará que el microorganismo no es capaz de consumir y transforma el sustrato de forma eficiente.

El rendimiento biomasa-sustrato, Y_{xs}, suele tener valores entre 0.1 y 1 (Joelianto y cols., 2001; Rivera y cols., 2006; Angelova y cols., 2010; Alvarez-Ainza y cols., 2021). Un valor alto de este parámetro indica que el microorganismo es eficiente para transformar el sustrato en biomasa. Por esta razón, si el objetivo de la fermentación es producir biomasa, se buscará usar un microorganismo con un elevado valor de este parámetro.

Cuando se utilizan condiciones de operación tipo lote-alimentado es importante estudiar el valor del flujo de alimentación, F_e. Este parámetro, a diferencia de los parámetros anteriores, se considera un parámetro de operación el cual es posible controlar durante la fermentación. El flujo de alimentación permite introducir sustrato al biorreactor a lo largo de la fermentación. El valor de este parámetro dependerá de las dimensiones del biorreactor, numerosos trabajos han reportado valores de este parámetro en la optimización de biorreactores tipo lote alimentado (Yüzgeç, 2010; Rocha y cols., 2014; Patel y Padhiyar, 2016; Bin Mohd Zain y cols., 2018). En biorreactores de gran capacidad, por ejemplo 200 m^3, el flujo de alimentación puede ser de 200 L/ h. En cambio, en biorreactores de escala laboratorio, alrededor de 3 L, el flujo de alimentación

será mucho menor, alrededor de 0.5 L/h.

2.2.2. Modelos de aprendizaje automático para la optimización de procesos de fermentación

Para la comprensión y optimización de procesos en sistemas complejos es común el uso de modelos mecanisticos. Sin embargo, el desarrollo de estos modelos puede demandar mucho tiempo y conocimiento del proceso. Además, muchos fenómenos no pueden ser completamente descritos por modelos mecanisticos tractables computacionalmente. Otro problema con los modelos mecanisticos es que pueden ser computacionalmente costos de simular, lo que limita su uso para la simulación de procesos complejos (Schweidtmann y cols., 2021).

Los modelos de aprendizaje automático tienen ventajas resaltables sobre las técnicas tradicionales de modelado, e.g., no requieren tanto tiempo e investigación para ser desarrollados, en general, son más veloces a la hora de realizar cálculos complejos y tienen un alto grado de precisión en su espacio de entrenamiento. Por otro lado, también tienen algunas desventajas, como la falta de interpretabilidad y la alta cantidad de datos que requieren para su entrenamiento (Dobbelaere y cols., 2021).

En los años ochenta y noventa se realizaron esfuerzos por combinar técnicas de inteligencia artificial y aprendizaje automático con la ingeniería química sin que se lograrán satisfacer las expectativas que se tenían (Dobbelaere y cols., 2021). Los modelos de inteligencia artificial y aprendizaje automático que se usaron en esos años por la comunidad de ingeniería química fueron los sistemas expertos y las redes neuronales superficiales. Estas tecnologías tuvieron un impacto limitado debido a dos principales factores (Schweidtmann y cols., 2021):

1. La falta de datos y su accesibilidad, así como de poder computacional.
2. El éxito de tecnologías emergentes para la comunidad de ingeniería química, en particular, modelos mecanisticos, algoritmos de optimización (e.g., algoritmos genéticos) y modelos de control predictivo.

En los últimos años, el incremento en la disponibilidad de datos y recursos computacionales ha impulsado un renovado interés por el uso de herramientas de aprendizaje automático en ingeniería química (Dobbelaere y cols., 2021). El creciente interés y uso de técnicas de aprendizaje

automático puede ser atribuido a un rápido avance en el desarrollo de hardware (e.g., unidades de procesamiento gráfico (UPG)) y a los algoritmos de aprendizaje profundo que permiten el entrenamiento de modelos complejos, así como a sistemas digitales que aceleran la acumulación de datos (Gao y cols., 2022). De forma general, el aprendizaje automático se ha vuelto muy accesible gracias a la creación de paquetes como Scikit-learn (Pedregosa y cols., 2011), Tensorflow y Keras.

Antes de usar cualquier tipo de modelo de aprendizaje automático, es necesario llevar a cabo una serie de pasos para asegurar la calidad de los datos empleados. La serie de pasos que suele seguirse para asegurar la calidad de los datos que va desde su generación hasta su almacenamiento y uso en la construcción de modelos se conoce como curación de datos (Dobbelaere y cols., 2021).

Las aplicaciones más comunes del aprendizaje automático es para simular sistemas con una gran cantidad de datos. Sin embargo, la mayoría de los estudios en ingeniería química generan bases de datos pequeñas. Se han desarrollado diferentes estrategias de aumentación de datos para trabajar con conjuntos de datos escasos (Mowbray y cols., 2021):

1. La solución más sencilla es añadir ruido aleatorio al conjunto de datos original. Debido a la estocasticidad inherente del proceso, se puede asumir que cualquier dato generado se encuentra dentro de cierto rango de los datos originales.

2. La segunda estrategia es la construcción de modelos mecanisticos del proceso. Estos modelos son ajustados a los datos experimentales originales. Una vez que los modelos han sido calibrados pueden ser usados para simular diferentes condiciones de operación. Comparado con el método anterior, este método brinda información física adicional en los datos sintéticos y mejora la precisión del modelo de aprendizaje automático.

Redes neuronales artificiales

Una red neuronal toma un vector de entrada de p variables $X = (X_1, X_2, \ldots, X_p)$ y construye una función no lineal $f(X)$ para predecir la respuesta Y. La Figura 2 muestra una red neuronal simple para modelar una respuesta cuantitativa utilizando $p = 4$ predictores. En la terminología de las redes neuronales, las cuatro variables $X_1, \ldots, X_4$ constituyen las unidades en la capa de

entrada. Las flechas indican que cada una de las entradas de la capa de entrada alimenta cada una de las K unidades de la capa oculta.

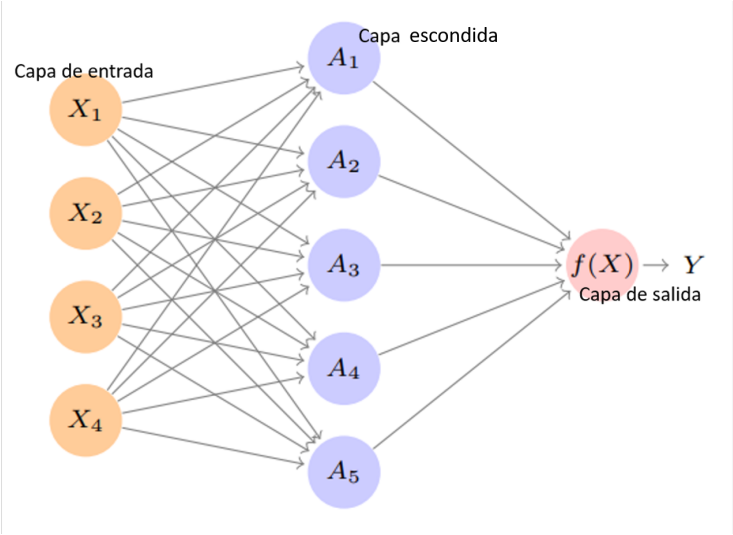


Figura 2.- Red neuronal con una sola capa escondida. La capa escondida calcula activaciones $A_k = h_k(X)$ que son transformaciones no lineales de las variables de entrada $X_1, X_2, \ldots, X_p$. Por lo tanto A_k no se observan de manera directa. La capa de salida es un modelo de regresión lineal que usa a las activaciones A_k como entradas, resultando en una función $f(X)$. Adaptado de (James, G. et al. (2013). An Introduction to Statistical Learning with Applications in R. Springer.).

El modelo de red neuronal tiene la siguiente forma.

$$\begin{aligned} f(X) &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X) \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j) \end{aligned} \tag{12}$$

Se construye mediante dos pasos. Primero la K activaciones $A_k, k = 1, ..., K$, en la capa escondida son calculadas como funciones de las variables de entrada $X_1, \ldots, X_p$,

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j) \tag{13}$$

donde $g(z)$ es una función de activación no lineal que debe ser especificada. Podemos pensar

en cada A_k como una transformación distinta $h_k(X)$ de las variables originales. Estas K funciones de activación de la capa escondida son alimentadas a la capa de salida, lo cual resulta en.

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k \tag{14}$$

un modelo de regresión lineal en las $K = 5$ activaciones. Todos los parámetros $\beta_0, \ldots, \beta_K$ y $w_{10}, \ldots, w_{K_p}$ necesitan ser estimados de los datos.

En las primeras redes neuronales, se utilizaba la función de activación sigmoidea.

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}} \tag{15}$$

La elección preferida para las redes neuronales modernas es la función de activación ReLU (unidad lineal rectificada), la cual toma la forma.

$$g(z) = \begin{cases} 0 & \text{si} \quad z < 0 \\ z & \text{si} \quad z >= 0 \end{cases} \tag{16}$$

La función de activación ReLU puede ser calculada y almacenada de forma más eficiente que la función de activación sigmoidea.

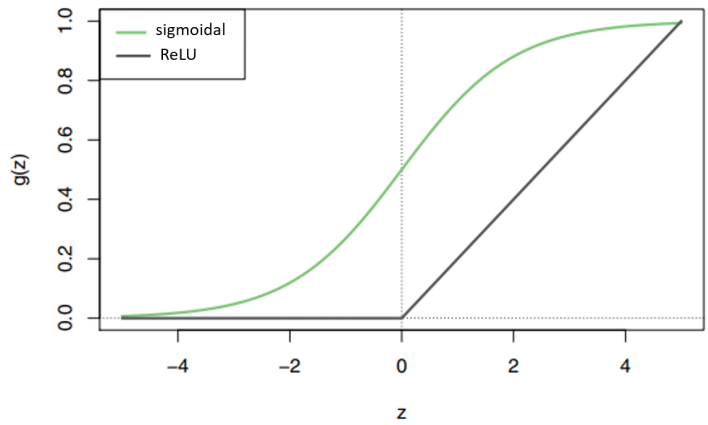


Figura 3.- Funciones de activación. La función de activación ReLU es más popular que la función sigmoidea por su eficiencia y computabilidad.

Los hiperparámetros determinan la estructura y precisión de las redes neuronales. Para asegurar la robustez y precisión de las red neuronal, los hiperparámetros deben ser seleccionados de manera cuidadosa. Los hiperparámetros típicos en una red neuronal incluyen el número de capas escondidas, el número de neuronas por capa, la tasa de aprendizaje, el número de épocas, las funciones de activación en cada capa y el algoritmo de optimización (Moon y cols., 2022).

La primera vez que se publicó un artículo en el que se hacía uso de una red neuronal en ingeniería bioquímica fue en 1990 para la predicción en línea de variables de fermentación (Thibault y cols., 1990). En un artículo de revisión sobre la aplicación de aprendizaje automático en ingeniería bioquímica, publicado en 2021, los autores señalan que de la bibliografía consultada, que va del año 2000 al año 2020, 129 de los 200 artículos consultados hacen uso de alguna forma de red neuronal (Mowbray y cols., 2021).

Las principales aplicaciones de las redes neuronales en la ingeniería de biorreactores son:

1. La estimación y predicción de variables de estado (e.g., la concentración de biomasa celular).
2. Mejorar la eficiencia y desempeño del biorreactor a través de la optimización de las condiciones de operación.
3. Monitorear las condiciones del proceso para propósitos de seguridad y control.

Para lograr la adopción de estas herramientas de modelado en la comunidad de ingeniería química es importante desarrollar modelos, código y datos que sean fáciles de usar. Datos de alta calidad y modelos de libre acceso motivan a los investigadores a usar modelos de aprendizaje automático como una herramienta y les brinda la oportunidad de enfocarse es sus problemas en lugar de en la programación y recolección de datos. Vale la pena además mencionar que sin la correcta documentación de los modelos de aprendizaje automático sus resultados son a menudo difíciles de reproducir (Mowbray y cols., 2021). Producir descripciones rigurosas y transparentes de los modelos de aprendizaje automático y permitir que los resultados sean reproducibles es esencial para convencer a la comunidad no académica de ingeniería química de sus beneficios.

2.3. Optimización de problemas no lineales, no convexos y multimodales mediante técnicas globales de optimización

En los problemas de optimización convexos, donde las variables de decisión son continuas y tanto la función objetivo como la región factible son convexas, cualquier óptimo local es un óptimo global (Himmelblau y cols., 2001). Muchos problemas no satisfacen estas condiciones, y a menudo es difícil verificar si ellos las satisfacen o no (Figura 4). Modelos que incluyen restricciones de igualdad no lineales caen dentro de esta última categoría. Problemas de optimización no convexos pueden ocurrir cuando se estima el valor de los parámetros de un modelo mediante mínimos cuadrados o funciones de máxima verisimilitud.

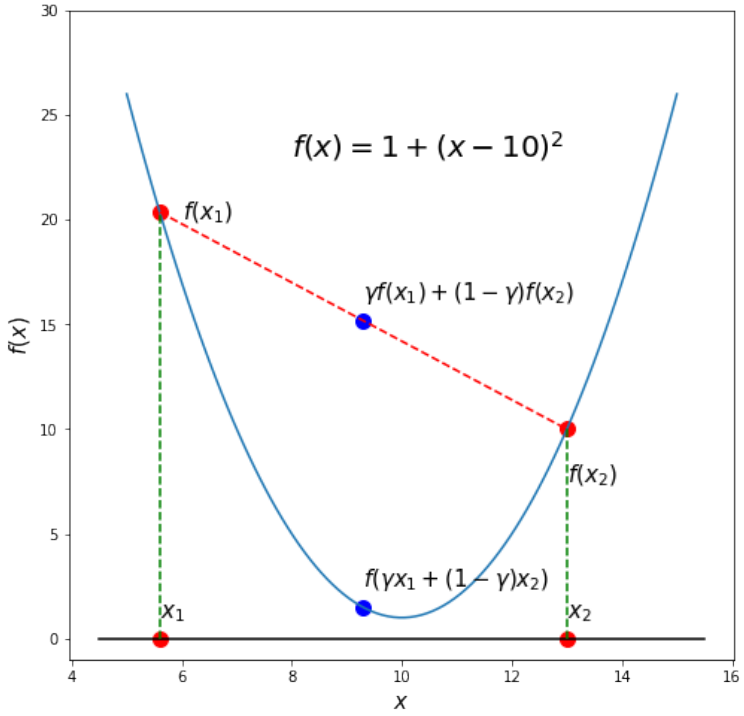


Figura 4.- Ejemplo de función convexa. Una función convexa es aquella en la cual para cualquier par de puntos x_1 y x_2 en el dominio de la función siempre se cumple que la combinación lineal de los valores de la función evaluada en estos dos puntos es mayor que la función evaluada en un punto que es la combinación lineal de estos dos puntos. Es decir, $\gamma f(x_1) + (1 - \gamma) f(x_1) \geq f(\gamma x_1 + (1 - \gamma) x_2))$, donde $\gamma \in [0, 1]$. En este caso es fácil ver que esta condición se satisface, pero en la mayoría de los casos suele ser mucho más complicado. Imagen creada por el autor utilizando Python.

En problemas de optimización no convexos obtener el mínimo global es complicado utilizando algoritmos clásicos de optimización (Dutta, 2016). Para esta clase de problemas existen un conjunto de técnicas de optimización globales conocidas como metaheurísticas (Hedar y Ahmed, 2004).

Los algoritmos metaheurísticos de optimización, o metaheurísticas, pueden tratar con problemas no convexos, no lineales y multimodales (Figura 5) sujetos a restricciones lineales y no lineales con variables de decisión continuas y discretas (Cuevas y cols., 2019). Más importante aun, en problemas de optimización donde la función objetivo no tiene una expresión analítica, sino que es el resultado de simulaciones, los algoritmos metaheurísticos pueden ser empleados de manera eficiente (Talbi, 2009).

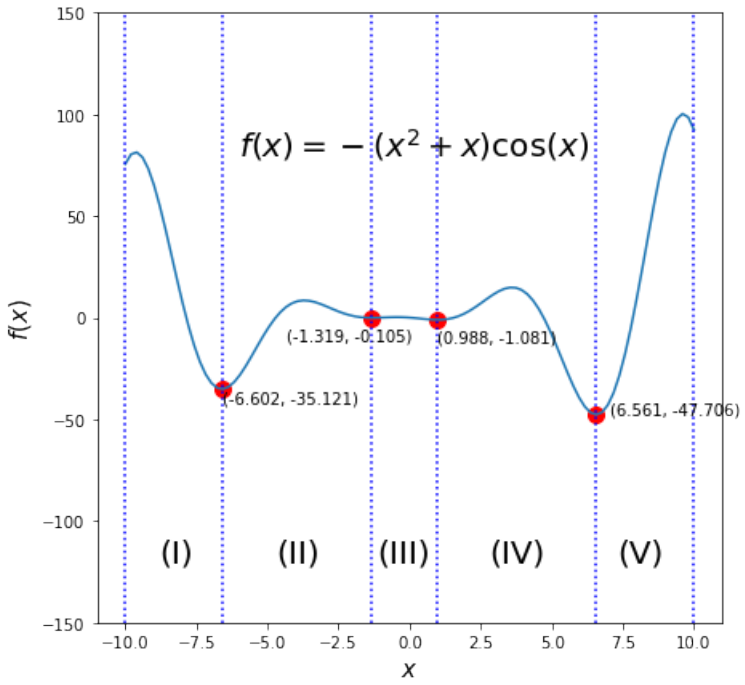


Figura 5.- Ejemplo de función multimodal con tres mínimos locales y un mínimo global en el intervalor $-10 < x < 10$. En esta clase de funciones emplear un método de optimización clásico como el método del gradiente suele llevar a un mínimo local en lugar de un mínimo global. En la figura se observa que la función tiene cuatro mínimos en el intervalo. Estos mínimos dividen el intervalo en cinco subintervalos. Si se emplea el método del gradiente y se comienza en cualquier intervalo diferente del intervalo IV o V se llegará a un mínimo local. Imagen creada por el autor utilizando Python.

Los algoritmos metaheurísticos no requieren hipótesis sobre el problema de optimización ni tampoco información adicional sobre la función objetivo (e.g., su gradiente). El tratamiento de la función objetivo como una caja negra es la más prominente y atractiva característica de los algoritmos metaheurísticos.

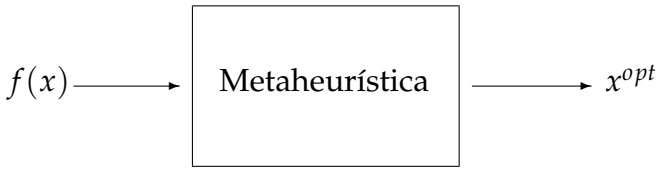


Figura 6.- Representación de un algoritmo metaheurístico como una caja con entradas y salidas. Se puede pensar en una algortimo metaheurístico de optimización como un tipo de caja que no requiere conocimiento detallado sobre el problema de optimización para obtener una solución óptima global. En específico, no requiere información sobre la continuidad de la función objetivo o su gradiente. Tampoco sobre la convexividad de la función objetivo o la región de búsqueda. Imagen creada por el autor utilizando LaTeX.

Como otros métodos de optimización, no garantizan encontrar la solución óptima global de manera exacta. No obstante, la solución que se encuentra suele estar próxima al óptimo global y logran esto incluso si los problemas son bastante complejos de resolver mediante métodos convencionales (Nesmachnow, 2014).

La mayoría de estos métodos se basan en procesos naturales y comportamientos de sistemas biológicos. Por ejemplo, el algoritmo de templado simulado busca imitar el proceso de enfriamiento de acero para obtener estructuras metálicas cristalinas. Por otro lado, los algoritmos genéticos (AGs) imitan el proceso de selección natural de la evolución que da lugar a organismo mejor adaptados a su entorno (Cortez y Cortez, 2014).

Debido a que los métodos metaheurísticos pueden resolver problemas con múltiples funciones objetivo y con restricciones y funciones no lineales, se emplean para encontrar soluciones a un número creciente de problemas complejos del mundo real. Por ejemplo, los AGs se han empleado para la optimización de fermentaciones por lotes alimentados (Oteiza y cols., 2018).

La palabra heurística tiene su origen en la palabra griega antigua heuriskein, que significa el arte de descubrir estrategias nuevas para resolver problemas. El sufijo meta, otra palabra griega, significa “metodología de nivel superior”. El término metaheurístico fue introducido por

F. Glover en el artículo (Glover, 1986).

Un método de búsqueda heurístico comienza con alguna solución, explora todas las soluciones en algún vecindario de esta solución buscando una mejor solución y repite si una mejor solución es encontrada. El procedimiento es un algoritmo iterativo donde cada iteración implica la realización de una búsqueda de una nueva solución que puede ser mejor que la solución encontrada en la iteración anterior (Hillier, 2010). El componente "meta" de una metaheurística se refiere a un nivel de estrategia superior que controla un conjunto de técnicas heurísticas subyacentes con el objetivo de mejorar la capacidad de búsqueda del algoritmo resultante.

Los algoritmos metaheurísticos obtienen conocimiento sobre la estructura del problema de optimización mediante el uso de información generada de soluciones posibles evaluadas anteriormente. Esta información es usada para construir nuevas soluciones candidatas las cuales suelen ser mejores que las soluciones anteriores Cuevas y cols. (2019).

La idea principal en el desarrollo de las metaheurísticas fue proveer una guía para diseñar estrategias generales de optimización independientes del problema. De esta manera, las metaheurísticas no están enfocadas en resolver algún tipo particular de problema. Ellas emplean ideas simples con alta aplicabilidad a una gran variedad de problemas.

De acuerdo con Glover (1997) una metaheurística puede ser considerada como una "estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de esas que son normalmente generadas en una búsqueda de optimización local".

Intensificación y diversificación

Dos componentes principales de las metaheurísticas son la intensificación y la diversificación, o la explotación y la exploración (Figura 7). La diversificación significa generar diversas soluciones para explorar el espacio de búsqueda a una escala global, mientras que la intensificación significa enfocar la búsqueda en una región sabiendo que una solución buena se encuentra en esa región (Lones, 2011). La intensificación asegura que las soluciones van a converger a un óptimo, mientras que la diversificación, mediante aleatorización, permite a la búsqueda escapar de óptimos locales y, al mismo tiempo, incrementar la diversidad de soluciones. Una buena combinación entre estos dos componentes asegura que una solución cercana al óptimo global se alcance (Hedar y Ahmed, 2004).

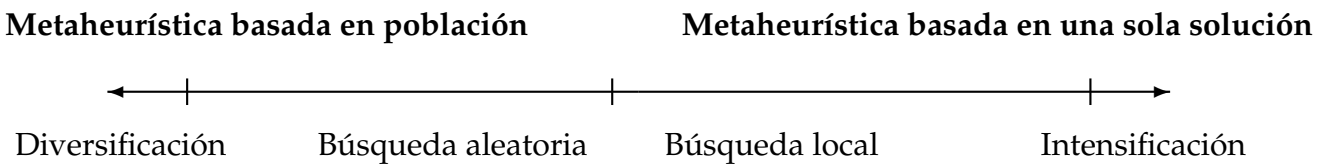


Figura 7.- Dos criterios en conflicto en el diseño de una metaheurística: exploración (diversificación) contra explotación (intensificación). En general, las metaheurísticas basadas en una sola solución están más orientadas a la explotación mientras que metaheurísticas basadas en población están más orientadas a la exploración. Imagen creada por el autor usando LaTeX.

Clasificación de los algoritmos metaheurísticos de optimización

Glover y Laguna (1998) clasifican a las metaheurísticas de acuerdo a tres atributos principales.

- Con y sin memoria adaptativa

  En primer lugar, distinguen entre métodos con y sin memoria adaptativa. Un ejemplo de algoritmo con memoria adaptativa es la búsqueda tabú, la cual usa una lista para guardar soluciones encontradas previamente de manera que no puedan ser seleccionadas durante un determinado número de iteraciones subsecuentes. Los AGs y el algoritmo de templado simulado son vistos como sin memoria adaptativa, aunque los AGs sí retienen información del pasado a través de la población.

- Deterministas y estocásticos

  En segundo lugar, distinguen entre algoritmos que emplean procedimientos deterministas para generar nuevas soluciones y algoritmos que emplean procedimientos estocásticos. Una metaheurística determinista resuelve el problema de optimización mediante la toma de decisiones deterministas, es decir, operadores en el algoritmo siguen reglas que no consideran variables aleatorias o distribuciones de probabilidad. La búsqueda tabú es un ejemplo de algoritmo determinista. En las metaheurísticas estocásticas algunas reglas aleatorias son aplicadas durante la búsqueda. Por ejemplo, una solución de la iteración actual puede llegar a formar parte de las soluciones de la siguiente iteración con una determinada probabilidad. Los AGs y el algoritmo de templado simulado son ejemplos de algoritmos estocásticos. En algoritmos deterministas usar la misma solución inicial

conducirá a la misma solución final, mientras que en un algoritmo estocástico diferentes soluciones pueden ser obtenidas de la misma solución inicial.

- Una solución y una población de soluciones

  Finalmente, distinguen entre los algoritmos que emplean una solución única o una población de soluciones durante el proceso de búsqueda. Algoritmos basados en una solución única, como el algoritmo de templado simulado, manipulan y transforman una sola solución durante la búsqueda. En los algoritmos basados en poblaciones, como los AGs, una población de soluciones va evolucionando durante la ejecución del algoritmo. Estas dos familias tienen características complementarias. Las metaheurísticas basadas en una solución están orientadas a la explotación, tienen la capacidad de intensificar la búsqueda en regiones locales. Las metaheurísticas basadas en poblaciones están orientadas a la exploración, permiten una mayor diversificación en el espacio de búsqueda.

2.3.1. Algoritmos genéticos en la optimización de procesos

Los algoritmos genéticos fueron inventados por John Holland en la universidad de Michigan en 1970s (Lones, 2011). Los AGs están basado en los principios de la selección natural. Los elementos básicos de la selección natural: reproducción, cruzamiento y mutación, son usados en el procedimiento. Los AGs simulan los procesos biológicos que permiten a las consecutivas generaciones en una especie adaptarse a su ambiente. El proceso de adaptación es aplicado principalmente a través de la herencia de padres a hijos y a través de la supervivencia de los más aptos (Rao, 2019).

Los AGs comienzan con un conjunto inicial de soluciones al que se le llama población. A las soluciones en una población se les llama individuos o cromosomas. En la presentación original de los AGs los individuos en la población eran representados mediante números binarios. Actualmente existen versiones de AGs donde se representa a los individuos como variables de valor real. En la representación de número binario cada individuo, o cromosoma, consiste en un número fijo de bits (0 o 1) los cuales son llamados genes.

Para evaluar y clasificar cromosomas en una población una función fitness debe ser definida. El valor de la función fitness evaluada en un determinado cromosoma se conoce como el fitness

de ese cromosoma. Tres operadores deben ser especificados para construir la estructura completa de un AG: operadores de selección, cruce y mutación. Estos operadores son aplicados de manera secuencial por un número de iteraciones determinado.

Función fitness

La función fitness permite conocer qué tan buena es una solución. Debe ser establecida de forma tal que una buena solución le corresponda un valor grande de la función fitness. Si se busca maximizar la función objetivo, en algunos casos es posible establecer la función objetivo directamente como la función fitness. Si se busca minimizar la función objetivo, la función fitness debe ser planteada de tal forma que el mínimo de la función objetivo corresponda al máximo de la función fitness. Por ejemplo, si se quiere minimizar la función $f(x)$ la función fitness puede ser escrita de la siguiente manera.

$$fitness(x) = \frac{1}{1 + f(x)} \tag{17}$$

De esta forma, el valor óptimo de x que genera el valor máximo de $fitness(x)$ da lugar al valor mínimo de $f(x)$. El aspecto importante que hay que considerar es que la mejor solución en el AG tiene que estar asociada con el mayor valor de la función fitness.

Codificación

La codificación de las soluciones se refiere a la forma en la que las soluciones en la población son representadas. Existen dos principales tipos de representaciones en un AG: la codificación como números binarios y la codificación como números reales. La implementación del algoritmo con codificación binaria es más sencilla. Sin embargo, la codificación real puede ser necesaria si el problema tiene un vector de tipo real como solución. Es tal caso, es posible programar un AG que trabaje completamente con vectores de tipo real o emplear una función que permita transformar vectores de tipo binario en vectores de tipo real.

Operador de selección

La selección es el primer operador aplicado a la población para seleccionar a los cromosomas con valores fitness altos y formar la población de padres. El operador de selección se usa para escoger a los cromosomas por arriba del promedio de la población actual e insertar sus múltiples copias en la población de padres basado en un procedimiento probabilístico (Rao, 2019).

Selección proporcional

El operador de selección más común se denomina selección proporcional al fitness, también conocido como selección de ruleta. En este método se seleccionan a los cromosomas en proporción a su fitness. Si $fitness(i)$ denota el fitness del cromosoma i en la población de tamaño N, la probabilidad de seleccionar al cromosoma i para la población de padres se puede calcular como.

$$p(i) = \frac{fitness(i)}{\sum_{j=1}^{N} fitness(j)} \tag{18}$$

donde: $p(i)$ es la probabilidad de seleccionar al cromosoma i.

Estas probabilidades son usadas para calcular la probabilidad acumulada de cada cromosoma, $p_a(i)$, mediante la adición de las probabilidades del cromosoma 1 hasta el cromosoma i.

$$p_a(i) = \sum_{j=1}^{i} p(j) \tag{19}$$

Se puede apreciar que el último cromosoma en la población tendrá una probabilidad acumulada, $p_a(N)$, igual a 1. Para la selección de un cromosoma se asocia el rango de probabilidad acumulada, $p_a(i) - p_a(i-1)$, con el cromosoma i en la población. Posteriormente, se genera un número aleatorio entre 0 y 1 y se escoge el cromosoma que tiene el rango de probabilidad acumulado en donde se encuentra el número aleatorio generado.

La implementación del proceso de selección proporcional se puede entender si se imagina una rueda de ruleta con su circunferencia dividida en segmentos, uno para cada cromosoma de la población, con las áreas de los segmentos proporcionales al *fitness* de los cromosomas. Si se gira la rueda de ruleta N veces (N siendo el tamaño de la población) y seleccionando, cada vez, el cromosoma elegido por el apuntador de la rueda de ruleta, se obtiene una población de padres de tamaño N (Figura 8).

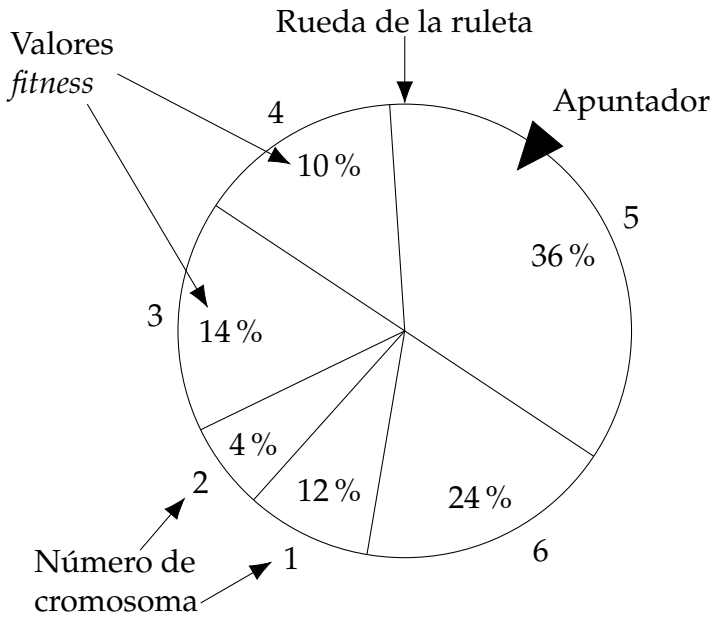


Figura 8.- Esquema de algoritmo de selección proporcional en los AGs. Cada cromosoma en la población tiene un espacio en la ruleta que es proporcional a su *fitness*. Por lo tanto, los cromosomas con un valor *fitness* alto tienen mayor probabilidad de ser seleccionados. Imagen creada por el autor usando LaTeX

.

Mediante este procedimiento el cromosoma con el mayor valor fitness será seleccionado más frecuentemente para formar parte de la población de padres debido a su mayor rango de probabilidad acumulada. Por esta razón, cromosomas con valores fitness altos en la población probabilísticamente obtienen más copias en la población de padres.

Selección de torneo

Existen algunos problemas con el método de selección proporcional. Este método supone que la magnitud del valor fitness de un cromosoma significa algo importante. En muchas ocasiones, se escoge la función fitness de tal manera que un cromosoma con un valor fitness mayor a otro cromosoma sea mejor, sin importar la magnitud de esta diferencia (Lones, 2011). Además, si un cromosoma posee un valor fitness bastante superior al resto, existe la posibilidad de que varias copias de este cromosoma sean insertadas en la población de padres, con lo que se reduce la diversidad de soluciones de forma prematura.

En la selección por torneo una muestra de cromosomas es tomada de forma aleatorio de la

población, este muestreo de soluciones se puede realizar con o sin remplazo. Esta muestra toma parte en un "torneo". En este torneo los cromosomas en la muestra compiten para seleccionar el cromosoma que formará parte de la población de padres. El cromosoma ganador es el que posee mayor fitness (Figura 9). El proceso se repite N veces para completar la población de padres. El torneo se puede realizar tomando una muestra de solo dos cromosomas (torneo binario) o puede ser generalizado a un número arbitrario de tamaño t, llamado tamaño del torneo (Blickle y Thiele, 1996).

La selección de torneo puede además ajustar la presión de selección para adaptarse a diferentes problemas. La presión de selección se incrementa mediante el aumento del tamaño de torneo, t. Si el tamaño de torneo es grande, los cromosomas con menor fitness tienen menor probabilidad de ser seleccionados, debido a que en una muestra grande de soluciones existe un mayor número de cromosomas que los pueden superar. Cuando el tamaño del torneo es uno, el proceso de selección es equivalente a la selección aleatoria, en cambio, cuando el tamaño del torneo es considerablemente superior al tamaño de la población, existe una gran probabilidad de elegir al mejor cromosoma en cada torneo, con lo que la selección se vuelve determinista (Miller y cols., 1995).

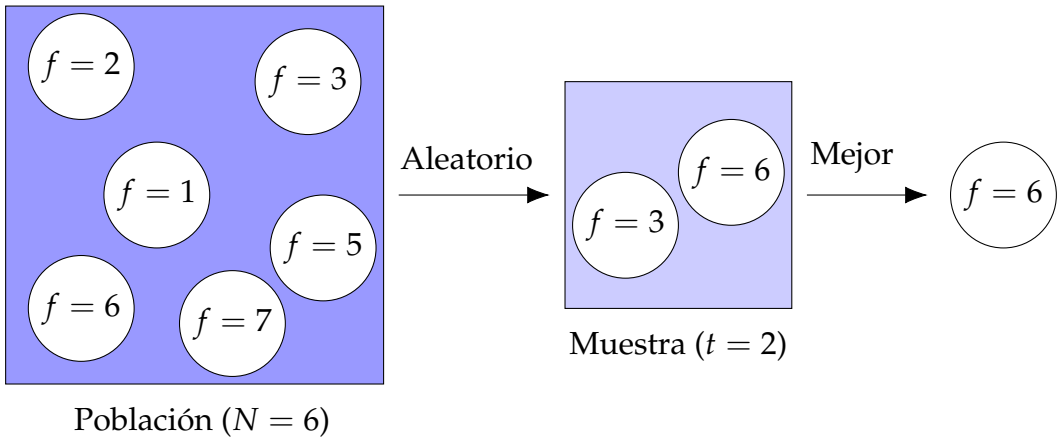


Figura 9.- Esquema de algoritmo de selección de torneo de los AGs. En una población de N individuos se toma una muestra de individuos de tamaño t. Se escoge el individuo con mayor *fitness* en esta muestra para forma parte de la población de padres. El proceso se repite N veces para completar la población de padres. Imagen creada por el autor usando LaTeX.

Es importante notar que no se forman nuevos cromosomas durante la etapa de selección, solo se copian los mejores cromosomas existentes en la población actual a la población de padres.

El operador de selección garantiza que cromosomas con valores fitness altos sobrevivan y se reproduzcan, mientras que cromosomas menos aptos mueran.

Operador de Cruce

Después de la selección el operador de cruce es aplicado sobre la población de padres. El propósito del cruce es crear nuevos cromosomas mediante el intercambio de información entre cromosomas en la población de padres. En la mayoría de los operadores de cruce, dos cromosomas son escogidos aleatoriamente en la población de padres y partes de estos son intercambiados para generar nuevos cromosomas que se denominan cromosomas hijos (Rao, 2019).

El cruce de cromosomas padres para generar cromosomas hijos no siempre da lugar a mejores cromosomas en términos de su valor fitness. Por esta razón, es conveniente preservar algunos cromosomas de la población de padres en la siguiente población. El operador de cruce se aplica a pares de padres con una probabilidad, p_c, que suele estar entre 0.6 y 0.8.

El operador de cruce juega el papel principal en los AGs, así que definir un operador de cruce adecuado es fundamental para lograr un buen resultado (Hedar y Ahmed, 2004).

Cruce de un solo punto

En el método de cruce de un solo punto se elige un sitio de cruce para dividir dos cromosomas padres en dos partes. Las partes que se obtienen son intercambiadas entre los padres para generar dos nuevos cromosomas hijos. En la Figura 10 se muestra un ejemplo de cómo se podría realizar el cruce entre cromosomas binarios.

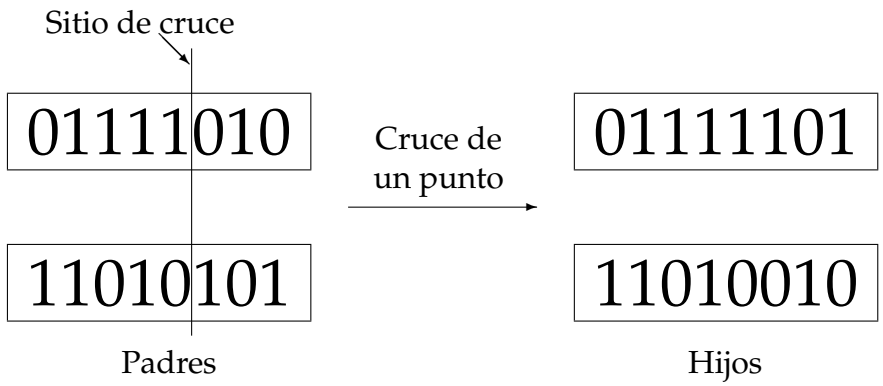


Figura 10.- Esquema de algoritmo de cruce de un solo punto en los AGs. Se escoge un sitio en la cadena de bits y se intercambian las partes de los padres en ambos lados del sitio para generar dos hijos. En este caso el sitio de cruce está entre el bit 5 y 6. Una vez que se escoge el sitio de cruce, los bits a la izquierda y derecha de este sitio son intercambiados entre los padres para generar dos cromosomas hijos. Imagen creada por el autor usando LaTeX.

Cruce uniforme

El problema con el cruce de un solo punto reside en la posible vinculación entre los elementos de un cromosoma. La probabilidad de separar el primer y último elemento de un cromosoma es alta mediante cruce de un solo punto, dado que casi cualquier sitio de cruce logrará esto. Por el contrario, la probabilidad de separar dos elementos consecutivos es baja, dado que se requiere un sitio específico de cruce para lograr esto. Si la organización del cromosoma es tal que los elementos en los extremos deben trabajar juntos, se estaría rompiendo constantemente buenos pares de individuos que se generan en la búsqueda. Es posible tratar a todos los elementos de un cromosoma de forma justa con respecto a la vinculación si se realiza el cruce en cada posición de forma independiente de las demás posiciones. En el cruce uniforme se recorren los elementos en un par de cromosomas padres y se realiza un intercambio de los elementos entre los padres con una determinada probabilidad (Figura 11).

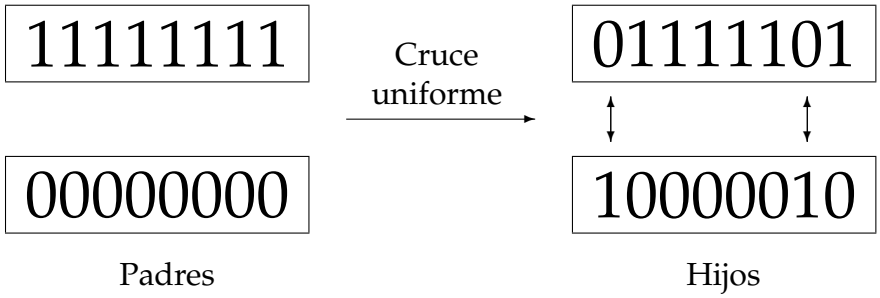


Figura 11.- Esquema de algoritmo de cruce uniforme en los AGs. Cada bit en los cromosomas padres tiene una probabilidad, p_c, de ser intercambiado. Es este caso se observa que el primer y séptimo bits han sido intercambiados. Imagen creada por el autor usando LaTeX.

Operador de mutación

La mutación es el último operador que se aplica en el algoritmo genético. El operador de mutación se aplica a la población resultado del operador de cruce. El propósito del operador de mutación es realizar cambios aleatorios en los genes de la descendencia para aumentar la diversidad de la población. Este operador permite que los AGs no queden atrapados en óptimos locales.

El operador de mutación no se presenta siempre, sino que ocurre con cierta probabilidad baja, dado que conlleva el riesgo de dañar el desempeño de cualquier individuo al que se le aplique. Si la tasa de mutación aumenta excesivamente, el AG se convertirá en el equivalente a una búsqueda aleatoria (Wirsansky, 2020).

Mutación de un solo bit

En la mutación de un solo bit, cuando el cromosoma es de tipo binario, un sitio de mutación a lo largo del cromosoma es seleccionado de forma aleatoria. El dígito en la posición seleccionada es "volteado" de 0 a 1 o de 1 a 0 con una probabilidad p_m. Este proceso solo ocurre una vez en cada individuo, por lo cual solo uno de sus bits puede llegar a ser modificado.

Mutación de bit a bit

En la mutación de bit a bit cada bit de cada cromosoma tiene la misma probabilidad, p_m, de ser modificado. A diferencia de la mutación de un solo bit, en este tipo de mutación es posible mutar varios de los bits en un cromosoma.

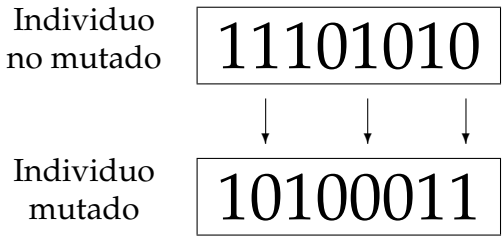


Figura 12.- Esquema de algoritmo de mutación de bit a bit en los AGs. En este ejemplo se aprecia como tres de los bits en el individuo original han sido modificados con lo que se genera un nuevo individuo mutado. Este esquema de mutación difiere de la mutación de un solo bit, donde solo es posible modificar un bit en cada individuo. Imagen creada por el autor usando LaTeX.

Una vez que se ha aplicado el operador de mutación se dice que una generación en el algoritmo ha sido completada. La nueva población de individuos que ha resultado de la aplicación de los operadores de selección, cruce y mutación debe poseer un valor fitness promedio superior al de la población que le precedió. Esta población es sometida de nuevo a los operadores de selección, cruce y mutación por un número determinado de iteraciones.

Terminación del algoritmo genético

Los operadores de selección, cruce y mutación se aplican por un número determinado de iteraciones, que se denomina número de generaciones, N_{gen}. Normalmente, se emplea un criterio para determinar si el algoritmo debe terminar antes de que se concluyan todas las generaciones establecidas al inicio. Este criterio puede ser un valor máximo de la función fitness que se ha alcanzado o un número específico de generaciones sin que se presentes cambios en la función fitness. Un diagrama con los diferentes pasos que se realizan en la ejecución del algoritmo se muestra la Figura 13.

Los AGs son métodos estocásticos, por lo que no se obtienen las mismas soluciones aun cuando el algoritmo se ejecute en las mismas condiciones. No obstante, la solución a la que llega suele estar siempre cerca al óptimo global.

Para un buen desempeño del AG es necesario establecer de forma adecuada los parámetros del algoritmo como son: tamaño de población, N_{pob}; número de generaciones, N_{gen}; probabilidad

de cruce, p_c y probabilidad de mutación, p_m. Estos parámetros dependen de la naturaleza del problema que se resuelve. Por ejemplo, si el problema es altamente no convexo con muchos óptimos locales, podría ser necesario un valor elevado de número de generaciones y tamaño de población.

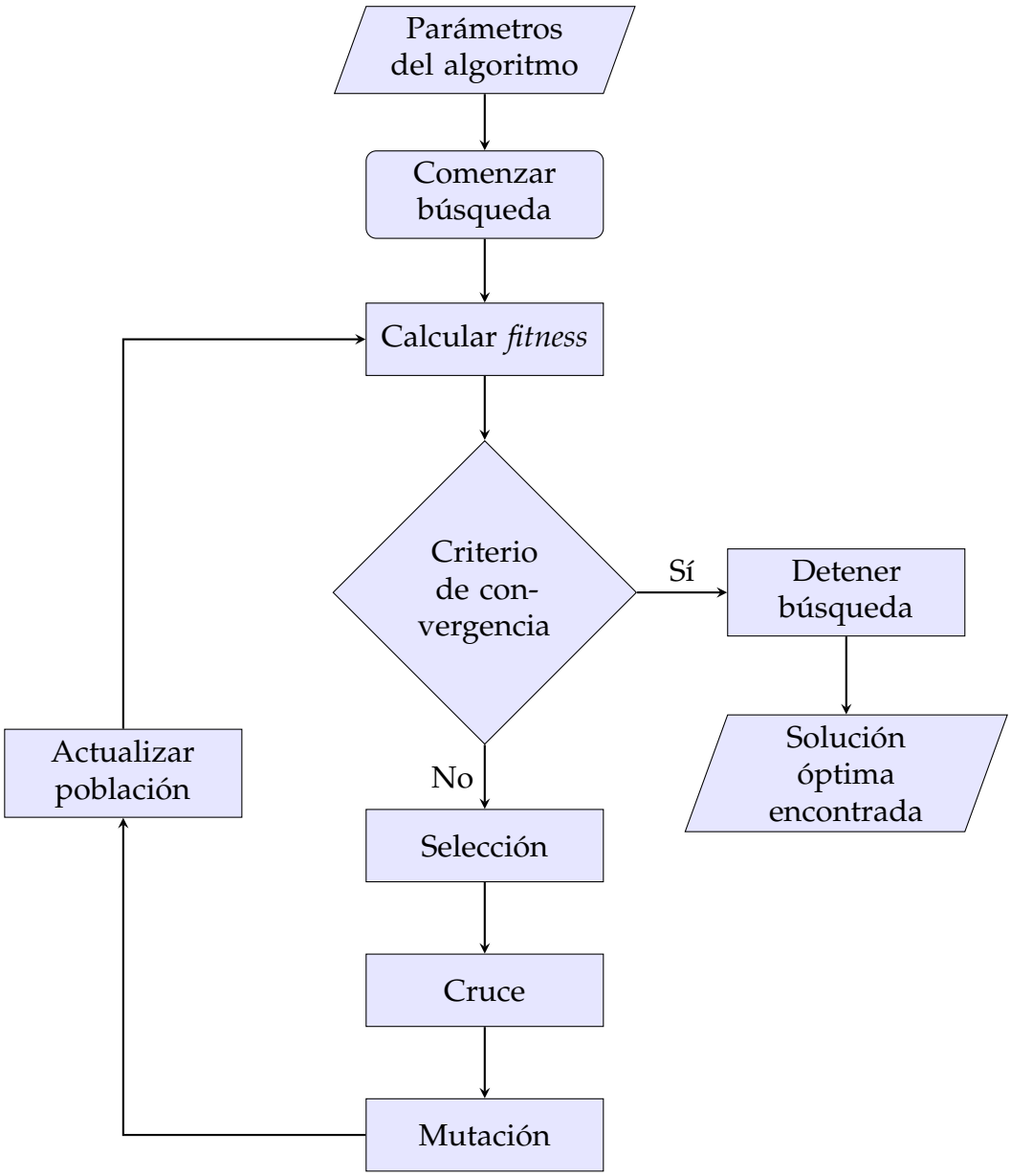


Figura 13.- Diagrama de flujo del funcionamiento de un AG. Al inicio de la búsqueda se deben introducir los parámetros del algoritmo y la función *fitness*. Posterior a esto, el algoritmo comienza las búsqueda mediante la generación de una población de cromosomas aleatoria. A continuación se obtiene el *fitness* de esta población de cromosomas y se prueba el criterio de convergencia. Si el criterio se satisface, el algoritmo se detiene y regresa a la población actual con su valor *fitness*. Si no se satisface, la población actual se utiliza para generar una nueva población mediante los operadores genéticos. La generación de nuevas poblaciones a través de los operadores genéticos continua hasta que el criterio de convergencia se satisface.

2.4. Necesidad de usar programas computacionales para la optimización y simulación de procesos de fermentaciones

Los procesos de fermentación son procesos fermentativos que implican el uso de microorganismos o enzimas para transformar materias primas en productos de interés industrial, como etanol, ácido láctico o antibióticos (Oliveira y cols., 2017). Estos procesos requieren un diseño y una operación adecuados para lograr una alta productividad y calidad del producto final. Para ello, es necesario realizar un análisis detallado de los datos experimentales obtenidos mediante ensayos en laboratorio o planta piloto, así como una simulación y optimización del proceso a escala industrial.

El análisis de los datos experimentales implica el uso de técnicas estadísticas para evaluar la influencia de las variables del proceso sobre el rendimiento y la cinética del mismo. Además, se requiere estimar los parámetros cinéticos que describen el comportamiento del sistema biológico mediante métodos numéricos o algoritmos genéticos. La simulación y optimización del proceso consiste en modelar matemáticamente el sistema físico-químico-biológico y resolver las ecuaciones que lo gobiernan mediante métodos numéricos o analíticos. Así, se puede predecir el comportamiento del proceso bajo diferentes condiciones operativas y buscar las mejores soluciones posibles.

Para realizar estas tareas se necesita el apoyo de programas computacionales que faciliten los cálculos necesarios y permitan visualizar los resultados obtenidos. Existen diversos programas comerciales que ofrecen estas funcionalidades, tales como `MATLAB`, `SuperPro Designer` y `Aspen Plus`. Estos programas cuentan con una amplia gama de herramientas para la simulación y optimización de procesos químicos e incluyen bases de datos termodinámicas, modelos cinéticos y módulos específicos para procesos fermentativos. Sin embargo, estos programas tienen algunas limitaciones, como el alto costo de las licencias, la dificultad para adaptarlos a las necesidades particulares de cada proceso o la dependencia de proveedores externos.

Por esta razón, muchos grupos de investigación han optado por desarrollar sus propios programas computacionales basados en software libre o código abierto. Estos programas tienen la ventaja de ser gratuitos, flexibles y personalizables según las preferencias del usuario. Además, se benefician del trabajo colaborativo de una gran comunidad de desarrolladores e investigadores que contribuyen a mejorarlos constantemente. Algunos ejemplos de estos programas son

`Python` y `R`, que son lenguajes de programación multipropósito que pueden usarse para análisis estadístico, simulación numérica, optimización e incluso desarrollo web. Estos lenguajes cuentan con numerosas librerías especializadas en diferentes áreas del conocimiento que facilitan su uso e implementación.

En conclusión, el uso de programas computacionales es una necesidad en la investigación de procesos fermentativos dado que permite realizar un análisis más profundo y riguroso de los datos experimentales así como una simulación y optimización más eficiente del proceso a escala industrial. Los programas comerciales ofrecen soluciones integradas y robustas pero también presentan inconvenientes como el costo, la rigidez y la dependencia. Los programas basados en software libre o código abierto ofrecen soluciones alternativas y ventajosas como la gratuidad, la flexibilidad y la colaboración. Por lo tanto, es importante conocer las características y capacidades de cada tipo de programa para elegir el más adecuado según las necesidades y objetivos de cada proyecto.

2.4.1. Programación en Python

Python es un lenguaje de programación interpretado, multiparadigma y de alto nivel que se caracteriza por su sintaxis clara y legible (*Introducción al lenguaje de programación Python (3º Edición)*, s.f.). Fue creado por Guido van Rossum a finales de la década de 1980 como un sucesor del lenguaje ABC (*Introducción al lenguaje de programación Python*, s.f.). Python soporta tanto la programación orientada a objetos como la programación imperativa y funcional (*Introducción al lenguaje de programación Python (3º Edición)*, s.f.). Además, ofrece estructuras de datos integradas, como listas, tuplas, conjuntos y diccionarios, y permite la definición de módulos, clases y excepciones (*Introducción — documentación de Python - 3.11.2*, s.f.). Python cuenta con una gran variedad de librerías que amplían sus funcionalidades en áreas como el aprendizaje automático, el análisis numérico y estadístico, la visualización de datos y el desarrollo web (*Introducción a la programación en Python 1: Aprendiendo a programar en Python - Coursera*, s.f.). Python es un lenguaje de código abierto que se rige por la licencia Python Software Foundation License (*Introducción — documentación de Python - 3.11.2*, s.f.).

3. Metodología

3.1. Estimación de parámetros cinéticos mediante la implementación de algoritmos genéticos

1. Descripción del problema y modelado del proceso.
    - Generación de datos simulados utilizando modelos de Monod y de inhibición por sustrato.
    - Adición de ruido normal para simular incertidumbre en datos experimentales.
    - Integración numérica con algoritmo Runge-Kutta de cuarto orden.
    - Definición de la función objetivo: minimización de la suma de los cuadrados del error (SSE).
    - Formulación de balances de materia para biomasa, sustrato y producto.
2. Implementación de algoritmos genéticos.
    - Configuración de parámetros del algoritmo genético.
    - Definición de los rangos de búsqueda para las variables.
    - Realización de múltiples estimaciones para verificar la precisión.
3. Análsis de resultados.
    - Comparación de valores obtenidos con valores originales.
    - Presentación de valores estadísticos y análisis de intervalos de confianza.
4. Integración en la aplicación web.
    - Evaluar la eficacia de los algoritmos genéticos para la estimación de parámetros.
    - Planificación para la inclusión del código y modelos en la aplicación web.

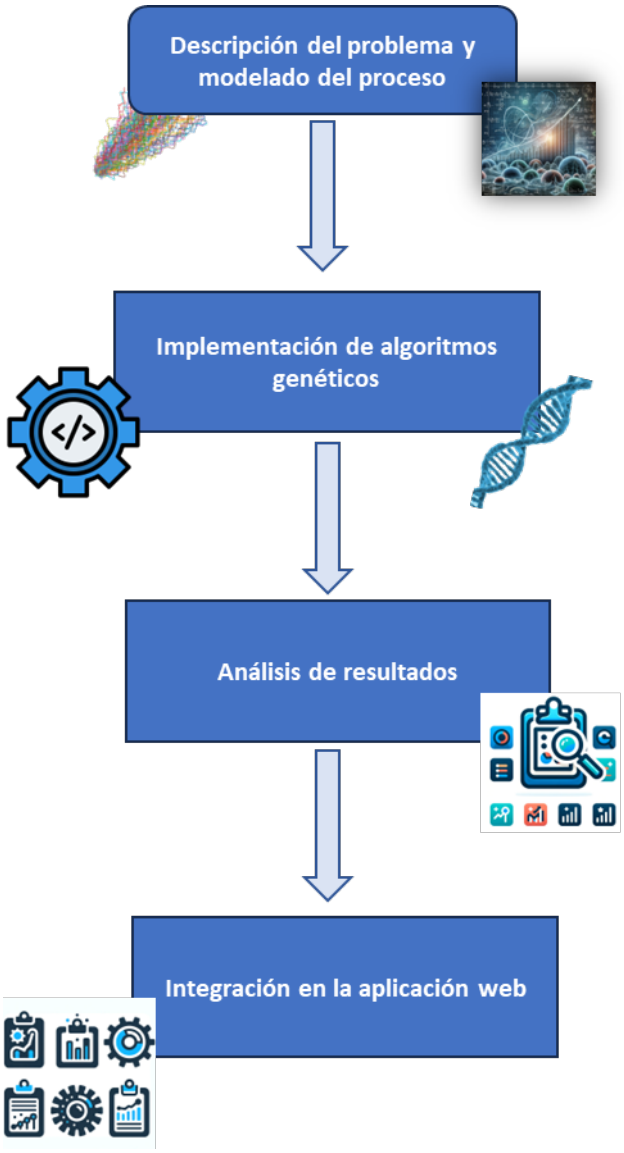


Figura 14.- Esquema de la metodología para la estimación de parámetros cinéticos.

3.2. Optimización del flujo de alimentación mediante la implementación de algoritmos genéticos

1. Descripción del problema y modelado del proceso.
    - Uso de un biorreactor tipo lote-alimentado para evitar efectos inhibitorios.
    - Objetivo de determinar el flujo óptimo de alimentación para maximizar la producción de biomasa.
    - Formulación de la función objetivo y restricciones para el flujo y volumen del biorreactor.
    - Desarrollo de balances de materia para biomasa, sustrato y producto.
2. Implementación de algoritmos genéticos.
    - Configuración de parámetros del algoritmo genético.
    - Evaluación de estrategias de flujo constante y flujo variable.
    - Revisión de parámetros de operación y condiciones iniciales.
3. Análisis de resultados.
    - Realización de estimaciones para determinar el flujo óptimo.
    - Comparación de la producción de biomasa con diferentes estrategias.
    - Análisis de la relación entre producción de biomasa y flujo de alimentación.
4. Consideraciones finales y futuras direcciones.
    - Evaluar la eficiencia de los algortimos genéticos para la optimización del flujo.
    - Sugerencias para investigaciones futuras y mejoras en el modelo.

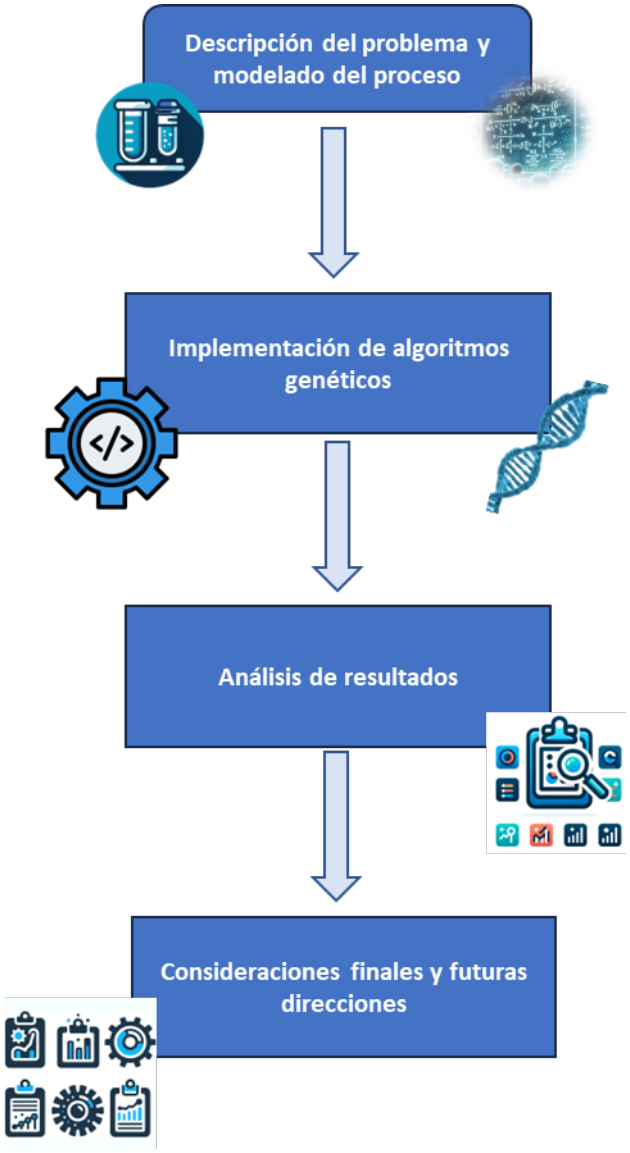


Figura 15.- Esquema de la metodología para la optimización del flujo de alimentación.

3.3. Construcción de la red neuronal artificial para simular procesos dinámicos de fermentación

1. Descripción del problema y modelación del proceso.
    - Revisión de configuraciones de redes neuronales para simular procesos dinámicos.
    - Utilización de trayectorias dinámicas para entrenar la red.
    - Entradas de la red: valores de las variables de estado en un instante temporal.
    - Salidas de la red: valores de las variables de estado en el siguiente instante temporal.
2. Generación de trayectorias de entrenamiento y validación.
    - Uso de valores de las variables biomasa, sustrato y producto en distintos instantes temporales para el entrenamiento.
    - Empleo de modelos mecanísticos para generar datos de entrenamiento y validación.
    - Generación de 100 condiciones iniciales aleatorias utilizando el método de muestreo hipercubo latino.
    - Simulación del sistema dinámicos usando el método numérico de Runge-Kutta de cuarto orden.
3. Estructura de la red neuronal.
    - Selección de la estructura basada en la revisión de la literatura y experimentos computacionales.
    - Uso de 8 capas ocultas con 32 neuronas cada una y función de activación.
    - Configuración de neuronas en la capa de entrada y salida igual al número de varibles del problema.
4. Función de pérdida.
    - Adopción de MSE (Mean Squared Error) como la función de pérdida.
    - Evaluación de la función de pérdida comparando la trayectoria predicha con la trayectoria real.

5. Algoritmo de entrenamiento.
    - Uso del algoritmo de optimización Adam para minimizar la función de pérdida.
6. Configuración de la red neuronal y experimentación con diferentes configuraciones.
    - Propuesta de tres configuraciones de red: Escalador de tiempo directo, Residual y Numérico.
    - Experimentación con cada configuración para evaluar su eficiencia en simular el proceso de fermentación.
7. Análisis de resultados.
    - Comparación de la eficiencia y efectividad de las tres configuraciones propuestas.
    - Evaluación de los modelos en base a su capacidad de predecir trayectorias de validación.
    - Selección de la configuración más adecuada para futuras simulaciones y análisis de optimización.
8. Consideraciones finales y futuras direcciones.
    - Evaluar aplicabilidad y precisión de las redes neuronales en la simulación de procesos dinámicos.
    - Planificación para la inclusión de la red neuronal más eficiente en la simulación y optimización de procesos de fermentación.

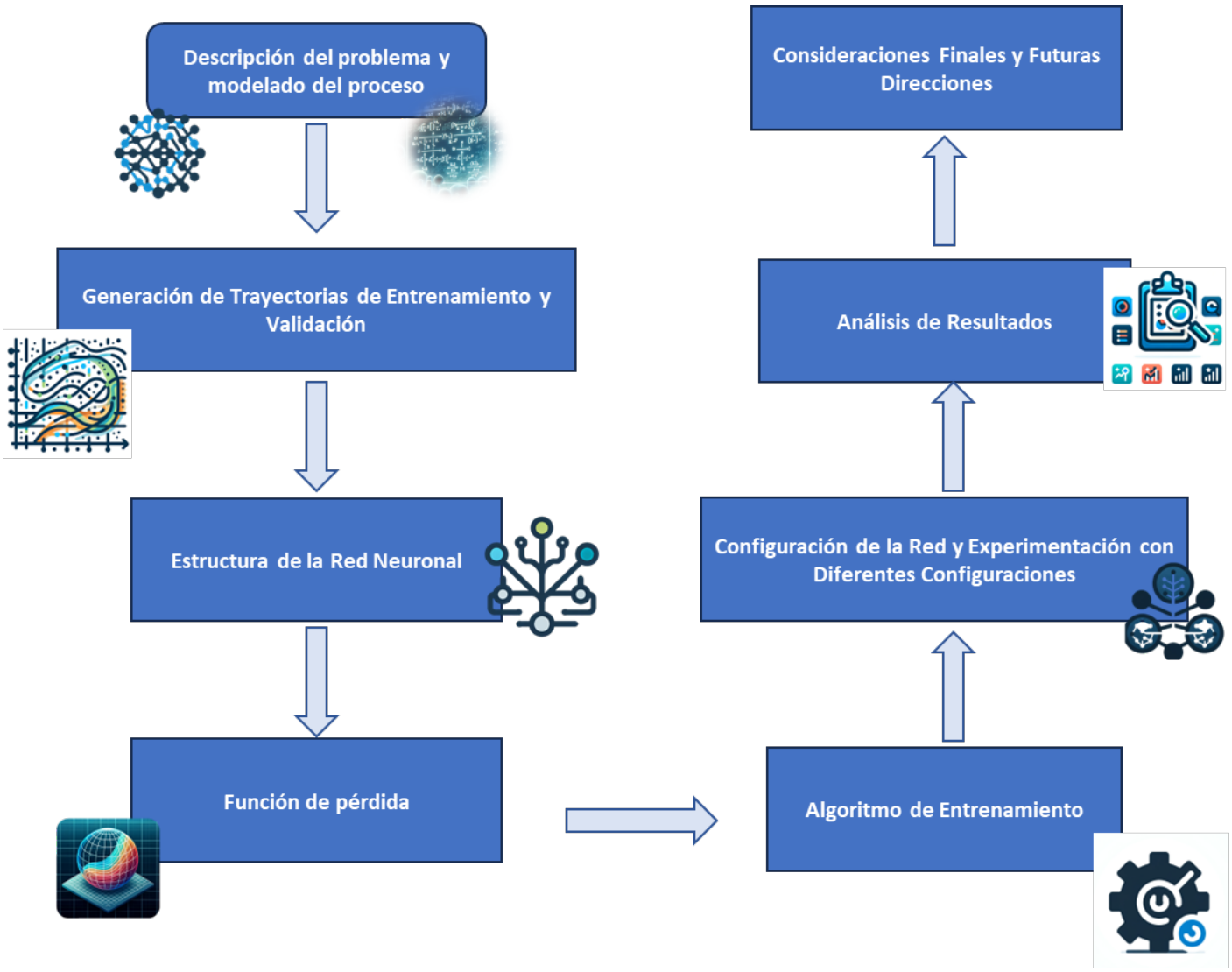


Figura 16.- Esquema de la metodología para la construcción de la red neuroanl artificial.

3.4. Construcción de plataforma web para simulación y análisis de procesos fermentativos

1. Recopilación y análsis de requerimientos.

   En esta fase inicial, se realiza un análisis detallado de los requerimientos funcionales y no funcionales de la aplicación web. Se identifican las necesidades de los usuarios finales y los objetivos técnicos del proyecto. Se contempla la integración de funciones para el ingreso y análisis de datos experimentales, la visualización de resultados y la gestión de cuentas de usuario.

2. Selección de la pila tecnológica.

   Se elige una pila tecnológica que respalde los objetivos de la aplicación, considerando tanto el desarrollo del *frontend* como del *backend*. Se evalúan lenguajes de programación, *frameworks*, bibliotecas y sistemas de bases de datos que ofrezcan robustez, seguridad y escalabilidad.

3. Diseño de la arquitectura de la aplicación web.

   Se define una arquitectura de sistema modular que facilite el mantenimiento y la expansión futura. El diseño contempla la separación del *frontend* y el *backend*, la comunicación a través de una *API RESTful* y la estructura de una base de datos coherente con lo requisitos de los datos experimentales.

4. Establecimiento de plazos.

   Se establece un cronograma de proyectos que detalla las etapas de desarrollo, pruebas e implementación. Los plazos se ajustan a los recursos disponibles y se establecen hitos para seguimiento.

5. Diseño de la interfaz de usuario.

   Se desarrolla el diseño de la interfaz de usuario centrado en la usabilidad y la accesibilidad. Se crean *wireframes* y *mockups* que ilustren el flujo de la aplicación, la disposición de los elementos en la pantalla y las interacciones del usuario.

6. Desarrollo de la aplicación web.

   El desarrollo se divide en etapas iterativas y modulares, permitiendo la implementación incremental de funcionalidades. Se utiliza el marco de trabajo Scrum que favorece la adaptación a cambios y mejora continua a partir de la retroalimentación recibida en las pruebas.

7. Realización de pruebas.

   Se llevan a cabo pruebas unitarias, de integración, funcionales, de usabilidad y de seguridad para asegurar la calidad del software. Las pruebas se automatizan en la medida de lo posible para garantizar la eficiencia del proceso.

8. Despliegue a producción.

   La aplicación se implementa en un entorno de producción, donde se realizan ajustes finales y se garantiza su correcto funcionamiento. Se establecen mecanismos de monitorización y se planifica un proceso de despliegue continuo para facilitar actualizaciones futuras.

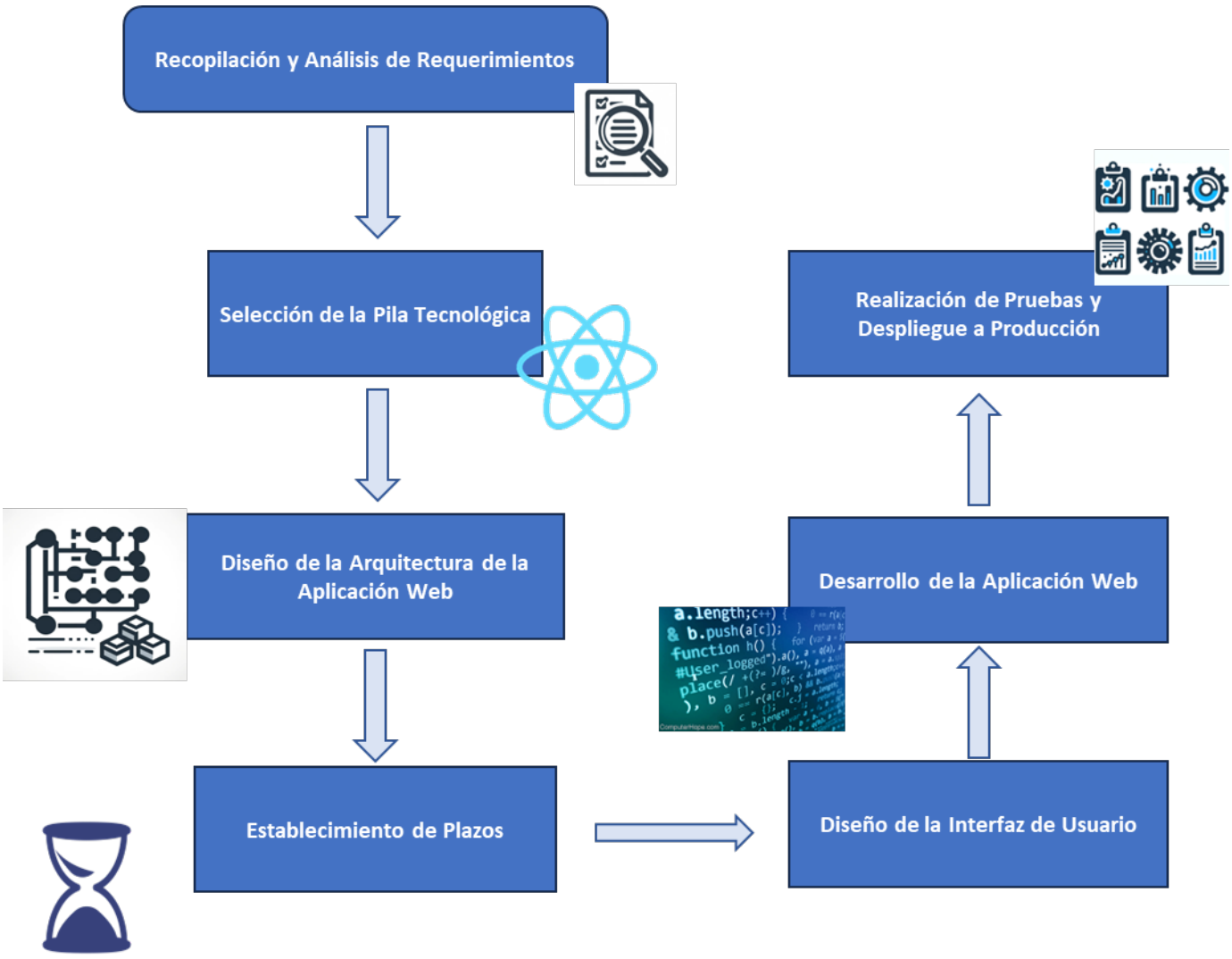


Figura 17.- Esquema de la metodología para el desarrollo de la aplicación web.

4. Resultados

4.1. Estimación de parámetros cinéticos mediante la implementación de algoritmos genéticos

4.1.1. Descripción del problema de optimización para la estimación de parámetros cinéticos

La estimación de los parámetros cinéticos en un proceso de fermentación es fundamental para caracterizar y optimizar este tipo de procesos. Los valores de dichos parámetros proporcionan información acerca de la capacidad de los microorganismos para transformar los sustratos en el biorreactor en los productos y biomasa deseados. De esta manera, se puede seleccionar el microorganismo y sustrato más adecuados para un proceso de fermentación en particular. Además, la obtención de los parámetros cinéticos es necesaria para calibrar modelos y realizar simulaciones en el estudio de estrategias de optimización.

La comparación del grado de ajuste obtenido con diferentes modelos durante la estimación de los parámetros cinéticos permite determinar cuál modelo describe de manera más precisa el proceso. Los datos utilizados para la estimación de parámetros se obtienen con frecuencia a partir de experimentos en biorreactores tipo lote, como el que se ilustra en la Figura 18.

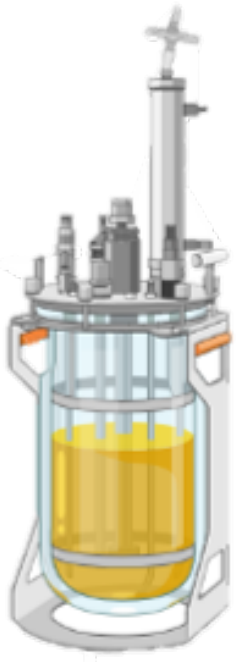


Figura 18.- Esquema del biorreactor tipo lote usado en la estimación de parámetros cinéticos. Imagen creada por el autor usando BioRender.

En esta sección, se propone una metodología para estimar los parámetros cinéticos de procesos de fermentación en biorreactores de operación tipo lote. Además de la estimación del valor de los parámetros, se busca establecer un intervalo de confianza al 95 % con el fin de evaluar la incertidumbre asociada a los valores estimados.

Con el objetivo de determinar la eficacia del algoritmo de optimización en la estimación de los parámetros cinéticos, se generaron datos simulados mediante el uso de los modelos de Monod y de inhibición por sustrato. Para considerar la incertidumbre presente en datos experimentales, se añadió ruido normal a los datos simulados. La integración numérica fue llevada a cabo utilizando el algoritmo Runge-Kutta de cuarto orden, con un intervalo de integración de 60 horas y un tamaño de paso de 1 hora. Las condiciones iniciales y los valores reales de los parámetros cinéticos se presentan en la Tabla 1. En la Fig. 19, se ilustran los datos generados mediante el uso del modelo de Monod.

Tabla 1.- Condiciones iniciales y valores de los parámetros cinéticos usados para generar los datos de procesos de fermentación en biorreactor de operación tipo lote.

Condición inicial	Valor (g/L)
x_0	0.2
s_0	40
p_0	0
Parámetro cinético	**Valor**
μ_{max}	1.2 1/h
y_{xs}	0.2
k_s	280 g/L
y_{px}	4
k_i	0.3 L/g

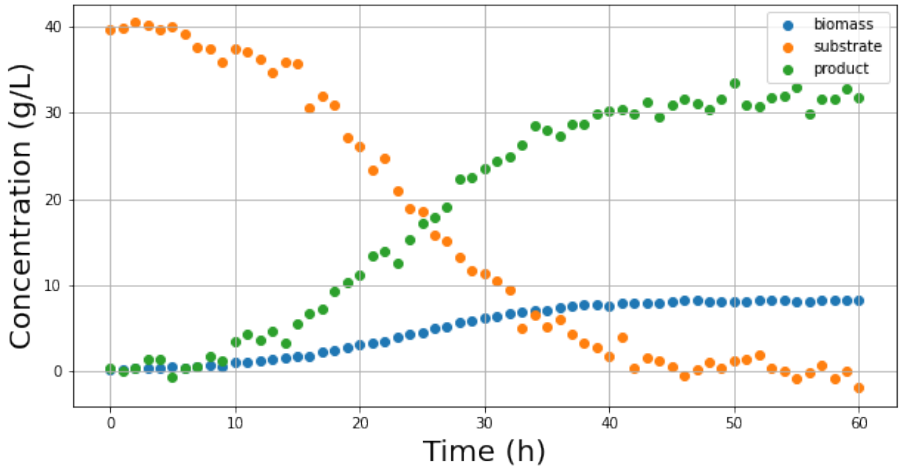


Figura 19.- Gráfica de los datos generados para llevar a cabo la estimación de los parámetros cinéticos y evaluar la eficiencia del algoritmo de optimización. La aleatoriedad que se observa se debe a que se añadió ruido a través de una distribución normal.

4.1.2. Planteamiento del problema de optimización para la estimación de parámetros cinéticos

Función objetivo

Dado que interesa obtener el vector de parámetros cinéticos, $\vec{p}$, que da lugar a la menor diferencia entre las predicciones realizadas con el modelo, $\vec{x}_i$, y las observaciones experimentales, $\vec{x}_{exp,i}$, se emplea la suma de los cuadrados del error SSE (por sus siglas en inglés, Sum of Squared Errors) como función a minimizar.

$$SSE(\vec{p}) = \sum_{i=1}^{n} \left(\vec{x}_i(\vec{p}) - \vec{x}_{exp,i}\right)^2 \tag{20}$$

donde n es el número de observaciones en los datos, $\vec{x}_i$ representa el vector con la i-ésima predicción hecha con el modelo y $\vec{x}_{exp,i}$ es el vector con la i-ésima observación experimental.

Balance de materia para la biomasa

En un biorreactor tipo lote no existen flujos en la alimentación ni en la salida (Fig.18). Por lo tanto, la única forma de cambiar la concentración de las especies es a través de las reacciones bioquímicas que se presentan.

Dado que no se considera una muerte o consumo de microorganismos, el cambio en la concentración de biomasa se debe únicamente a la generación de microorganismos.

$$\text{cambio en la cantidad de biomasa} = \text{generación de biomasa}$$

$$\frac{d(xV)}{dt} = Vr_x$$

donde x representa la concentración de biomasa en el biorreactor, V representa el volumen del biorreactor y r_x representa la velocidad de generación de biomasa en el interior del biorreactor. Dado que el volumen es constante es posible sacarlo de la derivada y dividir entre él para obtener.

$$\frac{dx}{dt} = r_x \tag{21}$$

Balance de materia para el sustrato

El cambio en la cantidad de sustrato se debe únicamente al consumo de los microorganismos, por lo que el balance de sustrato es.

$$\text{cambio en la cantidad de sustrato} = -\text{consumo de sustrato}$$

$$\frac{d(sV)}{dt} = -Vr_s$$

donde s representa la concentración de sustrato en el biorreactor y r_s representa la velocidad de consumo de sustrato en el interior del biorreactor. Sacando el volumen de la derivada y dividiendo entre él se obtiene.

$$\frac{ds}{dt} = -r_s \tag{22}$$

Balance de materia para el producto

El producto es generado en el interior del biorreactor como un producto metabólico los microorganismos. Además, no se considera que exista un consumo de este por parte de los microorganismos.

$$\text{cambio en la cantidad de producto} = \text{generación de producto}$$

$$\frac{d(pV)}{dt} = Vr_p$$

donde p representa la concentración de producto y r_p representa la velocidad de generación de producto en el interior del biorreactor. Sacando el volumen de la derivada y dividiendo entre él se obtiene.

$$\frac{dp}{dt} = r_p \tag{23}$$

Rendimientos

Para relacionar la velocidad de crecimiento de biomasa, r_x, con las velocidades de consumo de sustrato, r_s, y la velocidad de generación de producto, r_p, se emplean los rendimientos biomasa-sustrato y producto-biomasa, respectivamente. Estos rendimientos se definen como.

Rendimiento biomasa-sustrato

$$Y_{xs} = \frac{r_x}{r_s} \tag{24}$$

Rendimiento producto-biomasa

$$Y_{px} = \frac{r_p}{r_x} \tag{25}$$

donde Y_{xs} y Y_{px} son constantes.

Empleando (24) y (25) se puede sustituir los valores de r_s y r_p en (22) y (23) para obtener.

$$\begin{aligned} \frac{dx}{dt} &= r_x \\ \frac{ds}{dt} &= -\frac{1}{Y_{xs}} r_x \\ \frac{dp}{dt} &= Y_{px} r_x \end{aligned} \tag{26}$$

Expresiones cinéticas para describir la velocidad de crecimiento, r_x

La velocidad de crecimiento de biomasa, r_x, se expresa comúnmente como el producto de una tasa de crecimiento que depende de la concentración de sustrato multiplicada por la concentración de biomasa.

$$r_x = \mu(s)x \tag{27}$$

donde $\mu(s)$ representa la tasa de crecimiento dependiente de la concentración de sustrato. Se han propuesto diversas expresiones para representar esta tasa. Algunas de ellas consideran efectos de inhibición por parte del sustrato o producto. Para la optimización paramétrica se consideran dos modelos. El primero de ellos es el modelo de Monod. Este modelo es el más popular dentro de los modelos de crecimiento microbiano (Monod, 1949).

$$\mu(s) = \mu_{max}\left(\frac{s}{k_s + s}\right)$$

(28)

donde μ_{max} representa la tasa máxima de crecimiento y k_s, conocido como constante de afinidad al sustrato, represente la concentración de sustrato a la cual se alcanza la mitad de la tasa máxima de crecimiento, un valor bajo de este parámetro indica una alta afinidad por el sustrato.

Si existe un efecto de inhibición del sustrato sobre el crecimiento de los microorganismos (e.g., la tasa de crecimiento aumenta de forma no monótona con la concentración de sustrato) es necesario usar un modelo diferente a (28). Este efecto se considera en la siguiente ecuación (Andrews, 1968).

$$\mu(s) = \mu_{max}\left(\frac{s}{k_s + s + k_i s^2}\right) \tag{29}$$

donde k_i representa el grado de inhibición del sustrato sobre el crecimiento del microorganismo. El modelo de Monod es una caso especial de (29) cuando $k_i = 0$, (i.e., cuando no hay efecto de inhibición).

Considerando lo anterior, el problema de optimización se plantea como sigue.

Minimizar:

$$SSE(\vec{p}) = \sum_{i=1}^{n}\left(\vec{x}_i(\vec{p}) - \vec{x}_{exp,i}\right)^2 \tag{20}$$

Sujeto a:

$$\begin{aligned}
\frac{dx}{dt} &= r_x \\
\frac{ds}{dt} &= -\frac{1}{Y_{xs}} r_x \\
\frac{dp}{dt} &= Y_{px} r_x
\end{aligned} \tag{26}$$

$$x(0) = x_0 \tag{30}$$

$$s(0) = s_0 \tag{31}$$

$$p(0) = p_0 \tag{32}$$

$$x(t) \geq 0 \tag{33}$$

$$s(t) \geq 0 \tag{34}$$

$$p(t) \geq 0 \tag{35}$$

Aquí, r_x en (26) se puede sustituir por el modelo de Monod (28) o el modelo de inhibición por sustrato (29).

4.1.3. Resultados de la implementación para la estimación de parámetros cinéticos mediante algoritmos genéticos

En este estudio, se empleó el uso de algoritmos genéticos para la estimación de los parámetros cinéticos. Se justifica la elección de esta clase de algoritmos en la sección 2.3. El algoritmo genético requiere la configuración de parámetros específicos, tales como el tamaño de la población, la probabilidad de cruzamiento, la probabilidad de mutación y el número de generaciones. Los valores utilizados para estos parámetros se presentan en la Tabla 2. Estos parámetros fueron elegidos mediante un proceso de prueba y evaluación de diferentes opciones, con el objetivo de maximizar el rendimiento en términos de tiempo de convergencia y precisión de los parámetros estimados. Además, se llevó a cabo una revisión de la literatura existente, donde se examinaron los trabajos previos que utilizan algoritmos genéticos para estimar parámetros en modelos similares.

En el algoritmo, es necesario introducir los rangos de búsqueda para las variables que se van a optimizar. Estos rangos de búsqueda se muestran en la Tabla 3. Los límites se fijaron con base a los valores reportados por diferentes autores en la literatura (Sheng y Sheu, 2000; Joelianto y cols., 2001; Rivera y cols., 2006).

Tabla 2.- Parámetros del algoritmo genético empleados para la estimación de los parámetros cinéticos.

Parámetro del algoritmo	Valor
tamaño de población	50
probabilidad de cruzamiento	0.8
probabilidad de mutación	0.2
número de generaciones	50

Tabla 3.- Parámetros cinéticos que se optimizaron y sus rangos de búsqueda.

Parámetro cinético	Límite inferior	Límite superior
$\mu_{max}\ (1/h)$	1	3
$k_s\ (g/L)$	200	300
$k_i\ (L/g)$	0.05	0.5
Y_{xs}	0.05	1
Y_{px}	1	10

Para verificar la precisión de los valores obtenidos para los parámetros cinéticos, se llevaron

a cabo cinco estimaciones independientes para cada modelo. Los resultados se presentan en la Tabla 4. El tiempo requerido para llevar a cabo las estimaciones fue de 35 segundos, utilizando un sistema operativo Windows, un procesador AMD Ryzen 2900 MHz y 8GB de memoria RAM. Los algoritmos se implementaron utilizando la librería `geneticalgorithm` en el lenguaje de programación `Python` versión 3.10.

Tabla 4.- Valores de la función objetivo en las 5 estimación con ambos modelos.

Estimación	Monod	Inhibición
1	6.409	2.652
2	3.556	2.530
3	4.979	2.387
4	2.296	3.292
5	4.145	3.695
Promedio	4.277	2.911
Original	1.822	2.221

Como se puede observar en la Tabla 4, los valores promedio de la función objetivo resultaron superiores a los obtenidos con los valores originales de los parámetros. Sin embargo, es importante destacar que, a pesar de la presencia de ruido que se añadió a los datos, la diferencia es pequeña.

Los valores estadísticos para los parámetros cinéticos estimados con el modelo de Monod se presentan en la Tabla 5. Se puede ver que el intervalo de confianza del 95 % para los 4 parámetros incluye el valor original, lo que indica que el algoritmo es capaz de realizar estimaciones precisas.

Tabla 5.- Valores estadísticos de los parámetros cinéticos estimados usando el modelo de Monod. LI y LS sol los límites inferior y superior del intervalo de confianza, respectivamente.

Parámetro	Promedio	Desviación estándar	LI	LS	Original
$\mu_{max}\ (1/h)$	1.190	0.0855	1.115	1.265	1.2
$k_s\ (g/L)$	275.305	10.996	265.666	284.944	280
Y_{xs}	0.207	0.0454	0.167	0.247	0.2
Y_{px}	3.944	1.020	3.049	4.838	4

En la Tabla 6 se muestran los valores estadísticos de los parámetros cinéticos obtenidos con el modelo de inhibición por sustrato. Al igual que con el modelo de Monod, todos los intervalos de confianza incluyen el valor original.

Tabla 6.- Valores estadísticos de los parámetros cinéticos estimados usando el modelo de inhibición por sustrato. LI y LS sol los límites inferior y superior del intervalo de confianza, respectivamente.

Parámetro	Promedio	Desviación estándar	LI	LS	Original
$\mu_{max}\ (1/h)$	1.230	0.102	1.140	1.320	1.2
$k_s\ (g/L)$	261.890	27.765	237.552	286.228	280
$k_i\ (L/g)$	0.336	0.063	0.281	0.392	0.3
Y_{xs}	0.202	0.046	0.161	0.243	0.2
Y_{px}	4.104	1.030	3.201	5.008	4

En la Fig. 20 se muestran los estadísticos de la Tabla 5 junto con los valores obtenidos en cada estimación. Es posible observar que los valores estimados se distribuyen al rededor del valor original. Sin embargo, se nota que el valor promedio se encuentra relativamente cerca del valor original y el intervalo de confianza al 95 % siempre incluye el valor original, lo cual sugiere una buena precisión en la estimación de los parámetros.

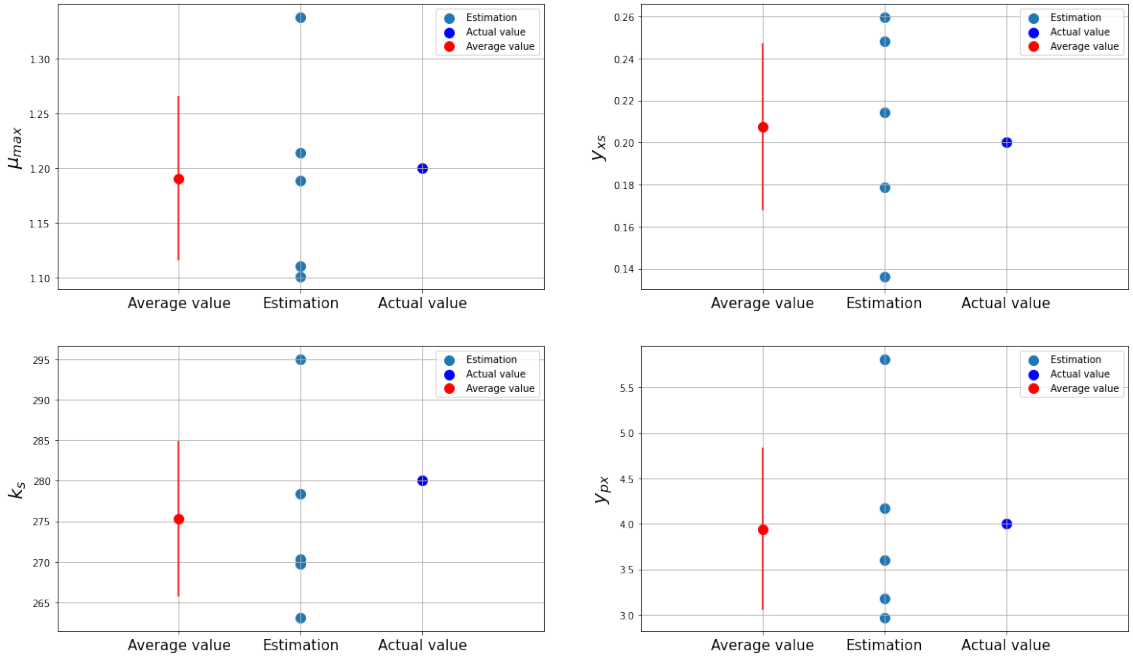


Figura 20.- Valores de los parámetros cinéticos del modelo de Monod que fueron estimados y sus intervalos de confianza. Se observa que todos los intervalos de confianza incluyen el valor original del parámetros.

En la Fig. 21 se muestran los estadísticos de la Tabla 6 junto con los valores estimados. Los

resultados son similares a los obtenidos con el modelo de Monod.

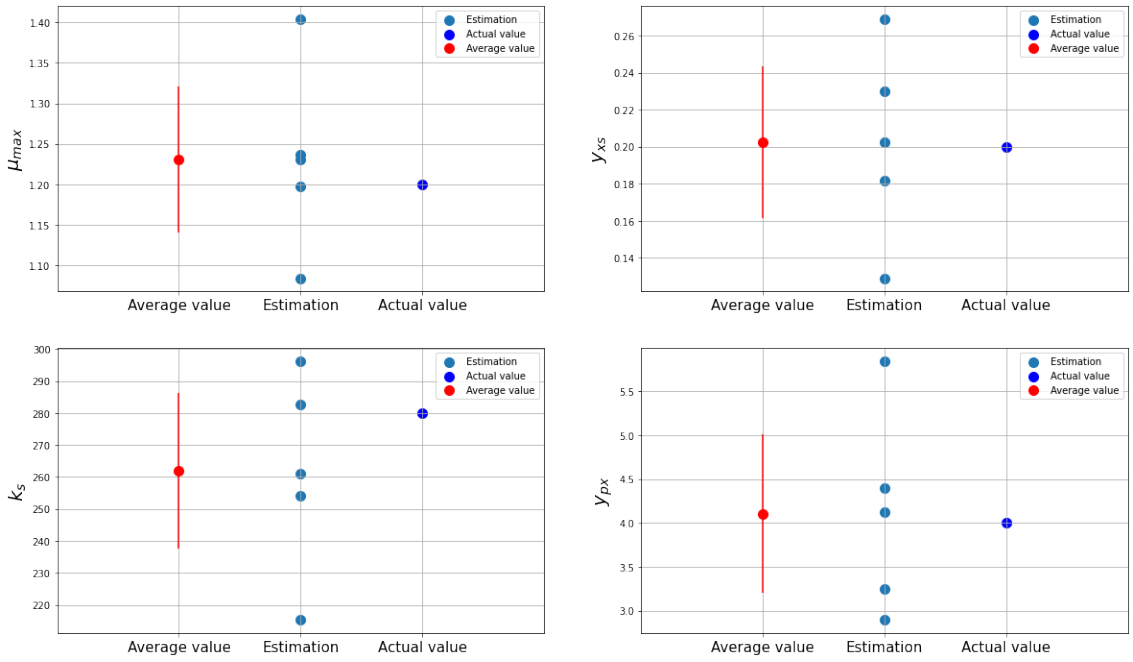


Figura 21.- Valores de los parámetros cinéticos del modelo de inhibición que fueron estimados y sus intervalos de confianza. Se observa que todos los intervalos de confianza incluyen el valor original del parámetros.

La utilización de algoritmos genéticos se ha demostrado efectiva para estimar de manera precisa y rápida los valores de los parámetros cinéticos de fermentaciones en biorreactores tipo lote, así como sus intervalos de confianza. A partir de esta conclusión, se plantea la integración del código y los modelos utilizados en el desarrollo de una aplicación web que automatice este análisis.

Inclusión del análisis en la aplicación web

Este análisis, que es fundamental para comprender y optimizar los procesos de fermentación, ha sido incorporado en la primera versión de la aplicación web. La decisión de integrar la estimación de parámetros cinéticos mediante algoritmos genéticos se basó en su relevancia crítica para el ajuste y la precisión de los modelos de fermentación. Al ser un componente esencial para la comprensión de estos procesos, su inclusión en la versión inicial de la aplicación permite a los usuarios realizar evaluaciones y ajustes significativos de manera eficiente y automatizada.

4.2. Optimización del flujo de alimentación mediante la implementación de algoritmos genéticos

4.2.1. Descripción del problema de optimización del flujo de alimentación en el biorreactor tipo lote-alimentado

La utilización de un biorreactor de operación tipo lote-alimentado es requerida en aquellos casos en los que elevadas concentraciones de sustratos presentan un efecto inhibitorio sobre el crecimiento de los microorganismos (Lim y Shin, 2013) (véase Fig. 22). Por consiguiente, para evitar estos efectos de inhibición se alimenta gradualmente el sustrato durante todo el proceso. La fermentación se lleva a cabo durante un periodo de tiempo específico y una vez finalizada se recuperan tanto la biomasa como los sustratos no consumidos.

En esta sección, se propone una metodología para obtener el flujo óptimo de alimentación a un biorreactor de operación tipo lote-alimentado.

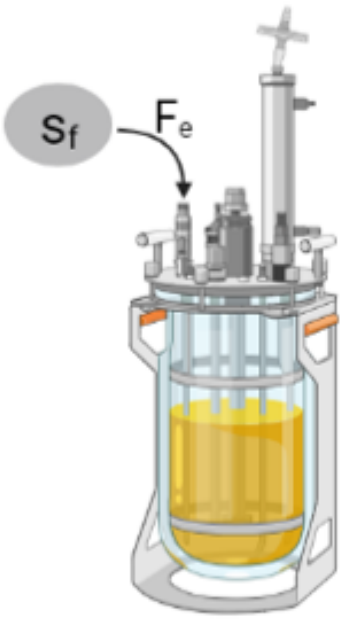


Figura 22.- Esquema de un biorreactor tipo lote-alimentado para suministrar sustrato de forma gradual durante la fermentación. Imagen creada por el autor usando BioRender.

4.2.2. Planteamiento del problema de optimización del flujo de alimentación

Función objetivo

El problema de optimización consiste en determinar el flujo de alimentación al biorreactor, representado por la variable F_e, que permite maximizar la producción de biomasa al finalizar el proceso de fermentación. Dado que tanto el volumen como la concentración de biomasa experimentan cambios durante todo el proceso, se formula la función objetivo como sigue:

$$P_x(F_e) = x(t_f)V(t_f) \tag{36}$$

donde $x(t_f)$ representa la concentración de biomasa al final de la fermentación, $V(t_f)$ representa el volumen al final de la fermentación y t_f representa el tiempo que dura la fermentación.

Se consideran las siguientes restricciones para el flujo de alimentación y el volumen en el interior del biorreactor durante la búsqueda de la solución.

$$0 < F_e(t) < 2\ L/h \tag{37}$$

$$0 < V(t_f) \leq 200\ L \tag{38}$$

Los límites aquí establecidos se basan en el conocimiento que se tiene sobre los valores comunes del flujo de alimentación y el volumen de biorreactores. Además, se han considerado los rangos de estos dos parámetros empleados en la literatura (Patel y Padhiyar, 2016; Rocha y cols., 2014; Yüzgeç, 2010).

Para garantizar que el volumen permanece en el rango establecido en (38) se emplea una función de penalización. Esta función resta un valor arbitrariamente grande (en este caso se seleccionó un valor de 1000000) a la función objetivo cada vez que el volumen supera el límite superior impuesto por (38). Considerando lo anterior, la función objetivo se puede reescribir como.

$$P_x(F_e) = \begin{cases} x(t_f)V(t_f) & V \leq 200\ L \\ x(t_f)V(t_f) - 1000000 & V > 200\ L \end{cases}$$

De esta manera, si el volumen toma un valor superior a 200 L, el valor de la función objetivo recibe una penalización de -1000000. Por lo cual la solución es descartada en la búsqueda de soluciones que generan valores de la función objetivo grandes.

Balance de materia total

Dado que en este tipo de operación solo se tiene un flujo de alimentación, F_e, se observa que el balance de materia en el biorreactor es.

$$\text{cambio en la cantidad de masa} = \text{entrada de masa}$$

$$\frac{d(\rho V)}{dt} = F_e \rho$$

donde ρ representa la densidad, la cual se supone constante. Si se saca ρ de la derivada y se divide entre ella se llega a.

$$\frac{dV}{dt} = F_e \tag{39}$$

Lo cual indica que el volumen aumenta a un velocidad F_e.

Balance de materia para la biomasa

Dado que el flujo de alimentación no contiene biomasa se tiene que su balance es.

$$\text{cambio en la cantidad de biomasa} = \text{generación de biomasa}$$

$$\frac{d(xV)}{dt} = Vr_x$$

aplicando la regla de la cadena y reordenando.

$$V\frac{dx}{dt} + x\frac{dV}{dt} = Vr_x$$

$$V\frac{dx}{dt} + xF_e = Vr_x$$

despejando dx/dt se llega a.

$$\frac{dx}{dt} = r_x - \frac{F_e}{V}x \tag{40}$$

El termino negativo en (40) podría parecer extraño dado que se indicó que en este tipo de operación no hay un flujo de salida. Dado que tampoco se considera una muerte o un consumo de los microorganismos se podría esperar que todos los términos fueran positivos. Sin embargo, debido a que en el flujo de alimentación no existe biomasa y el volumen en el biorreactor aumenta durante el proceso, se genera un efecto de dilución sobre la biomasa. Por esta razón al introducir un flujo de alimentación, F_e, la concentración de biomasa en el interior del biorreactor disminuye.

Balance de materia para el sustrato

Dado que en el flujo de alimentación existe sustrato se tiene.

$$\text{cambio en la cantidad de sustrato} = \text{entrada de sustrato} - \text{consumo de sustrato}$$

$$\frac{d(sV)}{dt} = F_e s_f - Vr_s$$

aplicando la regla de la cadena.

$$V\frac{ds}{dt} + s\frac{dV}{dt} = F_e s_f - Vr_s$$

$$V\frac{ds}{dt} + F_e s = F_e s_f - Vr_s$$

despejando ds/dt se obtiene.

$$\frac{ds}{dt} = \frac{F_e}{V}s_f - r_s - \frac{F_e}{V}s$$

además, considerando el rendimiento biomasa-sustrato

$$Y_{xs} = \frac{r_x}{r_s}$$

se llega a.

$$\frac{ds}{dt} = \frac{F_e}{V}(s_f - s) - \frac{1}{Y_{xs}}r_x \tag{41}$$

Como se mencionó al inicio de esta sección, un biorreator tipo lote-alimentado se emplea porque se conoce existen efectos de inhibición del sustrato sobre el crecimiento de los microorganismos. Por lo tanto, aquí se emplea el modelo de inhibición por sustrato (29) para describir la velocidad de crecimiento, r_x.

De tal manera que el modelo que describe el proceso es.

$$\begin{aligned}
\frac{dV}{dt} &= F_e \\
\frac{dx}{dt} &= \mu_{max}\left(\frac{s}{k_s + s + k_i s^2}\right) - \frac{F_e}{V}x \\
\frac{ds}{dt} &= \frac{F_e}{V}(s_f - s) - \frac{1}{Y_{xs}}\mu_{max}\left(\frac{s}{k_s + s + k_i s^2}\right) \\
\frac{dp}{dt} &= Y_{px}\mu_{max}\left(\frac{s}{k_s + s + k_i s^2}\right) - \frac{F_e}{V}p
\end{aligned} \tag{42}$$

Considerando lo anterior, el problema de optimización se enuncia como sigue.

Maximizar:

$$P_x(F_e) = x(t_f)V(t_f) \tag{36}$$

Sujeto a:

$$\begin{aligned}
\frac{dV}{dt} &= F_e \\
\frac{dx}{dt} &= \mu_{max}\left(\frac{s}{k_s + s + k_i s^2}\right) - \frac{F_e}{V}x \\
\frac{ds}{dt} &= \frac{F_e}{V}(s_f - s) - \frac{1}{Y_{xs}}\mu_{max}\left(\frac{s}{k_s + s + k_i s^2}\right) \\
\frac{dp}{dt} &= Y_{px}\mu_{max}\left(\frac{s}{k_s + s + k_i s^2}\right) - \frac{F_e}{V}p
\end{aligned} \tag{42}$$

$$x(0) = 0.2\ g/L \tag{43}$$

$$s(0) = 40\ g/L \tag{44}$$

$$p(0) = 0\ g/L \tag{45}$$

$$V(0) = 10\ L \tag{46}$$

$$0 < F_e(t) < 2\ L/h \tag{37}$$

$$0 < V(t_f) \leq 200\ L \tag{38}$$

$$x(t) \geq 0 \tag{47}$$

$$s(t) \geq 0 \tag{48}$$

$$p(t) \geq 0 \tag{49}$$

4.2.3. Resultados de la implementación de algoritmos genéticos para la optimización del flujo de alimentación

Para determinar el flujo de alimentación óptimo en el biorreactor, se consideraron dos escenarios diferentes. En primer lugar, se evaluó el caso en el cual el flujo de alimentación se mantenía constante durante las 100 horas de la fermentación. En segundo lugar, se analizó el caso en el cual el perfil del flujo de alimentación variaba cada 10 horas durante las 100 horas de la fermentación. En lo sucesivo en esta sección, se denomina al flujo de alimentación constante como ”flujo”, y al perfil del flujo de alimentación como ”perfil”. Un perfil de alimentación se justifica debido a que a medida que progresa la fermentación las necesidades metabólicas de los microorganismos cambian. Por esta razón, es recomendable cambiar el flujo de alimentación de manera que el sustrato disponible en el biorreactor sea el que los microorganismos requieren en cada momento.

En la Tabla 7 se presentan los parámetros de operación utilizados en la optimización. El volumen inicial, la concentración de sustrato en la alimentación, y el tiempo de fermentación se establecieron con base a los valores comunes empleados en procesos de fermentación de tipo lote-alimentado. Las condiciones iniciales y el valor de los parámetros cinéticos se muestran en la Tabla 1.

Es esencial señalar que la precisión de los resultados obtenidos en la optimización está estrechamente relacionada con el valor de los parámetros de operación empleados (véase Tabla 7). Este es uno de los motivos que justifica el desarrollo de una aplicación web, que permite realizar la optimización del flujo de alimentación de manera eficiente y automatizada. El usuario solo debe especificar los valores de estos parámetros y la aplicación se encarga de hallar el flujo

de alimentación óptimo.

Tabla 7.- Parámetros de operación usados en la optimización del flujo de alimentación.

Parámetro de operación	Valor
s_f	50 g/L
t_f	100 h
x_0	0.2 g/L
s_0	40 g/L
V_0	10 L

En ambas estrategias de optimización, se emplearon algoritmos genéticos. La justificación del uso de algoritmos genéticos para la optimización de este problema de optimización se describe en la sección 2.3. En la Tabla 8 se presentan los parámetros del algoritmo genético utilizados en la optimización. Dado que el valor de estos parámetros tienen un gran impacto en la capacidad del algoritmo para encontrar la solución óptima, se llevaron a cabo experimentos computacionales y se consultaron los valores utilizados en las referencias disponibles para determinar los valores correctos. El tamaño de la población y el número de generaciones son de especial importancia dado que valores demasiado bajos de estos parámetros pueden dar como resultado soluciones subóptimas, mientras que valores excesivamente elevados pueden requerir un tiempo de cómputo muy prolongado.

Tabla 8.- Valores de los parámetros del algoritmo genético empleados en la optimización del flujo de alimentación.

Parámetro del algortimo	Valor
tamaño de población (F_e flujo)	20
tamaño de población (F_e perfil)	50
probabilidad de cruzamiento	0.8
probabilidad de mutación	0.1
número de generaciones (F_e flujo)	20
número de generaciones (F_e perfil)	50

Para comprobar la eficiencia del algoritmo genético en la determinación del flujo y perfil óptimos, se llevó a cabo la estimación de estos valores 5 veces. El tiempo requerido para llevar a cabo las optimizaciones fue de 3.6 segundos para el flujo, y de 27 segundos para el perfil. Se utilizó un equipo con sistema operativo Windows, procesador AMD Ryzen 2900 MHz y 8GB de

RAM. Para implementar los algoritmos se hizo uso de la librería `geneticalgorithm` del lenguaje de programación `Python` versión 3.10.

En la Tabla 9 se presentan los valores óptimos obtenidos, así como la producción de biomasa correspondiente. Como se esperaba, la optimización con el perfil de alimentación resultó en valores superiores de producción de biomasa en comparación con la optimización con el flujo de alimentación. Esto se debe a la capacidad de modificar el suministro de sustrato de acuerdo a las necesidades metabólicas de los microorganismos con el perfil. La producción máxima de biomasa fue de 823.659 gramos con el perfil, mientras que con el flujo fue de solo 694.679 gramos. Además, se observa que el valor promedio del perfil se acercó al valor del flujo, el cual fue de aproximadamente 1 L/h.

Tabla 9.- Valores del flujo y perfiles óptimos, así como biomasa producida.

Estimación	Flujo		Perfil	
	$F_e\ (L/h)$	Producción de biomasa (g)	F_e promedio (L/h)	Producción de biomasa (g)
1	1.025	694.257	1.099	818.357
2	0.913	692.587	1.075	823.659
3	0.934	693.693	1.103	819.120
4	1.001	694.679	1.084	819.834
5	0.971	694.667	1.137	809.152

La Figura 23 ilustra la relación entre la producción de biomasa y el flujo de alimentación. También se presentan los valores óptimos determinados mediante las cinco estimaciones. La variabilidad observada en los flujos óptimos se debe a la naturaleza estocástica de los algoritmos genéticos utilizados para encontrar dichos valores. Sin embargo, se observa una tendencia hacia un valor común entre todos los flujos óptimos determinados. Además, se puede apreciar que la producción de biomasa es más sensible a cambios en el flujo de alimentación en rangos inferiores a 1 L/h, mientras que a valores superiores a 1 L/h, la función muestra una menor variabilidad.

La relación observada entre la producción de biomasa y el flujo de alimentación en la Figura 23 puede ser explicada por dos fenómenos complementarios. Por un lado, el aumento del flujo de alimentación al biorreactor proporciona una mayor cantidad de sustrato disponible para la generación de biomasa por parte de los microorganismos. Sin embargo, a medida que el flujo de alimentación continúa incrementando, se alcanzan concentraciones elevadas de sustrato en el biorreactor que comienzan a causar efectos de inhibición sobre los microorganismos. Además,

el aumento del volumen del biorreactor debido al incremento del flujo de alimentación puede contribuir a un efecto de dilución de la biomasa.

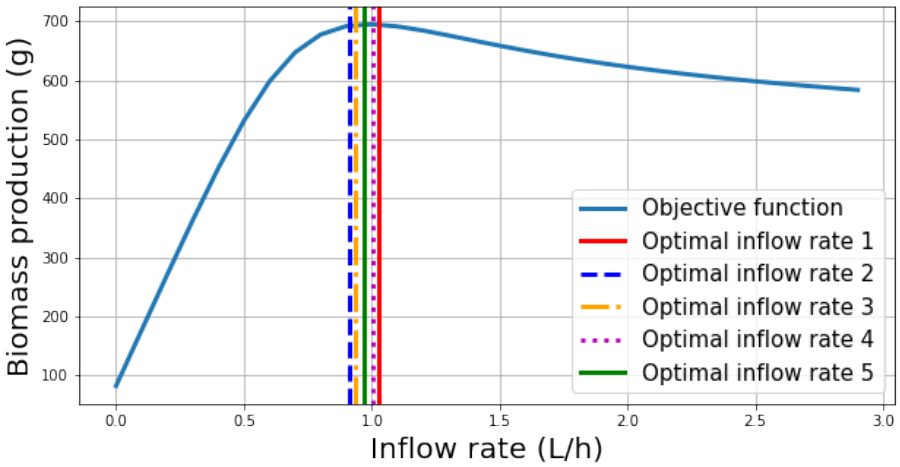


Figura 23.- Producción de biomasa con función del flujo de alimentación. Se muestran como líneas verticales los valores óptimos obtenidos en cada una de las estimaciones.

La Fig. 24 presenta los perfiles óptimos de flujo de alimentación determinados mediante las cinco estimaciones. Los perfiles muestran un patrón característico de un flujo que aumenta progresivamente a medida que avanza la fermentación, alcanzando un valor máximo de 2 L/h aproximadamente a las 50 horas. Este patrón se puede atribuir a la dinámica del proceso: al inicio de la fermentación existe una elevada concentración de sustrato en el biorreactor, por lo que no se requiere un flujo de alimentación elevado, a medida que los microorganismos consumen el sustrato, se hace necesario incrementar el flujo de alimentación para continuar con el proceso. Sin embargo, una vez que se alcanza el límite superior de 2 L/h impuesto por la restricción (38), el flujo no puede continuar aumentando. La Fig. 25 muestra que, a pesar de esto, la concentración de biomasa sigue incrementando después de alcanzar este límite, lo que indica que los microorganismos siguen consumiendo sustrato y, por lo tanto, el flujo debe mantenerse en su valor máximo para continuar con la fermentación.

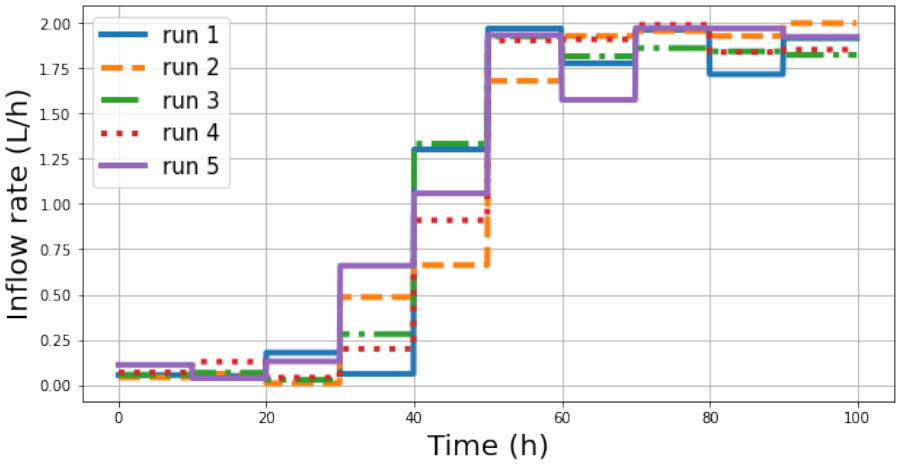


Figura 24.- Perfiles de alimentación óptimo de las cinco estimaciones.

La Fig. 25 ilustra que el volumen máximo alcanzado durante el proceso de fermentación no supera el límite superior impuesto por la restricción 38, por lo que no es necesario activar la penalización en la función objetivo. Sin embargo, será importante investigar el efecto de la penalización en la función objetivo mediante el uso de límites de volumen inferiores en futuras investigaciones, para comprender cómo dicha penalización influye en la obtención del flujo y perfil óptimos.

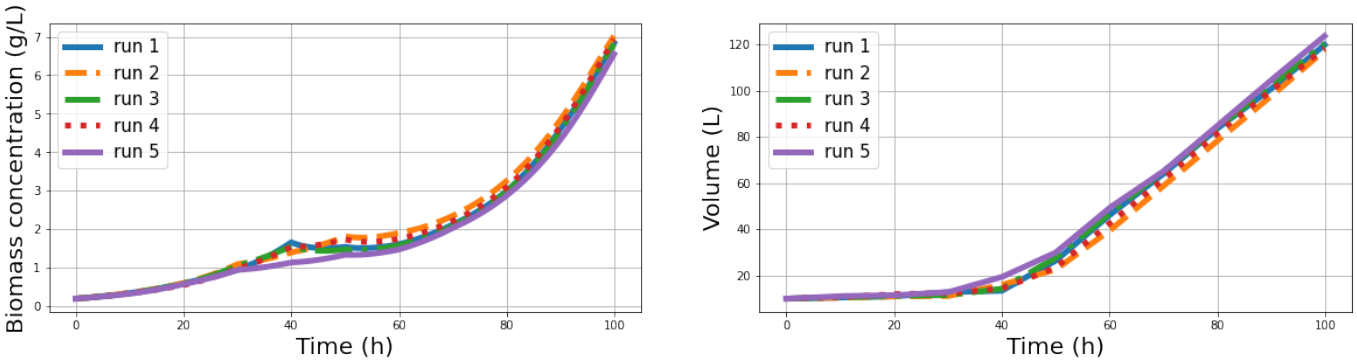


Figura 25.- Cambio en la concentración de biomasa (izquierda) y volumen (derecha) para cada uno de los perfiles de alimentación óptimos.

Los resultados obtenidos en este estudio indican que es posible utilizar algoritmos genéticos para estimar los flujos y perfiles de alimentación óptimos en un proceso de fermentación. Sin embargo, es importante tener en cuenta que la solución obtenida con el perfil óptimo solo considera una discretización del flujo de alimentación en 10 intervalos. El tiempo de cómputo

requerido para obtener un perfil óptimo con una discretización mayor puede resultar prohibitivo.

Inclusión del análisis en la aplicación web

La optimización del flujo de alimentación es un aspecto avanzado en la gestión de procesos de fermentación, y su implementación requiere un análisis detallado y una validación exhaustiva. Aunque esta funcionalidad es altamente valiosa, se ha decidido posponer su integración para una versión futura de la aplicación. Esta decisión se tomó para asegurar que la versión inicial de la aplicación mantuviera un enfoque claro y sencillo, permitiendo así una validación efectiva con los usuarios y evitando la complejidad excesiva. La incorporación de esta funcionalidad en futuras versiones dependerá de la retroalimentación de los usuarios y de la evolución de las necesidades de la industria.

4.3. Funcionamiento de la red neuronal artificial para la simulación de procesos dinámicos de fermentación

De acuerdo con la literatura consultada, existen varias configuraciones que permiten utilizar una red neuronal artificial en la simulación de procesos dinámicos. La mayoría de estas técnicas utilizan trayectorias dinámicas del proceso para entrenar a la red. Las trayectorias consisten en valores de las variables de proceso en diferentes valores de tiempo. Las entradas a la red son los valores de las variables de estado en un instante temporal i y como salidas de la red, los valores de las variables de estado en un instante temporal $i + 1$, o bien, alguna medida del cambio de las variables que puede ser usado para actualizar sus valores. Esto es análogo a la forma en que funciona un algoritmo de integración numérica, donde se requiere conocer el valor de las variables de estado en un tiempo determinado, así como la derivada y el tamaño del paso en el tiempo para predecir el valor de las variables de estado en el siguiente paso de tiempo.

Como se aprecia en la Fig. 26, la red neuronal busca crear una representación del campo de velocidades del proceso, lo que le permite generar predicciones acerca de la trayectoria futura del mismo, dada una condición inicial específica. Esto se logra mediante un proceso de entrenamiento en el cual se utiliza un gran volumen de trayectorias previamente registradas, las cuales sirven como referencia para la red neuronal en cuanto al comportamiento esperado del sistema. Es importante mencionar que la calidad de las predicciones generadas por la red está estrechamente relacionada con la cantidad y calidad de los datos de entrenamiento utilizados.

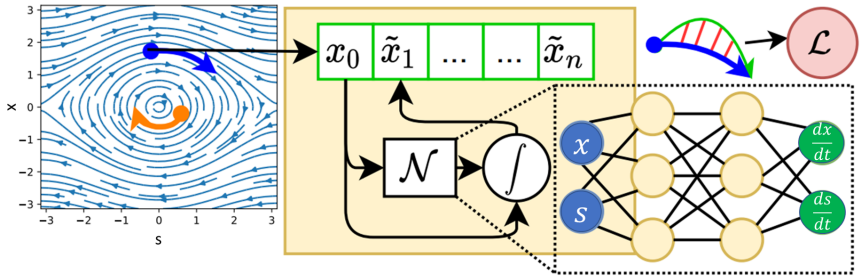


Figura 26.- Red neuronal para simular un proceso dinámico. A partir de una condición inicial dada x_0, el siguiente estado del sistema x_{i+1} se obtiene alimentando el estado actual x_i en la red N, produciendo una derivada que se integra usando un esquema de integración. La función de pérdida se evalúa comparando lo predicho con la trayectoria de entrenamiento.
Adaptado de (Legaard, C. et al. (2023). Constructing neural network based models for simulating dynamical systems. ACM Computing Surveys, 55(11), 1-34.)

La red se entrena mediante una función de pérdida que evalúa qué tan cercana es la trayectoria predicha por la red en comparación con la trayectoria real. Una vez que la red ha sido entrenada, esta puede ser usada para predecir trayectorias a partir de nuevas condiciones iniciales a las que la red no había sido expuesta.

4.3.1. Generación de trayectorias de entrenamiento y validación

La forma en que se realizó el entrenamiento de la red neuronal fue mediante la utilización de valores de las variables biomasa, x; sustrato, s; y producto, p en diferentes instantes temporales. Específicamente, se utilizaron los valores de estas variables en un instante temporal t_i, como entrada para la red neuronal, y se utilizaron los valores correspondientes en el instante temporal t_{i+1} como salida deseada para la red. De esta manera, mediante un proceso iterativo de entrenamiento se logró que la red neuronal aprendiera a generar predicciones precisas acerca de como cambiarían los valores de estas variables.

Los valores de las variables utilizados para el entrenamiento de la red neuronal pueden provenir de diferentes fuentes, tales como mediciones experimentales o simulaciones realizadas mediante modelos mecanísticos, como son las ecuaciones diferenciales ordinarias. La práctica de utilizar modelos mecanísticos con alto poder predictivo para generar datos para el entrenamiento

de modelos de aprendizaje automático es conocida en la literatura científica como "modelamiento sustituto". Esta técnica es útil en situaciones donde los datos experimentales son escasos o costosos de obtener. Sin embargo, es importante destacar que es crucial asegurar la validez y precisión del modelo mecanístico utilizado para generar los datos de entrenamiento, dado que cualquier error o imprecisión en el modelo se reflejará en el rendimiento del modelo de aprendizaje automático generado. La generación de modelos de aprendizaje automático para remplazar modelos mecanístico suele ser común en áreas de ingeniería donde el modelo se utiliza para fines de optimización y se requieren modelos que sean computacionalmente económicos de evaluar. En el presente trabajo se utilizó el modelo mecanístico dado por las ecuaciones (27)-(29) para generar los datos de entrenamiento y validación.

Con el fin de generar un conjunto de trayectorias de entrenamiento para la red neuronal, se procedió a generar 100 condiciones iniciales de forma aleatoria. Para asegurar que la distribución de las condiciones iniciales fuera adecuada y representativa, se utilizó el método de muestreo hipercubo latino (MHL). Este método se ha demostrado eficiente en generar un conjunto de muestras aleatorias que cubren de manera uniforme el espacio de los parámetros de entrada del modelo. Además, el método de muestreo hipercubo latino es fácil de implementar y rápido de ejecutar, lo cual lo hace atractivo para su uso en simulaciones con altas demandas computacionales. Los rangos utilizados para generar las condiciones iniciales fueron de 0.2 a 10 g/L para x, de 0 a 40 g/L para s y de 0 a 40 g/L para p.

Con el fin de validar el desempeño del modelo de red neuronal entrenada, se generaron 10 condiciones iniciales mediante un proceso similar al utilizado en el entrenamiento. Sin embargo, los rangos de las variables se establecieron de manera diferente con el objetivo de evaluar condiciones más realista de un proceso de fermentación. En este caso los rangos utilizados fueron de 1 a 6 g/L para x, de 15 a 35 g/L para s y 0 g/L para p.

Una vez establecidas las condiciones iniciales para el entrenamiento y la validación, se procedió a obtener las trayectorias correspondientes mediante el uso de un integrador numérico, combinando las condiciones iniciales establecidas con el modelo mecanístico (27)-(29). En particular, en este trabajo se utilizó el método numérico Runge-Kutta de cuarto orden para simular el sistema dinámico. La simulación se llevó a cabo a lo largo de un intervalo temporal de 1 hora, tomando en cuenta un tamaño de paso de 1 hora también. Esto permitió generar solo

una observación hacia adelante, lo que resultó en trayectorias con 2 observaciones. En la Fig. 27 se presentan cinco de las cien trayectorias de entrenamiento generadas. A pesar de que, en el intervalo temporal de 1 hora considerado, se aprecia que las variables de estado no experimentan cambios significativos, es importante señalar que los pequeños cambios observados proporcionan suficiente información para que la red neuronal aprenda el comportamiento dinámico del proceso.

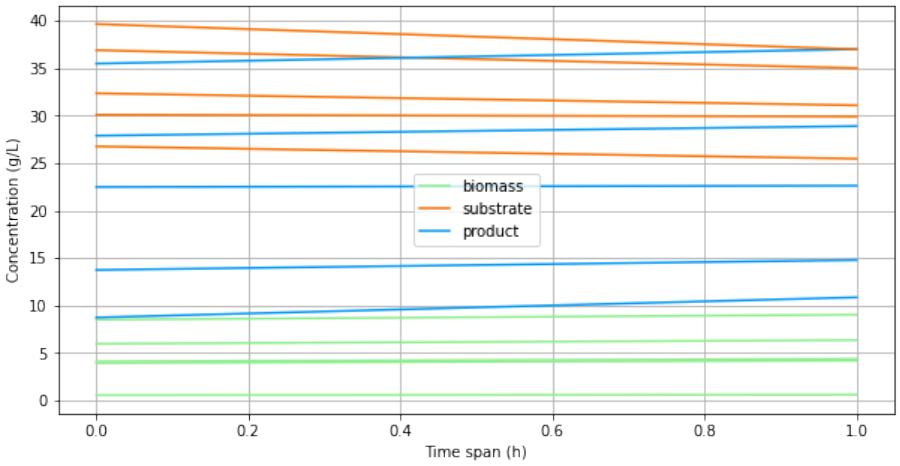


Figura 27.- Trayectorias de entrenamiento con intervalos de 1 hora.

La generación de las trayectorias de validación se llevó a cabo de manera análoga a la utilizada para las trayectorias de entrenamiento, con la única diferencia que el intervalo de simulación considerado fue de 60 horas, con un tamaño de paso de 1 hora. Este procedimiento permitió obtener un total de 61 observaciones. En la Fig. 28, se presentan cinco de las diez trayectorias de validación generadas.

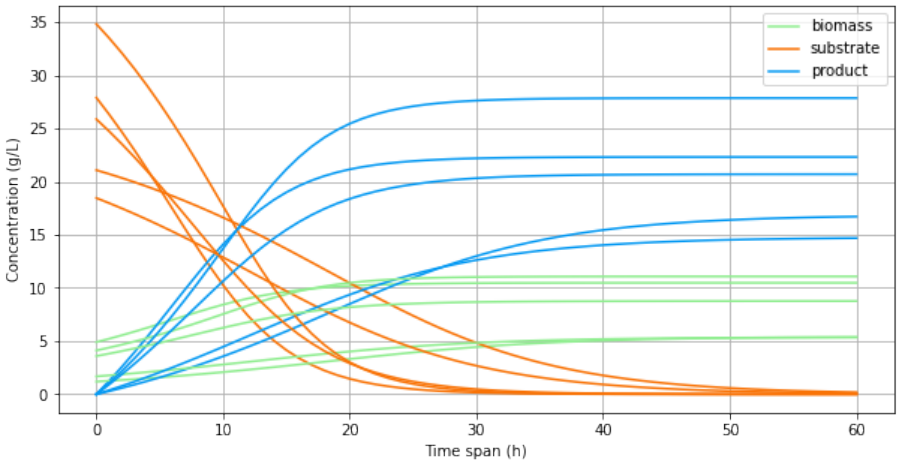


Figura 28.- Trayectorias de validación con intervalos de tiempo de 1 hora.

4.3.2. Estructura de la red neuronal

La estructura de la red se escogió de acuerdo a la revisión de la literatura y a experimentos computacionales realizados. En la capa de entrada y de salida debe existir un número de neuronas igual al número de variables en el problema, en este caso 3. Fuera de esta restricción, el resto de la estructura se debe investigar para encontrar la opción que genere los mejores resultados. Se decidió utilizar 8 capas escondidas con 32 neuronas cada una y función de activación *softplus*.

La función de activación *softplus* es una función no lineal que se utiliza con frecuencia en redes neuronales y en otros modelos de aprendizaje automático, se define como:

$$\text{softplus} = ln\left(1 + e^{x}\right) \tag{50}$$

En la Fig. 29 se muestra su comportamiento. La función *softplus* es suave en todo el rango de valores de x, tiende a 0 a medida que x tiende a $-\infty$ y tiende a ∞ a medida que x tiende a ∞.

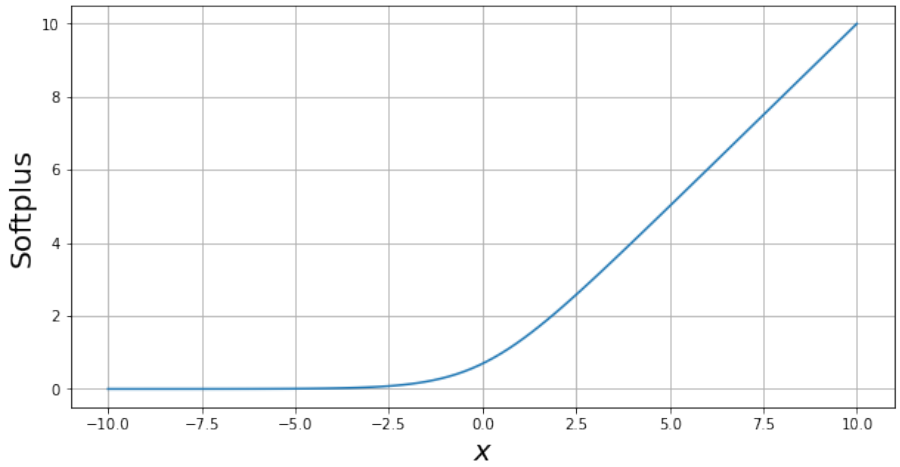


Figura 29.- Función *softplus* usada en las capas escondidas de las redes neuronales.

4.3.3. Función de pérdida

La función de pérdida que se escogió es el valor promedio de los errores al cuadrado, MSE (por sus siglas en inglés, Mean Squared Error). La función de pérdida es evaluada comparando la trayectoria predicha con la red con la trayectoria de entrenamiento. La función de pérdida se escribe como.

$$MSE_{ts}(\hat{X}, X) = \frac{1}{l}\sum_{i=0}^{l-1} MSE_c(\hat{X}, X) \tag{51}$$

$$MSE_c(\hat{x}, x) = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(\hat{x}_{ij} - x_{ij}\right)^2 \tag{52}$$

donde $X \in \mathbb{R}^{l\times m\times n}$ son las trayectorias de entrenamiento, $\hat{X} \in \mathbb{R}^{l\times m\times n}$ son las trayectorias predichas, l es el número de trayectorias, m es la longitud de la trayectorias, n es el número de variables de estado y $\hat{x}_{ij}$ representa el valor de la j-ésima variable de estado en la i-ésima observación de tiempo en la trayectoria predicha.

MSE se utiliza con frecuencia como una función de pérdida en el proceso de entrenamiento de un modelo de aprendizaje automático, dado que es fácil de calcular y minimizar. Una vez que se ha entrenado el modelo, la MSE se puede utilizar para evaluar el rendimiento del modelo en tareas de regresión, dado que mide la diferencia entre los valores predichos y los valores

verdaderos en una escala continua.

4.3.4. Algoritmo de entrenamiento

El algoritmo de entrenamiento que se empleó para minimizar la función de pérdida fue Adam. Adam (por sus siglas en inglés, Adaptive Moment Estimation) es un algoritmo de optimización utilizado para entrenar modelos de aprendizaje automático. Es una variante del algoritmo de descenso de gradiente estocástico, que es un método comúnmente utilizado para minimizar una función de pérdida en un modelo de aprendizaje automático.

Adam funciona adaptando el tamaño de los pasos de descenso del gradiente en cada iteración basándose en los valores anteriores de los gradientes. Esto permite que Adam tenga un rendimiento similar al de otros algoritmos de descenso de gradiente estocástico, pero con la ventaja de que no requiere que el usuario especifique un tamaño de paso fijo.

4.3.5. Configuración de la red

Se propusieron tres diferentes configuraciones de la red neuronal para determinar cual de ellas generaba las mejores predicciones. La diferencia entre cada una de estas configuraciones consiste en la forma en que la red predice el valor de las variables en el instante temporal t_{i+1} a partir de su valor en el instante temporal t_i.

En todas las configuraciones mostradas, durante la etapa de entrenamiento la red es alimentada con trayectorias de 1 h para aprender el comportamiento dinámico de proceso, mientras que en inferencia solo la condición inicial se introduce a la red y esta produce la trayectoria para el intervalo de tiempo especificado.

Escalador de tiempo directo

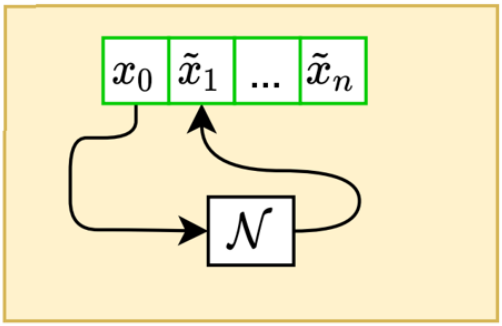
La configuración más simple de la red es donde el valor del estado al instante temporal t_{i+1} se obtiene directamente del valor del instante temporal t_i, es decir.

$$\hat{x}_{i+1} = N(\hat{x}_i) \tag{53}$$

donde N representa la red neuronal con una estructura arbitraria y $\hat{x}_0 = x_0$. La red es

entrenada para producir una estimación del siguiente estado, $\hat{x}_{i+1}$, a partir del estado actual $\hat{x}_i$.

La Fig. 30 a ilustra un esquema de la configuración propuesta. La red neuronal recibe el valor del estado en el instante temporal t_i, y genera el valor en el instante temporal t_{i+1}. Este valor generado por la red puede ser retroalimentado para producir el estado siguiente, y así sucesivamente. En la Fig. 30 b se representa el proceso de entrenamiento, el cual consta de generar una trayectoria a partir de la condición inicial, y luego medir la discrepancia entre la predicción y la trayectoria real, para actualizar los parámetros de la red neuronal a partir de ese valor. La Fig. 30 c ilustra que durante la etapa de inferencia, la red neuronal puede ser utilizada de manera similar a la etapa de entrenamiento para generar una nueva trayectoria.



(a) Esquema de la configuración de la red.

```
X̃[:,0,:] = X_{t_0}
for i in 0...m-1
  X̃[:,i+1,:] = N(X̃[:,i,:])
loss = L_ts(X,X̃)
optimizer.step(loss)
```

(b) Etapa de entrenamiento de la red.

```
X̃[:,0,:] = X_{t_0}
for i in 0...m-1
  X̃[:,i+1,:] = N(X̃[:,i,:])
```

(c) Etapa de inferencia.

Figura 30.- Escalador de tiempo directo.

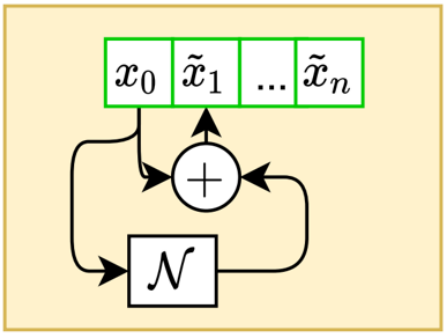
Escalador de tiempo residual

En lugar de predecir el siguiente valor del estado de forma directa, una alternativa es entrenar a la red neuronal para predecir la diferencia entre el estado en el instante temporal t_i y el estado en el instante temporal t_{i+1} (véase Fig. 31). Esta diferencia, conocida como incremento del estado, puede ser sumada al estado actual para generar el siguiente estado en la trayectoria. Es decir, mediante la predicción del incremento del estado, se puede generar el siguiente estado en la trayectoria mediante una simple suma. Esta técnica puede resultar en una mayor eficiencia en el proceso de inferencia, dado que se evita la necesidad de predecir directamente el siguiente valor

del estado.

$$\hat{x}_{i+1} = \hat{x}_i + N(\hat{x}_i) \tag{54}$$

En esta configuración, la red neuronal genera una estimación del cambio en el valor del estado, lo cual se asemeja a la forma en que funcionan los métodos numéricos de integración. Sin embargo, es importante notar que a diferencia de los métodos numéricos convencionales, en esta configuración no se ha considerado explícitamente el tamaño del paso temporal en la actualización del estado.



(a) Esquema de la configuración de la red.

```
X̃[:,0,:] = X_0
for i in 0...m-1
  ΔX = N(X̃[:,i,:])
  X̃[:,i+1,:] = X̃[:,i,:] + ΔX
loss = L_ts(X,X̃)
optimizer.step(loss)
```

(b) Etapa de entrenamiento de la red.

```
X̃[:,0,:] = X_0
for i in 0...m-1
  ΔX = N(X̃[:,i,:])
  X̃[:,i+1,:] = X̃[:,i,:] + ΔX
```

(c) Etapa de inferencia.

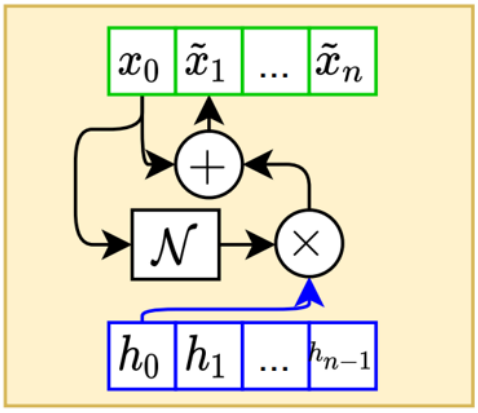
Figura 31.- Escalador de tiempo residual.

Escalador de tiempo numérico

De forma alternativa, el tamaño de paso puede considerarse en la configuración mediante escalar la contribución de la red por el tamaño de paso h_i

$$\hat{x}_{i+1} = \hat{x}_i + h_i * N(\hat{x}_i) \tag{55}$$

La predicción de la red es multiplicada por el tamaño de paso y es añadida al estado actual para formar una predicción del siguiente estado (véase Fig. 32).



(a) Esquema de la configuración de la red.

```
X̃[:,0,:] = X_0
for i in 0...m-1
  ΔX = N(X̃[:,i,:])
  X̃[:,i+1,:] = X̃[:,i,:] + h[i]*ΔX
loss = L_ts(X,X̃)
optimizer.step(loss)
```

(b) Etapa de entrenamiento de la red.

```
X̃[:,0,:] = X_0
for i in 0...m-1
  ΔX = N(X̃[:,i,:])
  X̃[:,i+1,:] = X̃[:,i,:] + h[i]*ΔX
```

(c) Etapa de inferencia.

Figura 32.- Escalador de tiempo numérico.

4.3.6. Resultados obtenidos con las diferentes configuraciones

La Tabla 10 presenta una comparación de los resultados obtenidos durante el proceso de entrenamiento para las tres configuraciones de la red neuronal propuestas. Se observa que el valor de la función de pérdida fue prácticamente cero para las tres configuraciones, lo que indica que la red neuronal tiene la capacidad de describir adecuadamente las trayectorias. Sin embargo, es importante tener en cuenta que un valor bajo de la función de pérdida de entrenamiento no necesariamente indica un buen rendimiento en términos de generalización, dado que puede ser el resultado de un sobreajuste.

Para evaluar la capacidad de la red neuronal en la predicción de nuevas trayectorias, se utilizaron las trayectorias de validación. Los resultados obtenidos se presentan en la Tabla 10. Es posible observar que a diferencia de las trayectorias de entrenamiento, el valor de la función de pérdida fue significativamente mayor. En particular, para el caso del escalador de tiempo directo, la red neuronal no logró predecir de forma satisfactoria las trayectorias de validación, lo cual se ve reflejado en un valor elevado de la función de pérdida (45.128). Sin embargo, para los otros dos escaladores, el valor de la función de pérdida fue prácticamente cero, lo cual indica un buen

rendimiento en términos de generalización.

Tabla 10.- Valores de la función de pérdida tras finalizar el entrenamiento de la red.

Configuración de la red	Entrenamiento	Validación	Tiempo de entrenamiento (s)
Directo	0.0417	45.1284	7
Residual	0	0.0176	7
Numérico	0	0.0246	17

La Tabla 10 también presenta información sobre los tiempos de entrenamiento para las tres configuraciones propuestas. Se observa que para los escaladores de tiempo directo y residual, el tiempo de entrenamiento fue de aproximadamente 7 segundos, mientras que para el escalador numérico fue de alrededor de 17 segundos. Es importante mencionar que el entrenamiento se llevó a cabo en un ambiente de computación en la nube (Google Colaboratory) utilizando una instancia de CPU con un núcleo, a 2.80 GHz y 13 GB de RAM. Estos resultados sugieren que el escalador numérico requiere un tiempo de entrenamiento significativamente mayor comparado con los escaladores de tiempo directo y residual.

La Fig. 33 presenta una comparación de la evolución de las funciones de pérdida de entrenamiento para las tres configuraciones propuestas. Es posible observar que la configuración utilizando el escalador de tiempo directo convergió a un valor mayor de pérdida en comparación con los otros escaladores, tal como se indica en la Tabla 10. Además, se puede apreciar que los escaladores de tiempo residual y numérico alcanzaron la convergencia en un tiempo menor comparado con el escalador de tiempo directo. Estos resultados sugieren que los escaladores de tiempo residual y numérico presentan una mayor eficiencia en términos de convergencia de la función de pérdida en comparación con el escalador de tiempo directo.

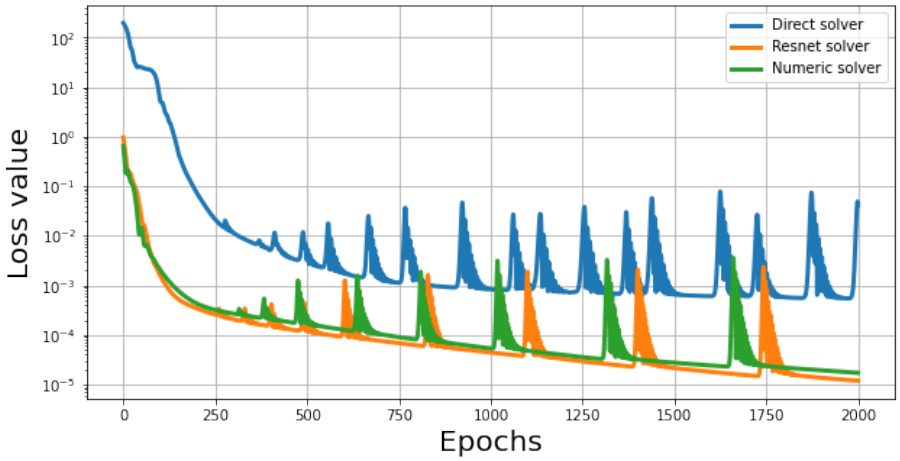


Figura 33.- Evolución de la función de pérdida con las tres configuraciones de la red propuestas.

La Fig. 34 ilustra una de las diez trayectorias de validación utilizadas para evaluar el rendimiento de las tres configuraciones propuestas. Se puede apreciar que los escaladores de tiempo residual y numérico lograron reproducir de manera precisa la trayectoria de validación, mientras que el escalador de tiempo directo presenta una mayor diferencia en comparación. Este efecto es especialmente evidente en la trayectoria de concentración de biomasa. Estos resultados sugieren que los escaladores de tiempo residual y numérico tienen una mayor capacidad de reproducir el comportamiento dinámico en comparación con el escalador de tiempo directo.

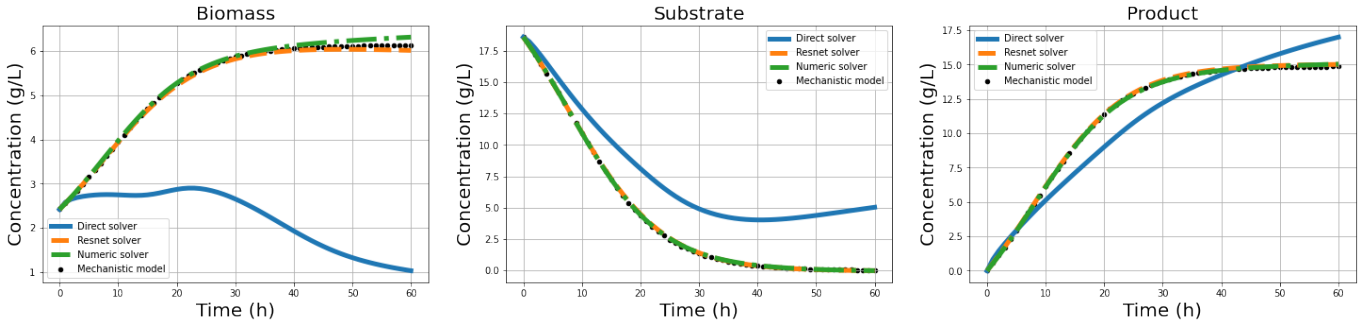


Figura 34.- Trayectoria de validación. Comparación entre la trayectoria de entrenamiento y las trayectorias predichas con las tres configuraciones de la red propuestos.

A partir de la comparación de las tres configuraciones propuestas, se puede concluir que las configuraciones de escaladores de tiempo residual y numérico lograron predecir adecuadamente las trayectorias de validación. Sin embargo, dado que el escalador de tiempo residual presentó un tiempo de entrenamiento significativamente menor en comparación con el escalador numérico,

se planea utilizar la configuración del escalador de tiempo residual para futuras simulaciones y análisis de optimización.

Inclusión del análisis en la aplicación web

La construcción de redes neuronales artificiales para simular procesos dinámicos de fermentación representa un paso hacia la adopción de técnicas de aprendizaje automático avanzadas en la industria química. Aunque este análisis demuestra el potencial de las redes neuronales en la modelización avanzada, su integración en la aplicación web se ha reservado para versiones posteriores. La razón de esta decisión se basa en la complejidad inherente a las redes neuronales y en la necesidad de una interfaz de usuario y una lógica de aplicación más sofisticadas. La inclusión de esta funcionalidad en etapas futuras permitirá un desarrollo y una implementación más reflexivos y centrados en el usuario, tras recibir feedback de la versión inicial de la aplicación.

4.4. Desarrollo de aplicación web para automatizar el análisis y la optimización de procesos de fermentación a nivel biorreactor

4.4.1. Diseño de la base de datos

La base de datos se encuentra estratégicamente estructurada en dos secciones principales. La primera sección comprende tablas encargadas de la gestión y almacenamiento de datos experimentales. A continuación, se proporciona una descripción detallada de dichas tablas.

Diseño de la base de datos: Sección de datos experimentales

La base de datos comprende tres tablas principales encargadas de la gestión y almacenamiento de los datos experimentales. Las tres tablas son:

1. `Experiment`.
2. `ExperimentVariable`.
3. `ExperimentVariableValue`.

Estas tablas capturan y almacenan la información esencial relacionada con los experimentos, variables y sus respectivos valores. A continuación, se presenta una descripción detallada de cada tabla.

Tabla `Experiment`

La tabla `Experiment` contiene registros que representan experimentos individuales realizados en el laboratorio. Esta tabla cuenta con los siguientes campos:

- `date` (`DateTime`): Fecha y hora de realización del experimento.
- `author` (`CharField`): Nombre del autor responsable del experimento.
- `supervisor` (`CharField`): Nombre del supervisor a cargo del experimento.
- `laboratory` (`CharField`): Designación del laboratorio en el que se llevó a cabo el experimento.
- `substrate` (`ForeignKey` a `Substrate`): Sustrato utilizado en el experimento.

- `microorganism` (`ForeignKey` a `Microorganism`): Microorganismo empleado en el experimento.
- `product` (`ForeignKey` a `Product`): Producto utilizado en el experimento.
- `experiment_type` (`CharField`): Tipo de experimento realizado, seleccionado de un conjunto predefinido de opciones.
- `observations` (`TextField`): Observaciones adicionales o notas relacionadas con el experimento.

Tabla `ExperimentVariable`

La tabla `ExperimentVariable` almacena metadatos relacionados con las variables experimentales medidas durante un experimento. Los campos que conforman esta tabla son:

- `experiment` (`ForeignKey` a `Experiment`): Experimento asociado con la variable.
- `variable_name` (`CharField`): Nombre de la variable.
- `variable_unit` (`CharField`): Unidades de medida de la variable.
- `variable_type` (`CharField`): Tipo de variable, puede ser discreta o continua.
- `detection_method` (`CharField`): Método de detección o medición de la variable.
- `observations` (`TextField`): Observaciones adicionales o notas referentes a la variable.

Tabla `ExperimentVariableValue`

La tabla ExperimentVariableValue guarda el conjunto de valores asociados a cada variable del experimento. Considerando que durante un experimento se pueden obtener múltiples observaciones de una misma variable, esta tabla está diseñada para almacenar todos estos valores. Los campos que integran esta tabla son:

- `variable` (`ForeignKey` a `ExperimentVariable`): Variable a la que se asocia el valor.
- `value` (`FloatField`): Valor medido de la variable.

- `value_type` (`CharField`): Indica si el valor es medido o se obtiene de fuentes bibliográficas.

A continuación, se describe la relación existente entre las tablas `Experiment`, `ExperimentVariable` y `ExperimentVariableValue`.

`Experiment` a `ExperimentVariable` (Relación Uno a Muchos): Un experimento puede implicar varias variables. Por lo tanto, la tabla `ExperimentVariable` tiene una referencia `ForeignKey` a la tabla `Experiment`, asociando cada variable a su respectivo experimento.

`ExperimentVariable` a `ExperimentVariableValue` (Relación Uno a Muchos): Una variable puede tener múltiples valores. Así, la tabla `ExperimentVariableValue` tiene una referencia `ForeignKey` a la tabla `ExperimentVariable`, asociando cada valor a su respectiva variable.

Las tablas `Experiment`, `ExperimentVariable` y `ExperimentVariableValue` proporcionan una estructura metódica para la gestión y almacenamiento de datos experimentales. Estas tablas almacenan la información requerida sobre los experimentos, las variables y sus valores. Las relaciones establecidas entre estas tablas facilitan la recuperación eficiente de los datos y su posterior análisis.

Diseño de la base de datos: Sección de análisis

La segunda sección de la base de datos se enfoca en el análisis de los datos. Esta sección comprende cinco tablas principales que gestionan y almacenan los datos relacionados con los análisis de los datos experimentales. Estas tablas son:

1. `Analysis`.
2. `AnalysisVariable`.
3. `AnalysisVariableValue`.
4. `AnalysisKineticParameter`.
5. `AnalysisKineticParameterValue`.

Estas tablas recopilan y almacenan información crítica relacionada con los análisis, sus variables, valores, parámetros cinéticos y los respectivos valores de estos parámetros. A continuación, se presenta una descripción detallada de cada una de estas tablas.

Tabla `Analysis`

La tabla `Analysis` almacena registros que representan análisis individuales vinculados a uno o varios experimentos. Los campos que integran esta tabla son:

- `experiments` (`ManyToManyField` a `Experiment`): Experimentos asociados al análisis.
- `date` (`DateTime`): Fecha y hora de realización del análisis.
- `author` (`CharField`): Autor del análisis.
- `supervisor` (`CharField`): Supervisor del análisis.
- `analysis_type` (`CharField`): Tipo de análisis realizado, seleccionado de un conjunto predefinido de opciones.
- `observations` (`TextField`): Observaciones adicionales o notas relacionadas con el análisis.

Tabla `AnalysisVariable`

La tabla `AnalysisVariable` contiene metadatos asociados a las variables consideradas durante un análisis. Esta tabla incluye los siguientes campos:

- `analysis` (`ForeignKey` a `Analysis`): Análisis asociado con la variable.
- `variable_name` (`CharField`): Nombre de la variable.
- `variable_units` (`CharField`): Unidad de medida de la variable.
- `variable_type` (`CharField`): Categoría de la variable, puede ser discreta o continua.
- `observations` (`TextField`): Observaciones adicionales o notas sobre la variable.

Tabla `AnalysisVariableValue`

La tabla `AnalysisVariableValue` almacena los valores asignados a cada variable del análisis. En esta tabla, se registran múltiples observaciones de una misma variable. Los campos de esta tabla son:

- `variable` (`ForeignKey` a `AnalysisVariable`): Variable a la que corresponde el valor.

- `value` (`FloatField`): Valor de la variable.
- `value_type` (`CharField`): Indica la clasificación del valor (inferior, superior, promedio, desviación, óptimo).

Tabla `AnalysisKineticParameter`

La tabla `AnalysisKineticParameter` almacena los parámetros cinéticos derivados de un análisis. Cada entrada en esta tabla representa un parámetro cinético individual y contiene los siguientes campos:

- `analysis` (`ForeignKey` a `Analysis`): Análisis al que pertenece el parámetro cinético.
- `parameter_name` (`CharField`): Nombre del parámetro cinético.
- `parameter_units` (`CharField`): Unidades del parámetro cinético.
- `observations` (`TextField`): Observaciones adicionales o notas sobre el parámetro cinético.

Tabla `AnalysisKineticParameterValue`

La tabla `AnalysisKineticParameterValue` está diseñada para almacenar los valores obtenidos para cada parámetro cinético durante un análisis. Al igual que en las tablas de valor de variable, pueden registrarse múltiples observaciones de un mismo parámetro cinético. Los campos que comprende esta tabla son:

- `parameter` (`ForeignKey` a `AnalysisKineticParameter`): Parámetro cinético al que corresponde el valor.
- `value` (`FloatField`): Valor del parámetro cinético.
- `value_type` (`CharField`): Indica la clasificación del valor (inferior, superior, promedio, desviación, óptimo).

A continuación, se describe la relación entre las tablas `Analysis`, `AnalysisVariable`, `AnalysisVariableV` `AnalysisKineticParameter` y `AnalysisKineticParameterValue`.

`Analysis` a `AnalysisVariable` (Relación Uno a Muchos): Un análisis puede implicar varias variables. Por lo tanto, la tabla `AnalysisVariable` tiene una referencia `ForeignKey` a la tabla `Analysis`, asociando cada variable a su respectivo análisis.

`AnalysisVariable` a `AnalysisVariableValue` (Relación Uno a Muchos): Una variable de análisis puede tener múltiples valores. Así, la tabla `AnalysisVariableValue` tiene una referencia `ForeignKey` a la tabla `AnalysisVariable`, asociando cada valor a su respectiva variable de análisis.

`Analysis` a `AnalysisKineticParameter` (Relación Uno a Muchos): Un análisis puede derivar en varios parámetros cinéticos. De este modo, la tabla `AnalysisKineticParameter` tiene una referencia `ForeignKey` a la tabla `Analysis`, asociando cada parámetro cinético a su respectivo análisis.

`AnalysisKineticParameter` a `AnalysisKineticParameterValue` (Relación Uno a Muchos): Un parámetro cinético puede tener múltiples valores. Así, la tabla `AnalysisKineticParameterValue` tiene una referencia `ForeignKey` a la tabla `AnalysisKineticParameter`, asociando cada valor a su respectivo parámetro cinético.

Las tablas `Analysis`, `AnalysisVariable`, `AnalysisVariableValue`, `AnalysisKineticParameter` y `AnalysisKineticParameterValue` proporcionan una estructura sistemática para la gestión y almacenamiento de datos de análisis. Estas tablas almacenan la información necesaria sobre los análisis, las variables de análisis y sus valores, y los parámetros cinéticos y sus valores. Las relaciones establecidas entre estas tablas permiten una recuperación eficiente de los datos y facilitan su análisis posterior.

A continuación, se presenta el esquema de la base de datos implementado para la aplicación:

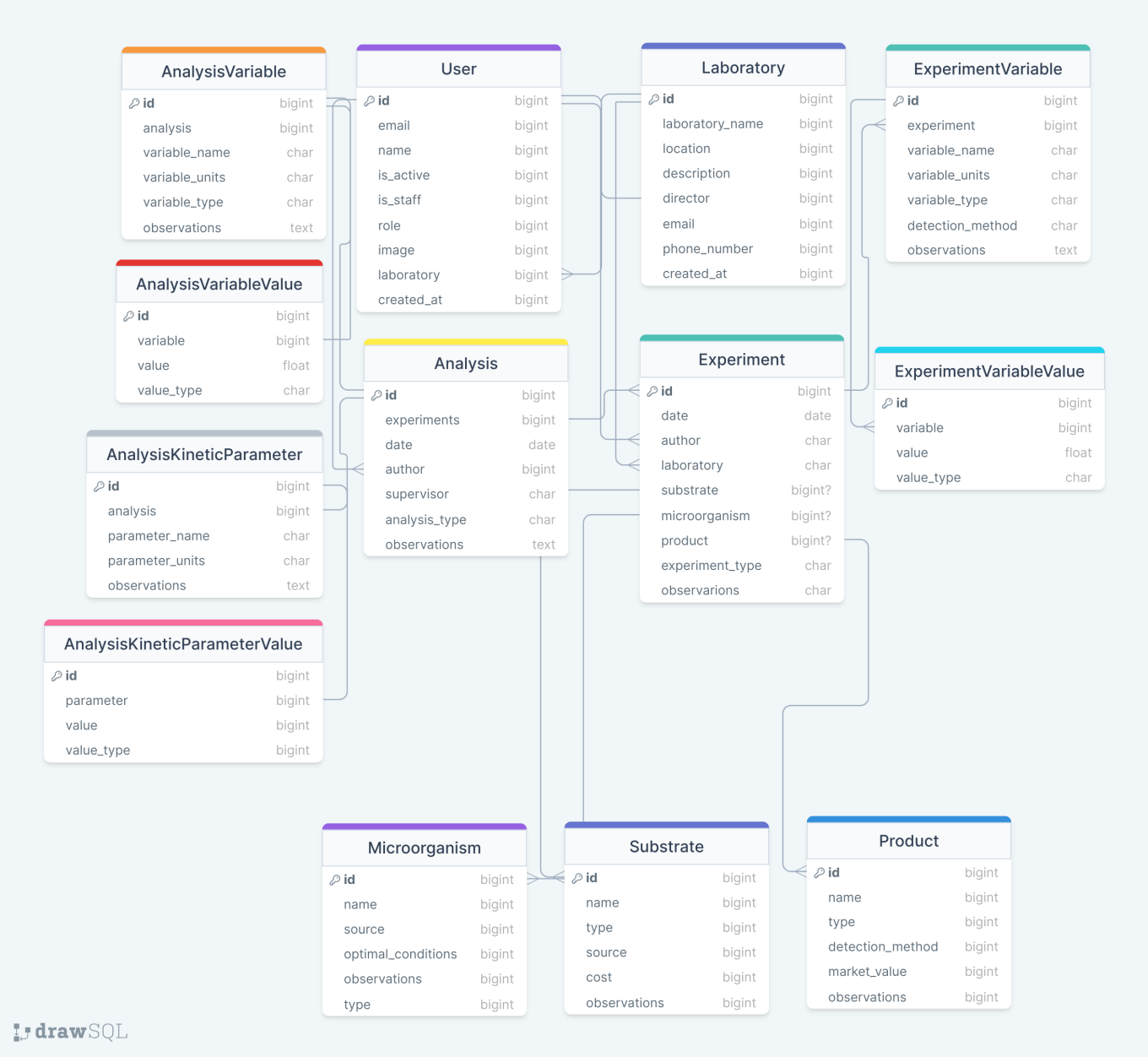


Figura 35.- Esquema del diseño de la base de datos de la aplicación. Imagen creada por el autor usando drawSQL.

4.4.2. Diseño de la interfaz de usuario

Como se mencionó en la metodología, el diseño de la aplicación web se compone de cuatro partes principales:

1. Encabezado con Navegación: El encabezado se colocará en la parte superior de cada página. Contendrá el logo de la aplicación a la izquierda, reforzando la identidad de la marca, y una

barra de navegación a la derecha. La barra de navegación incluirá enlaces a *Home*, *Login*, *Account*, *Experiments*, *Laboratories*, *Entities*, *Analysis* y *Users*. Esto facilitará la navegación a las secciones clave de la aplicación. Para los usuarios que no han iniciado sesión, solo podrán acceder a la sección de *Login*, y necesitarán iniciar sesión para acceder al resto de la aplicación.

2. Panel de la barra lateral: Un panel de la barra lateral proporcionará controles y opciones para configurar el análisis estadístico, la visualización, el llenado de formularios y el filtrado de tablas. Se diseñará con controles fáciles de usar, como controles deslizantes, casillas de verificación y menús desplegables, ofreciendo a los usuarios un buen grado de control sobre los datos con los que están trabajando. El usuario podrá ocultar el panel de la barra lateral si lo necesita.

3. Panel principal: El panel principal mostrará los resultados según la entrada del usuario desde el panel de la barra lateral. Tendrá un diseño flexible para acomodar varios tipos de representación de datos, incluidos gráficos, resultados de análisis estadísticos y tablas con datos experimentales. El diseño se ajustará dinámicamente en función de los datos que se muestren.

4. Modales: Los modales se usarán para mostrar datos adicionales o capturar la entrada del usuario sin interrumpir la estructura general de la aplicación. Se utilizarán para tareas como ingresar nuevos datos de experimentos, confirmar acciones o mostrar información adicional relacionada con los datos presentados en el panel principal.

La aplicación web contará con las siguientes secciones:

1. *Home*: La página de inicio de la aplicación, aquí se mostrará información específica del usuario, así como la opción de cerrar sesión.

2. *Login*: Una página de inicio de sesión segura para que los usuarios accedan a sus cuentas antes de tener acceso al resto de la aplicación.

3. *Account*: Aquí, los usuarios pueden administrar la configuración de su cuenta y ver sus perfiles.

4. *Experiments*: Una base de datos de todos los experimentos realizados por el usuario, con capacidades de búsqueda, ordenación y filtrado.

5. *Laboratories*: Esta sección permite el registro a acceso a laboratorios.

6. *Entities*: Esta sección permite el registro a acceso a entidades tales como microorganismos, sustratos y productos.

7. *Analysis*: Los usuarios pueden ingresar datos de experimentos cinéticos y llevar a cabo la estimación de parámetros cinéticos. Además, en esta sección el usuario puede introducir datos de diseños experimentales y llevar a cabo análisis estadísticos y de optimización.

8. *Users*: Una sección de administración para que los administradores administren las cuentas y los permisos de los usuarios.

A continuación, se muestra un esquema del diseño de la aplicación web:

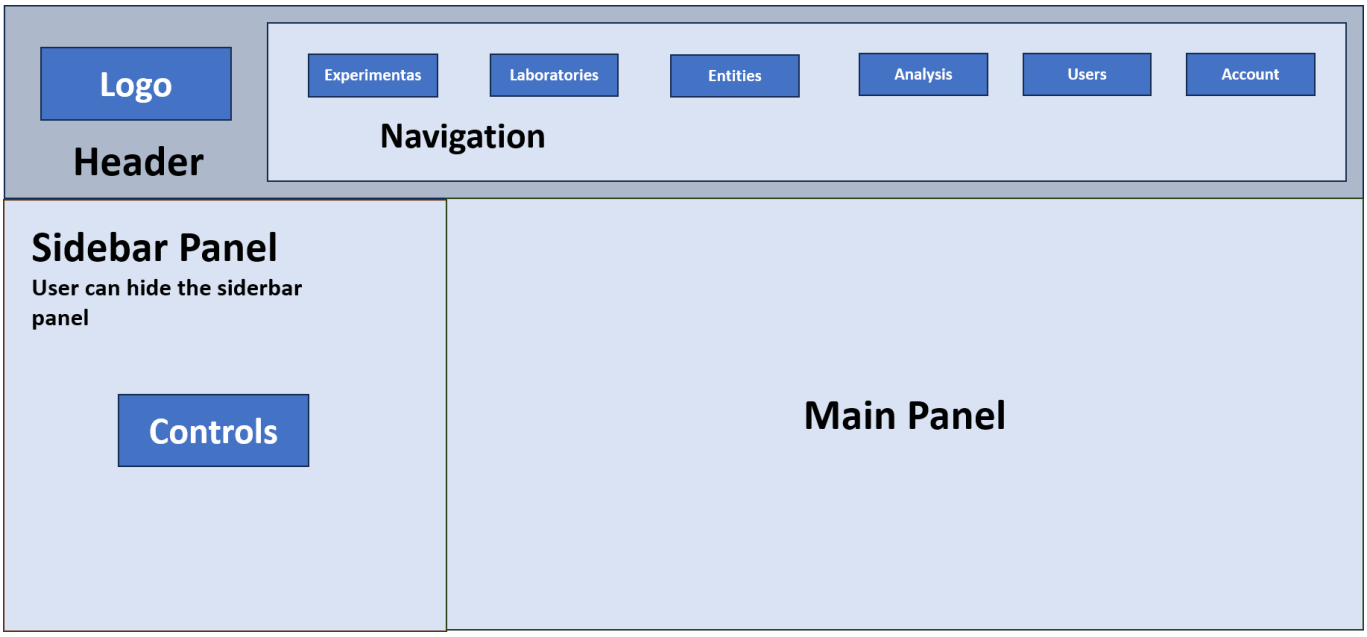


Figura 36.- Diseño de la aplicación web. En el esquema es posible apreciar todas las secciones de la aplicación web, así como su ubicación en la interfaz de usuario.

4.4.3. Capturas de pantalla de la aplicación web

A continuación se muestra una imagen de la aplicación en la sección *Home*

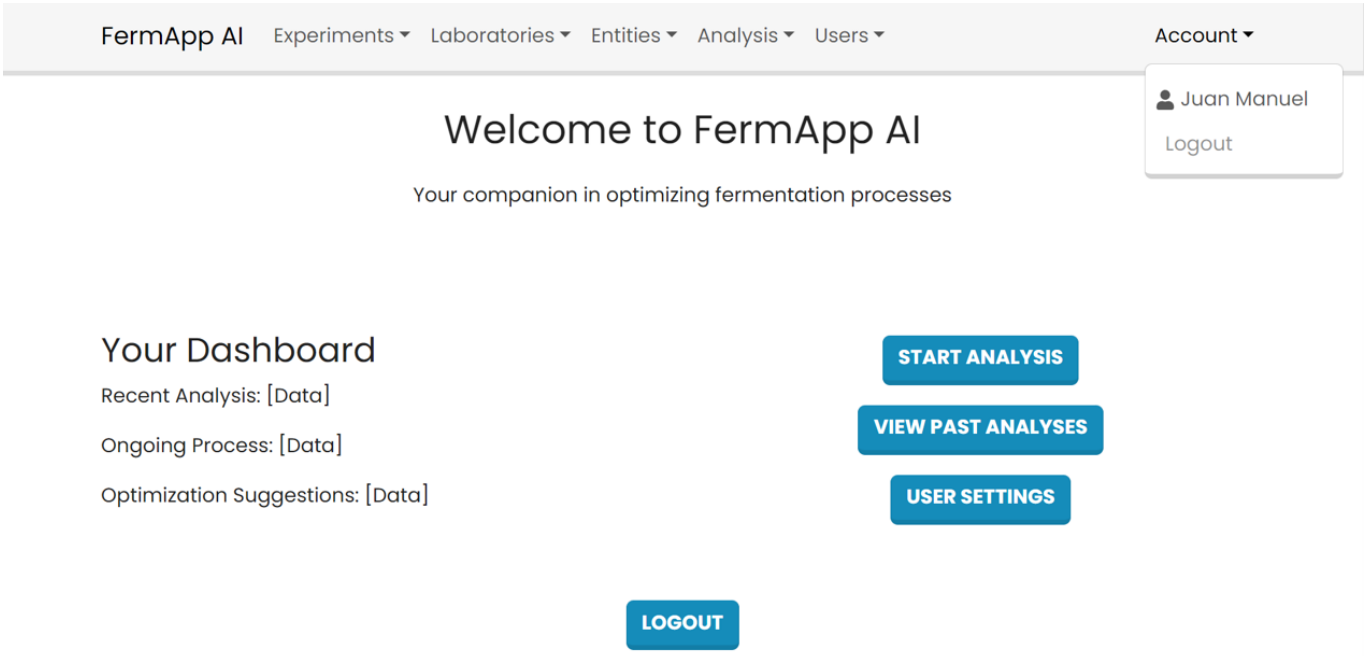


Figura 37.- Interfaz Principal de la Aplicación Web FermApp®.

La captura de pantalla corresponde a la sección *Home* de la aplicación web FermApp®, diseñada para asistir en la optimización de procesos de fermentación. En el encabezado superior, se muestra el logotipo de FermApp®, flanqueado por un menú de navegación que incluye accesos directos a las secciones *Experiments*, *Laboratories*, *Entities*, *Analysis*, y *Users*. Este menú ofrece al usuario una navegación fluida y directa a las funcionalidades principales de la plataforma. En la esquina superior derecha, se encuentra el área de *Account*, que indica al usuario activo, 'Juan Manuel', con opciones para gestionar su cuenta o cerrar sesión, enfatizando la personalización de la experiencia del usuario.

Centrándonos en el cuerpo principal de la interfaz, encontramos un mensaje de bienvenida que enfatiza la intención de la aplicación de ser un aliado en la optimización de la fermentación, subrayando el carácter interactivo y enfocado al usuario de la herramienta. A continuación, se despliega el *Dashboard*, el cual sirve como el epicentro de la actividad del usuario, presentando enlaces rápidos y eficientes para *START ANALYSIS*, *VIEW PAST ANALYSES*, y *USER SETTINGS*. Estos enlaces invitan al usuario a embarcarse en nuevas tareas analíticas, revisar análisis previos o personalizar la configuración de su perfil.

A continuación de muestra una captura de pantalla de la aplicación en la sección *Parameter Estimation*

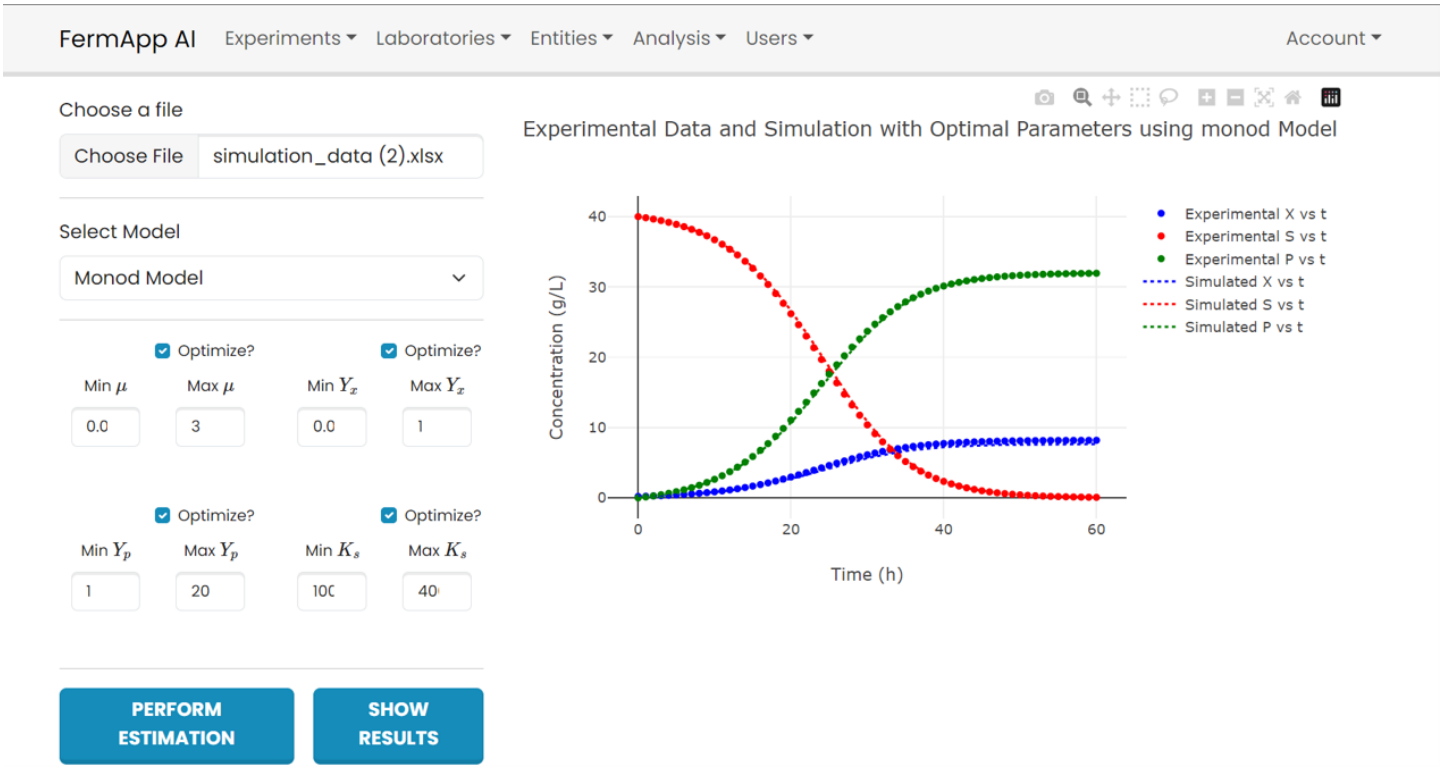


Figura 38.- Interfaz de Estimación de Parámetros con Modelo Monod en FermApp®.

Esta captura de pantalla muestra la sección de estimación de parámetros cinéticos de la aplicación web FermApp®, donde los usuarios pueden cargar datos experimentales y simular procesos de fermentación utilizando el modelo Monod. En la parte superior de la página, la barra de navegación proporciona acceso rápido a las secciones *Experiments*, *Laboratories*, *Entities*, *Analysis*, y *Users*, facilitando la navegación entre las diversas funcionalidades de la aplicación.

En la sección central izquierda, el usuario tiene la opción de *Choose File* para cargar los datos de la simulación en formato xlsx. Justo debajo, en la sección *Select Model*, el usuario puede seleccionar *Monod Model* de un menú desplegable, dado que la aplicación tiene la capacidad para trabajar con múltiples modelos de estimación. A continuación, se proporciona una serie de parámetros cinéticos con la opción de *Optimize*, permitiendo al usuario establecer los rangos mínimos y máximos para la tasa de crecimiento máximo (μ), el coeficiente de rendimiento de sustrato en biomasa (Y_{xs}), el coeficiente de rendimiento de producto en biomasa (Y_{px}), y la constante de afinidad del sustrato (K_s). Esto indica la funcionalidad de optimización de la aplicación, que puede ajustar automáticamente estos parámetros para encontrar el mejor ajuste para los datos experimentales.

En la parte central derecha, se presenta un gráfico de dispersión titulado *Experimental Data and Simulation with Optimal Parameters using monod Model*. Este gráfico visualiza tanto los datos experimentales como la simulación resultante, con puntos azules, rojos y verdes que representan las concentraciones experimentales de biomasa (X), sustrato (S) y producto (P) en función del tiempo. Las líneas discontinuas de colores correspondientes muestran los resultados simulados para cada una de estas concentraciones, permitiendo una comparación directa entre los datos y la simulación. Esta representación visual de los datos proporciona una herramienta intuitiva para analizar la precisión de la estimación de parámetros y el ajuste del modelo.

Al pie de la sección de parámetros se encuentran dos botones prominentes, *PERFORM ESTIMATION* y *SHOW RESULTS*, lo que indica las acciones que el usuario puede realizar: ejecutar la estimación de parámetros y visualizar los resultados de la simulación, respectivamente.

A continuación se muestra una imagen de la aplicación en la sección *Linear Regression*

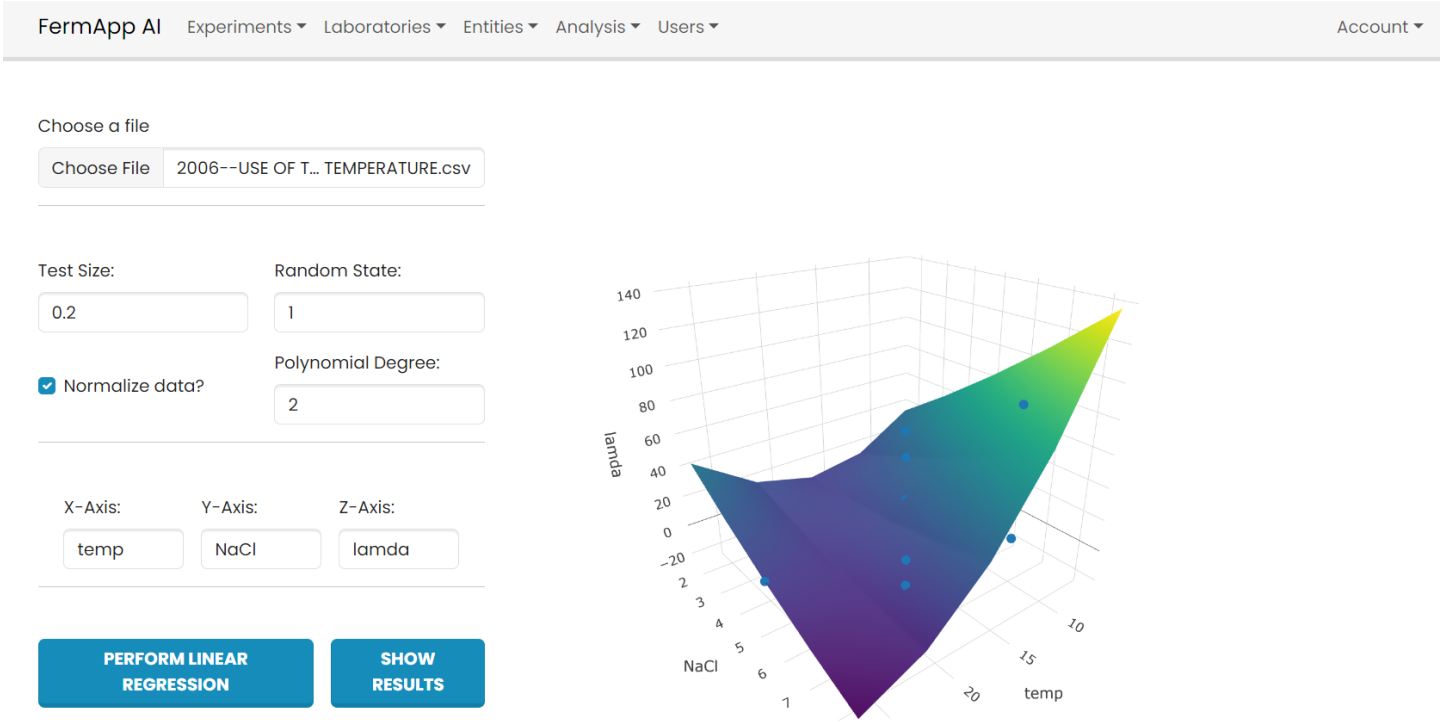


Figura 39.- Interfaz de Análisis de Regresión Lineal en FermApp®.

La Figura muestra la interfaz de usuario para la sección de análisis de regresión lineal en la aplicación web FermApp®. Se observa la barra de navegación superior que proporciona acceso a distintas funcionalidades de la plataforma, tales como *Experiments*, *Laboratories*, *Entities*, *Analysis*,

Users, y una opción para acceder a la cuenta del usuario. Central en la interfaz es el módulo de carga de datos, donde el usuario ha seleccionado el archivo '2006–USE O...ERATURE.csv' para su análisis. Justo debajo, se encuentran los controles para parametrizar el análisis de regresión, incluyendo el tamaño de la prueba (*Test Size*), el estado aleatorio (*Random State*), y el grado del polinomio (*Polynomial Degree*), con una casilla de verificación activada para normalizar los datos. Los campos para especificar las variables de los ejes *X-Axis*, *Y-Axis*, y *Z-Axis* están configurados respectivamente con las variables *temp*, *NaCl*, y *lamda*. En la parte inferior de la interfaz, los botones *PERFORM LINEAR REGRESSION* y *SHOW RESULTS* permiten ejecutar el análisis y mostrar los resultados, respectivamente. El gráfico a la derecha ilustra una superficie de respuesta tridimensional generada a partir del análisis, destacando la relación entre las variables seleccionadas y la respuesta modelada.

4.5. Casos de estudio para el análisis y la optimización de procesos fermentativos mediante FermApp®

La aplicación FermApp® ha sido utilizada para el análisis de datos experimentales de dos cinéticas de fermentación, resultando en la estimación eficiente de parámetros cinéticos utilizando el modelo de Monod. Los datos obtenidos y procesados por FermApp® se discuten a continuación.

Análisis de la Producción de Compuestos por *Gibberella fujikuroi* En el primer caso, el análisis realizado con FermApp® sobre los datos del estudio de la producción de compuestos secundarios mediante *Gibberella fujikuroi* arrojó los siguientes parámetros óptimos, como se detalla en el trabajo de Rodríguez (2006):

- Tasa máxima específica de crecimiento (μ_{max}): 0.140 h^{-1}
- Coeficiente de rendimiento de sustrato en biomasa (Y_x): 0.067 g/g
- Coeficiente de rendimiento de producto en biomasa (Y_p): 1.673 g/g
- Constante de afinidad de sustrato (K_s): 312.671 g/L

A continuación se muestra una imagen de la comparación entre los datos experimentales y la simulación realizada con los parámetros óptimos generada con la aplicación FermApp®.

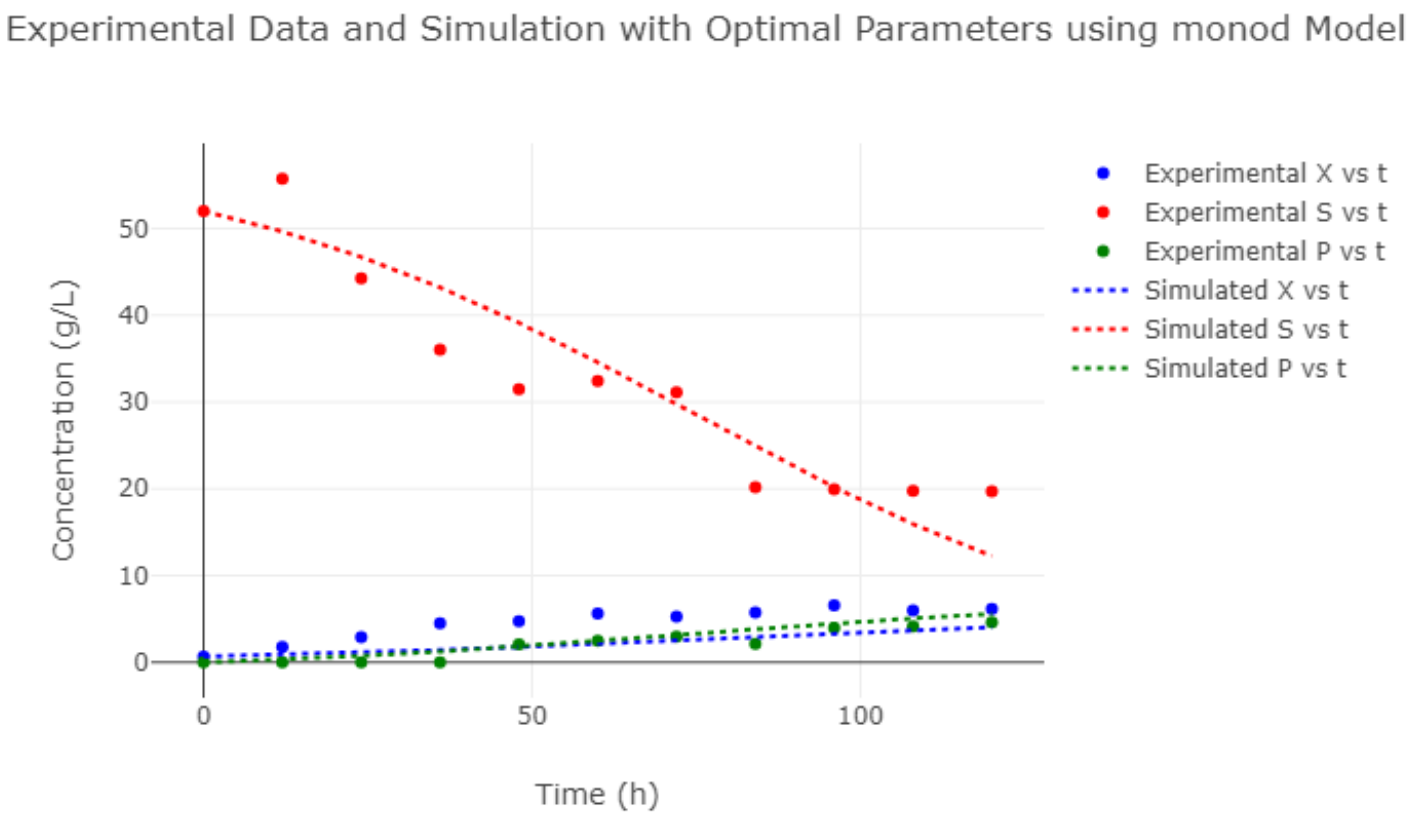


Figura 40.- Datos experimentales de la producción de compuestos por *Gibberella fujikuroi* y simulación del modelo de Monod con parámetros óptimos.

Estos valores sugieren una tasa de crecimiento moderada con un rendimiento de biomasa relativamente bajo en comparación con la producción. El alto valor de Y_p indica una producción de producto considerable por unidad de biomasa. La constante K_s elevada puede interpretarse como una baja afinidad del microorganismo por el sustrato, lo cual podría ser indicativo de la necesidad de altas concentraciones de sustrato para alcanzar la tasa máxima de crecimiento. El error cuadrático medio (MSE) de 28.751, aunque indica ciertas discrepancias entre los datos experimentales y el modelo, refleja una correlación aceptable, teniendo en cuenta la complejidad inherente a los sistemas biológicos. Como se observa en la Figura 41, los datos de producción de etanol y las simulaciones exhiben un alto nivel de concordancia, demostrando la eficacia del modelo para replicar tendencias experimentales reales.

Evaluación de Parámetros Cinéticos en Fermentaciones de Xilitol con la Aplicación FermApp®

En el segundo conjunto de datos, proporcionados por Martínez Corona (2013), FermApp® determinó los siguientes parámetros cinéticos:

- Tasa máxima específica de crecimiento (μ_{max}): 2.342 h^{-1}
- Coeficiente de rendimiento de sustrato en biomasa (Y_x): 0.076 g/g
- Coeficiente de rendimiento de producto en biomasa (Y_p): 9.911 g/g
- Constante de afinidad de sustrato (K_s): 361.687 g/L

A continuación se muestra una imagen de la comparación entre los datos experimentales y la simulación realizada con los parámetros óptimos generada con la aplicación FermApp®.

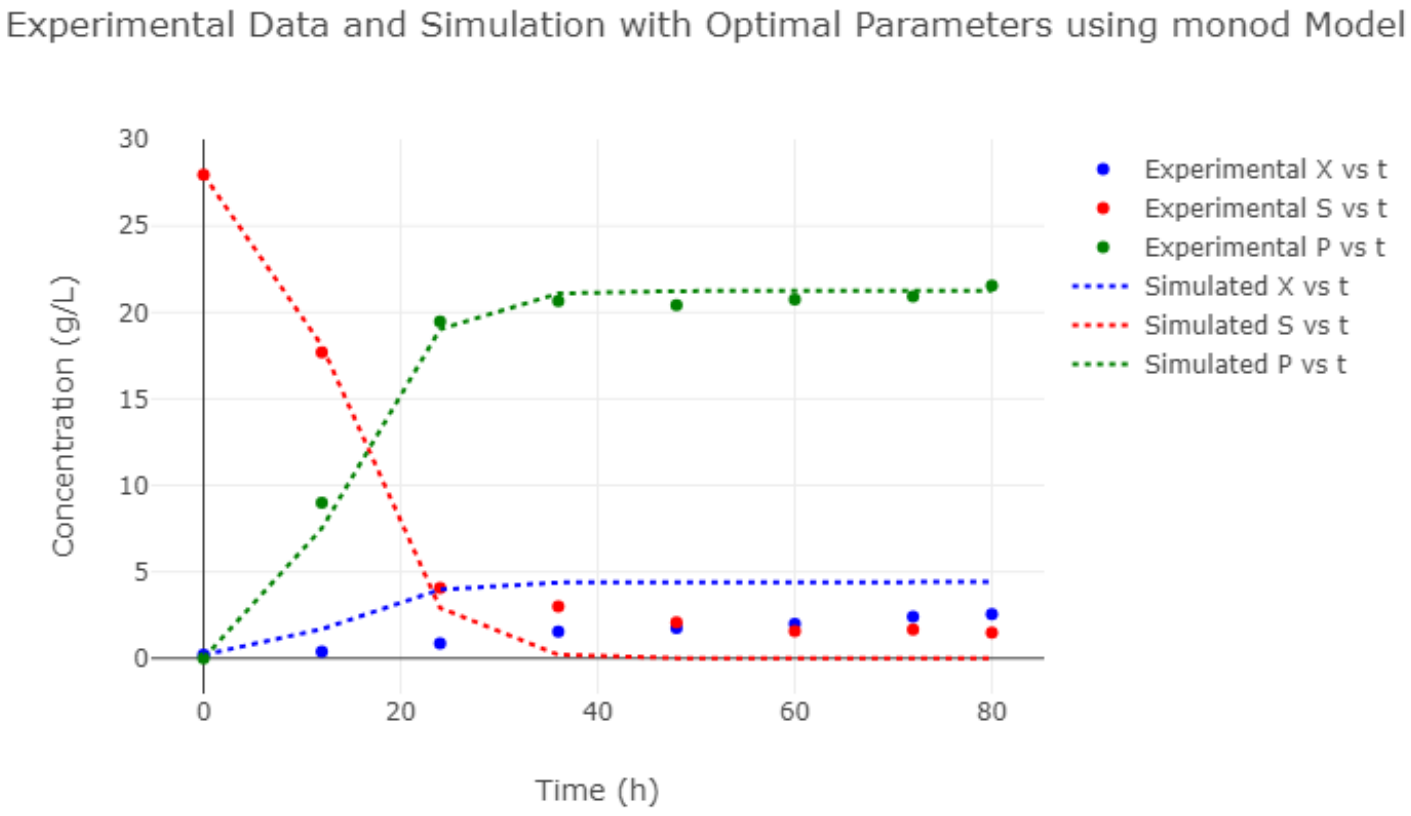


Figura 41.- Datos experimentales de fermentaciones de xilitol y simulación del modelo de Monod con parámetros óptimos.

Este conjunto de parámetros revela una tasa de crecimiento significativamente alta, lo que implica una robusta capacidad de crecimiento del microorganismo bajo las condiciones de fermentación analizadas. El coeficiente Y_p extremadamente alto señala una producción de producto excepcionalmente eficiente en relación con la biomasa generada. Similar al primer análisis, la constante K_s grande sugiere una baja afinidad por el sustrato. El MSE de 3.317 es notablemente bajo, lo que indica un excelente ajuste del modelo a los datos experimentales y una alta fiabilidad de los parámetros estimados por FermApp®.

Consideraciones Generales La comparación de los resultados de ambos análisis ilustra la versatilidad de FermApp® en el manejo de variados perfiles cinéticos. La eficacia de la aplicación se manifiesta en su capacidad para determinar parámetros cinéticos que reflejan las diferencias fundamentales entre las dos fermentaciones estudiadas. Aunque ambos casos presentan constantes de afinidad elevadas, las tasas de crecimiento y los coeficientes de rendimiento varían notablemente, resaltando las distintas estrategias metabólicas de los microorganismos o las condiciones de operación en cada proceso.

La utilización de FermApp® permite un análisis detallado y confiable que facilita la interpretación de dinámicas complejas en procesos de fermentación. Esto es crucial para la optimización de los procesos y el diseño de estrategias de control y escalado industrial. En ambos casos, a pesar de la complejidad inherente a los procesos biológicos, la aplicación ha demostrado ser una herramienta potente para el análisis y simulación de procesos de fermentación, capaz de proporcionar *insights* valiosos para la investigación y la industria.

5. Conclusiones

La utilización de algoritmos genéticos se ha demostrado efectiva en la estimación precisa y rápida de los parámetros cinéticos y sus intervalos de confianza en fermentaciones en biorreactores tipo lote. Además de esto, se plantea integrar el código y los modelos utilizados en una aplicación web para automatizar el análisis.

En cuanto a la optimización del flujo de alimentación, se ha demostrado que es posible utilizar algoritmos genéticos para llevar a cabo esta tarea de manera eficiente. Sin embargo, se debe tener en cuenta que la solución obtenida solo consideró una discretización del flujo de alimentación en 10 intervalos y el tiempo de cómputo requerido para una discretización mayor podría ser prohibitivo. Por lo tanto, se propone reemplazar el modelo mecanístico utilizado por una red neuronal con el fin de generar un perfil óptimo con un mayor número de intervalos y aumentar la producción de biomasa.

En cuanto a las configuraciones de las redes neuronales propuestas, se ha concluido que las configuraciones de escaladores de tiempo residual y numérico lograron predecir adecuadamente las trayectorias de validación y podrían ser utilizadas en la simulación de un biorreactor tipo lote-alimentado y en la optimización del flujo de alimentación. Sin embargo, se planea utilizar la configuración del escalador de tiempo residual debido a su menor tiempo de entrenamiento para futuras simulaciones y análisis de optimización.

En relación al desarrollo de la aplicación web, es posible concluir que el diseño de la base de datos propuesto para la aplicación es sólido y robusto. Las secciones de datos experimentales y análisis permiten almacenar y gestionar de forma eficiente una amplia gama de datos provenientes de experimentos realizados en el laboratorio. Los tres componentes principales del área de datos experimentales, es decir, la tabla `Experiment`, `ExperimentVariable` y `ExperimentVariableValue`, proporcionan una estructura de datos bien organizada y fácilmente accesible. Lo mismo puede decirse de la sección de análisis, donde las tablas `Analysis`, `AnalysisVariable`, `AnalysisVariableValue`, `AnalysisKineticParameter` y `AnalysisKineticParameterValue` ofrecen un enfoque metódico para manejar y analizar los datos obtenidos a partir de los experimentos.

Por otro lado, el diseño de la interfaz de usuario (UI) de la aplicación web es intuitivo y

amigable, asegurando una buena experiencia para el usuario. El esquema de la aplicación permite una fácil navegación, lo que resulta en una mayor eficiencia y satisfacción del usuario. Las cuatro partes principales de la interfaz de usuario, incluyendo el Encabezado con Navegación, Panel de la barra lateral, Panel principal y Modales, contribuyen a la interactividad y funcionalidad de la aplicación. Además, la existencia de múltiples secciones, como *Home*, *Login*, *Account*, *Parameter Estimation*, *Process optimization*, *Visualization*, *Experiments*, *Analyses* y *Users*, garantizan una gestión integral y completa de los datos y procesos experimentales. En general, el diseño de la aplicación web y su base de datos ofrecen un entorno potente y flexible para realizar y analizar experimentos de laboratorio de una manera eficaz y eficiente.

6. Perspectivas

1. Incorporación de análisis con redes neuronales artificiales (ANN):

   - Desarrollar e integrar modelos basados en ANN implica diseñar estructuras computacionales inspiradas en el funcionamiento del cerebro humano para modelar complejas relaciones entre variables de entrada y salida. Estos modelos pueden ser particularmente útiles en la predicción de parámetros y en la optimización de procesos donde las relaciones no son lineales o son altamente complejas.
   - Al explorar diferentes arquitecturas de ANN, como redes densamente conectadas, recurrentes o convolucionales, puedes determinar cuál es más adecuada para la simulación de procesos específicos en biorreactores. Cada arquitectura tiene sus fortalezas, y la selección depende del tipo de datos y del proceso que se está modelando. El rendimiento se puede medir en términos de precisión predictiva, capacidad de generalización y eficiencia computacional.

2. Análisis de sensibilidad y optimización:

   - Un análisis de sensibilidad te permite identificar qué parámetros tienen el mayor impacto en los resultados del modelo. Esto es útil no solo para entender el sistema modelado sino también para guiar el diseño de experimentos y para priorizar esfuerzos en la medición y control de ciertas variables.
   - En cuanto a la optimización, puedes utilizar técnicas de optimización para ajustar el modelo de regresión lineal y encontrar las condiciones de operación que maximizan o minimizan una función objetivo, como el rendimiento del producto, la calidad o el coste operativo. La optimización puede llevarse a cabo mediante algoritmos determinísticos o estocásticos y puede requerir la integración de simulaciones de proceso dentro de un bucle de optimización.

Referencias

Alvarez-Ainza, M. L., García-Galaz, A., González-Ríos, H., Moreno-Ibarra, G. M., Torre-Martínez, M. D. l., Zamora-Quiñones, K. A., y Acedo-Félix, E. (2021). Characterization and selection of native yeast isolated from natural fermentation for the production of the artisanal beverage bacanora. *Biotecnia*, *23*(1), 21–27.

Amenaghawon, N., Okieimen, C., y Ogbeide, S. (2012). Kinetic modelling of ethanol inhibition during alcohol fermentation of corn stover using saccharomyces cerevisiae. *Int. J. Eng. Res. Appl*, *4*, 798–803.

Andrews, J. F. (1968). A mathematical model for the continuous culture of microorganisms utilizing inhibitory substrates. *Biotechnology and bioengineering*, *10*(6), 707–723.

Angelova, M., Tzonkov, S., y Pencheva, T. (2010). Parameter identification of a fed-batch cultivation of s. cerevisiae using genetic algorithms. *Serdica Journal of Computing*, *4*(1), 11p–18p.

Ariyajaroenwong, P., Laopaiboon, P., Salakkam, A., Srinophakun, P., y Laopaiboon, L. (2016). Kinetic models for batch and continuous ethanol fermentation from sweet sorghum juice by yeast immobilized on sweet sorghum stalks. *Journal of the Taiwan Institute of Chemical Engineers*, *66*, 210–216.

Bailey, J. E. (1998). Mathematical modeling and analysis in biochemical engineering: past accomplishments and future opportunities. *Biotechnology progress*, *14*(1), 8–20.

Bin Mohd Zain, M. Z., Kanesan, J., Kendall, G., y Chuah, J. H. (2018). Optimization of fed-batch fermentation processes using the backtracking search algorithm. *Expert Systems with Applications*, *91*, 286–297.

Blickle, T., y Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, *4*(4), 361–394.

Chen, W. W., Niepel, M., y Sorger, P. K. (2010). Classic and contemporary approaches to modeling biochemical reactions. *Genes & development*, *24*(17), 1861–1875.

Cinar, A., Parulekar, S. J., Undey, C., y Birol, G. (2003). *Batch fermentation: modeling: monitoring, and control*. CRC press.

Contois, D. (1959). Kinetics of bacterial growth: relationship between population density and specific growth rate of continuous cultures. *Microbiology*, *21*(1), 40–50.

Cortez, P., y Cortez. (2014). *Modern optimization with r*. Springer.

Cuevas, E., Barocio Espejo, E., y Conde Enríquez, A. (2019). Introduction to metaheuristics methods. En *Metaheuristics algorithms in power systems* (pp. 1–8). Springer.

De Ovalle, S., Cavello, I., Brena, B. M., Cavalitto, S., y González-Pombo, P. (2018). Production and characterization of a β-glucosidase from issatchenkia terricola and its use for hydrolysis of aromatic precursors in cabernet sauvignon wine. *Lwt*, *87*, 515–522.

Dobbelaere, M. R., Plehiers, P. P., Van de Vijver, R., Stevens, C. V., y Van Geem, K. M. (2021). Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats. *Engineering*, *7*(9), 1201–1211.

Dutta, S. (2016). *Optimization in chemical engineering*. Cambridge University Press.

Gao, H., Zhu, L.-T., Luo, Z.-H., Fraga, M. A., y Hsing, I.-M. (2022). *Machine learning and data science in chemical engineering* (Vol. 61) (n.º 24). ACS Publications.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, *13*(5), 533–549.

Glover, F. (1997). Tabu search and adaptive memory programming—advances, applications and challenges. *Interfaces in computer science and operations research*, 1–75.

Glover, F., y Laguna, M. (1998). Tabu search. En *Handbook of combinatorial optimization* (pp. 2093–2229). Springer.

Hedar, A.-r., y Ahmed, A. (2004). Studies on metaheuristics for continuous global optimization problems.

Hemmerich, J., Tenhaef, N., Wiechert, W., y Noack, S. (2021). pyfoomb: Python framework for object oriented modeling of bioprocesses. *Engineering in life sciences*, *21*(3-4), 242–257.

Hillier, F. S. (2010). Introduccion a la investigacion de operaciones.[book auth.] frederick s. hillier and gerald j. lieberman. *Introduction to operations research*, *9*, 978.

Himmelblau, D. M., Lasdon, L. S., y Edgar, T. F. (2001). *Optimization of chemical processes*. McGraw-Hill.

Introducción a la programación en python 1: Aprendiendo a programar en python - coursera. (s.f.). Descargado de https://www.coursera.org/learn/aprendiendo-programar-python

Introducción al lenguaje de programación python. (s.f.). Descargado de https://aprendizajeprofundo.github.io/Libro_Fundamentos_Programacion/Python/Cuadernos/py_01.html

Introducción al lenguaje de programación python (3º edición). (s.f.). Descargado de https://openlibra.com/es/book/introduccion-al-lenguaje-de-programacion-python-3o-edicion

Introducción — documentación de python - 3.11.2. (s.f.). Descargado de https://docs.python.org/es/3/reference/introduction.html

Joelianto, E., Dwihandayani, R., y Ling, L. (2001). Parameter estimation using genetic algorithm for fed-batch growth of saccharomyces cerevisiae. *IFAC Proceedings Volumes*, *34*(11), 226–230.

Lee, J., Pollard, J., y Coulman, G. (1983). Ethanol fermentation with cell recycling: computer simulation. *Biotechnology and bioengineering*, *25*(2), 497–511.

Lim, H. C., y Shin, H. S. (2013). *Fed-batch cultures: principles and applications of semi-batch bioreactors*. Cambridge University Press.

Lones, M. (2011). *Sean luke: essentials of metaheuristics.* Springer.

Martínez Corona, R. (2013). *Caracterización fermentativa de la producción de xilitol por Candida magnoliae utilizando hidrolizados de semilla de tamarindo como sustrato* (Tesis de maestría no publicada). Universidad Michoacana de San Nicolás de Hidalgo, Morelia, Michoacán.

Mears, L., Stocks, S. M., Albaek, M. O., Sin, G., y Gernaey, K. V. (2017). Mechanistic fermentation models for process design, monitoring, and control. *Trends in biotechnology*, *35*(10), 914–924.

Miller, B. L., Goldberg, D. E., y cols. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, *9*(3), 193–212.

Monod, J. (1949). The growth of bacterial cultures: Annual reviews in microbiology, v. 3.

Moon, J., Gbadago, D. Q., Hwang, G., Lee, D., y Hwang, S. (2022). Software platform for high-fidelity-data-based artificial neural network modeling and process optimization in chemical engineering. *Computers & Chemical Engineering*, *158*, 107637.

Mowbray, M., Savage, T., Wu, C., Song, Z., Cho, B. A., Del Rio-Chanona, E. A., y Zhang, D. (2021). Machine learning for biochemical engineering: A review. *Biochemical Engineering Journal*, *172*, 108054.

Nesmachnow, S. (2014). An overview of metaheuristics: accurate and efficient methods for optimisation. *International Journal of Metaheuristics*, *3*(4), 320–347.

Oliveira, C., Jesus, C., Ceneviva, L., Silva, F., Cruz, A., Costa, C., y Badino, A. (2017). Anabioplus: a new package for parameter estimation and simulation of bioprocesses. *Brazilian Journal of Chemical Engineering*, *34*, 1065–1082.

Oteiza, P. P., Rodríguez, D. A., y Brignole, N. B. (2018). Parallel cooperative optimization through hyperheuristics. En *Computer aided chemical engineering* (Vol. 44, pp. 805–810). Elsevier.

Patel, N., y Padhiyar, N. (2016). Multi-objective optimal control study of fed-batch bio-reactor. *IFAC-PapersOnLine*, *49*(7), 97–102.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, *12*, 2825–2830.

Phan-thien, Y. (2011). Biomass v2.0: A new tool for bioprocess simulation..

Phisalaphong, M., Srirattana, N., y Tanthapanichakoon, W. (2006). Mathematical modeling to investigate temperature effect on kinetic parameters of ethanol fermentation. *Biochemical engineering journal*, *28*(1), 36–43.

Rao, S. S. (2019). *Engineering optimization: theory and practice*. John Wiley & Sons.

Ribas-García, M., Hurtado-Vargas, R., Domenech-López, F., y Garrido-Carralero, N. (2014). Verificación y validación del software fermenta 5.0 para la simulación de la fermentación alcohólica..

Rivera, E. C., Costa, A. C., Atala, D. I., Maugeri, F., Maciel, M. R. W., y Maciel Filho, R. (2006). Evaluation of optimization techniques for parameter estimation: application to ethanol fermentation considering the effect of temperature. *Process Biochemistry*, *41*(7), 1682–1687.

Rocha, M., Mendes, R., Rocha, O., Rocha, I., y Ferreira, E. C. (2014). Optimization of fed-batch fermentation processes with bio-inspired algorithms. *Expert Systems with Applications*, *41*(5), 2186–2195.

Rodriguez-Granrose, D., Jones, A., Loftus, H., Tandeski, T., Heaton, W., Foley, K. T., y Silverman, L. (2021). Design of experiment (doe) applied to artificial neural network architecture enables rapid bioprocess improvement. *Bioprocess and biosystems engineering*, *44*(6), 1301–1308.

Rodríguez, G., Abel Jerónimo y Vera Verduzco. (2006). *Producción de bikaverina en matraz agitado con Gibberella fujikuroi* (Tesis de licenciatura no publicada). Instituto Tecnológico de Morelia, Morelia, Michoacán. (Monografía)

Samal, K. K. R., Babu, K. S., y Das, S. K. (2022). Multi-output spatio-temporal air pollution forecasting using neural network approach. *Applied Soft Computing*, 109316.

Schweidtmann, A. M., Esche, E., Fischer, A., Kloft, M., Repke, J.-U., Sager, S., y Mitsos, A. (2021). Machine learning in chemical engineering: a perspective. *Chemie Ingenieur Technik*, *93*(12), 2029–2039.

Sheng, F., y Sheu, J. W. (2000). Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast. *Chemical Engineering Science*, *55*(18), 3685–3695.

Silva, M. B., y Inácio, L. M. (2015). Software for simulation of second-generation ethanol production by a simultaneous saccharification and fermentation process. *Chemical Engineering & Technology*.

Soetaert, W., y Vandamme, E. J. (2010). *Industrial biotechnology: sustainable growth and economic success*. John Wiley & Sons.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.

Thebelt, A., Wiebe, J., Kronqvist, J., Tsay, C., y Misener, R. (2022). Maximizing information from chemical engineering data sets: applications to machine learning. *Chemical Engineering Science*, *252*, 117469.

Thibault, J., Van Breusegem, V., y Chéruy, A. (1990). On-line prediction of fermentation variables using neural networks. *Biotechnology and Bioengineering*, *36*(10), 1041–1048.

Trinh, C., Meimaroglou, D., y Hoppe, S. (2021). Machine learning in chemical product engineering: The state of the art and a guide for newcomers. *Processes*, *9*(8), 1456.

Villadsen, J., Nielsen, J., y Lidén, G. (2011). *Bioreaction engineering principles*. Springer Science & Business Media.

Wang, L., Ridgway, D., Gu, T., y Moo-Young, M. (2008). Kinetic modeling of cell growth and product formation in submerged culture of recombinant aspergillus niger. *Chemical Engineering Communications*, *196*(4), 481–490.

Wirsansky, E. (2020). *Hands-on genetic algorithms with python: applying genetic algorithms to solve real-world deep learning and artificial intelligence problems*. Packt Publishing Ltd.

Yüzgeç, U. (2010). Performance comparison of differential evolution techniques on optimization of feeding profile for an industrial scale baker's yeast fermentation process. *ISA transactions*, *49*(1), 167–176.

Zapata M, J. E., Hoyos R, M., y QuinchÍa B, L. A. (2005). Kinetic parameters of growth of saccharomyces cerevisiae affected by a varying magnetic field of low intensity and high frequency. *Vitae*, *12*(1), 39–44.

A. ANEXO A. Metodología del desarrollo de la aplicación web FermApp®

A.1. Recopilación y análisis de requerimientos

La etapa inicial de nuestra metodología del desarrollo de la aplicación se enfoca en la recompilación y análisis de requerimientos. Este paso primordial sienta las bases para las etapas sucesivas y es determinante para el éxito del proyecto, dado que implica un entendimiento detallado y la definición precisa de los requerimientos del proyecto, tanto desde una perspectiva técnica como de usuario. El presente proyecto está orientado hacia el desarrollo de una aplicación web que facilite la recolección y el análisis de datos experimentales derivados de procesos de fermentación a nivel de biorreactor. Los datos a manejar provienen principalmente de dos clases de experimentos: experimentos cinéticos y diseños experimentales.

A.1.1. Almacenamiento de los datos experimentales

Datos de cinéticas experimentales

Los experimentos cinéticos proporcionan datos temporales, recopilados a través de la toma de muestras de un biorreactor en intervalos de tiempo específicos durante un proceso de fermentación. Los datos recolectados en estos experimentos son utilizados para calcular valores como la concentración de biomasa, sustrato y producto. Dichos datos son fundamentales para la estimación de parámetros cinéticos, como la tasa de crecimiento máximo y la constante de afinidad. Estos parámetros son esenciales para la evaluación y comparación de diferentes procesos de fermentación, y contribuyen a la determinación de las condiciones operativas óptimas. Por lo tanto, el desarrollo de un sistema eficaz y automatizado para la estimación de estos parámetros es de vital importancia para nuestro proyecto. La aplicación web debe ser diseñada para recibir y gestionar eficientemente este tipo de datos temporales.

Datos de diseños experimentales

El segundo tipo de experimento, diseños experimentales, se centra en evaluar el impacto de una o más variables de proceso en medidas de salida específicas, las cuales pueden incluir la

productividad, el rendimiento o el tiempo de fermentación. Este tipo de experimentos normalmente se rigen por la metodología de Diseño de Experimentos (DOE). Los datos generados a partir de estos experimentos son analizados estadísticamente para sugerir mejoras en el proceso de fermentación. El propósito podría ser maximizar el rendimiento o minimizar el tiempo de fermentación. La aplicación web debe estar diseñada para aceptar, almacenar y gestionar de manera eficaz este tipo de datos experimentales.

Información del experimento

Además de manejar datos experimentales, la aplicación debe ser capaz de registrar información detallada sobre los experimentos, que incluya el microorganismo y sustrato utilizado, las observaciones, y todas las variables y parámetros considerados. Se debe proveer una interfaz intuitiva para que los usuarios puedan ingresar esta información de manera precisa y eficiente.

Filtrado y ordenamiento de datos experimentales

La aplicación web debe incorporar una funcionalidad que permita a los usuarios filtrar y ordenar los datos experimentales recopilados. Esto facilitará a los usuarios la localización de información específica de manera más eficiente. Por ejemplo, los usuarios deberían poder filtrar datos en base a criterios variados, como el tipo de experimento (cinético o de optimización de proceso), el microorganismo utilizado, o la fecha del experimento. Además, deberían poder ordenar los datos en base a distintos parámetros, como el rendimiento o el tiempo de fermentación. Esta funcionalidad es esencial para ayudar a los usuarios a acceder a sus datos de manera más efectiva, tomar decisiones mejor fundamentadas y, en última instancia, promover la optimización de sus procesos de fermentación. Al proveer herramientas sólidas y fáciles de usar para el filtrado y ordenamiento, la aplicación web puede mejorar significativamente la experiencia del usuario y la utilidad general del sistema.

A.1.2. Visualización de datos

Un componente esencial de nuestra aplicación web es la visualización eficiente de los datos. Esta herramienta es indispensable para comprender y analizar los conjuntos de datos recabados.

Visualización de experimentos cinéticos

En el caso de los datos cinéticos, la aplicación debe ofrecer la capacidad de crear gráficos de

dispersión, los cuales son usualmente utilizados para rastrear cambios temporales en variables como la concentración de biomasa y la concentración de sustrato. Dichos gráficos son esenciales para entender el comportamiento cinético de los procesos de fermentación.

Visualización de diseños experimentales

Para los datos obtenidos de diseños experimentales, la aplicación debe soportar la creación de gráficos tridimensionales que permitan visualizar las relaciones entre diferentes variables. Esta funcionalidad facilita el análisis de las relaciones entre una o más variables independientes y una variable dependiente.

Uso de gráficas interactivas

Para mejorar la experiencia del usuario y hacer que las visualizaciones sean más dinámicas y amigables, nuestra aplicación web utiliza gráficas interactivas mediante la biblioteca `Plotly.js`. Con `Plotly.js`, los usuarios pueden explorar los datos de manera interactiva al hacer zoom, desplazarse o seleccionar áreas específicas en los gráficos. Esto permite un análisis más detallado y una comprensión más profunda de los conjuntos de datos.

A.1.3. Análisis de datos

La aplicación debe contar con un robusto conjunto de herramientas para el análisis de datos.

Análisis de experimentos cinéticos

En el caso de los experimentos cinéticos, el análisis de datos usualmente involucra la determinación del valor de los parámetros cinéticos mediante una regresión no lineal. Este proceso implica la selección de un modelo de biorreactor apropiado y su ajuste a los datos a través de un proceso de optimización que minimiza la discrepancia entre los datos experimentales y las predicciones del modelo. Análisis estadísticos adicionales pueden brindar más información sobre los procesos y permitir la comparación de parámetros cinéticos.

Análisis de diseños experimentales

Para diseños experimentales, la meta es determinar la relación entre las variables independientes y dependientes. Esto se logra generalmente ajustando un modelo lineal a los datos y evaluando su adecuación. Si el modelo es inadecuado, se pueden considerar modelos alter-

nativos. Los análisis estadísticos adicionales pueden arrojar más luz sobre la relación entre las variables y los parámetros del modelo. En muchos casos, también es deseable encontrar condiciones óptimas para maximizar o minimizar la variable dependiente, lo cual requiere un análisis de optimización adicional.

Accesibilidad y gestión de los análisis

Un aspecto importante en el diseño de la aplicación web es asegurar que todos los análisis estén disponibles para su posterior consulta. La aplicación permite el almacenamiento de todos los análisis realizados. Los usuarios pueden acceder a sus análisis previos, ofreciéndoles un almacén organizado de información que puede ser revisada y reevaluada convenientemente en cualquier momento. La aplicación también provee características avanzadas como el filtrado y ordenamiento de análisis.

A.1.4. Registro y gestión de cuentas de usuario

Un requisito fundamental para nuestra aplicación web es la capacidad de los usuarios de registrarse para obtener una cuenta y administrar sus perfiles. El sistema de cuentas de usuario es esencial para la aplicación, dado que no solo brinda una experiencia personalizada para cada usuario, sino que también facilita la gestión segura de datos y el control de acceso.

Registro de usuario

La aplicación debe ofrecer un proceso de registro intuitivo y sencillo. Los futuros usuarios deberían poder registrarse proporcionando la información necesaria, como su nombre, nombre de usuario y contraseña. Tras un registro exitoso, los usuarios deben recibir un correo electrónico de confirmación para verificar su cuenta.

Autenticación de usuario

Una vez registrados, los usuarios deberían poder iniciar sesión en sus cuentas usando su nombre de usuario y contraseña. La aplicación debe gestionar de manera segura la autenticación de los usuarios, garantizando que las credenciales de los usuarios se almacenen y transmitan de forma segura.

Gestión de perfiles de usuario

Cada usuario debe contar con un perfil personal donde pueda visualizar y editar sus datos personales. También deberían tener la posibilidad de cambiar su contraseña y gestionar otras configuraciones de la cuenta desde su perfil.

Roles de usuario y control de acceso

La aplicación debe implementar roles de usuario y un sistema de control de acceso para garantizar que sólo los usuarios autorizados puedan acceder a ciertas funciones. Por ejemplo, los usuarios normales deberían poder ver y gestionar sus propios datos experimentales, mientras que los usuarios con rol de administrador deberían tener privilegios adicionales.

Gestión de usuarios con rol de administrador

Para los usuarios con rol de administrador, deben proporcionarse permisos adicionales. Los administradores deben poder crear nuevas cuentas de usuario, editar detalles de usuarios existentes, asignar roles y desactivar o eliminar cuentas de usuario. Los usuarios con rol de administrador deber ser capaces de manejar y/o eliminar cuentas de otros usuarios, así como poder acceder a los datos experimentales de otros usuarios, sin importante quien es el usuario que registró los datos experimentales.

A.2. Planificación

La etapa de planificación es el segundo paso en nuestra metodología y es fundamental para dar forma a la trayectoria del proyecto. Implica decidir sobre la pila de tecnología, la arquitectura del sistema y la estrategia de implementación para la aplicación web. La fase de planificación tiene como objetivo garantizar que el proyecto no solo sea viable sino también factible, sólido y fácil de usar.

A.2.1. Selección de pila de tecnología

Una parte crucial de la fase de planificación es la selección de la pila de tecnología. La elección de tecnologías influye directamente en las capacidades, el rendimiento y la escalabilidad del producto final. Para este proyecto, las tecnologías seleccionadas se eligen específicamente para satisfacer los requisitos y la funcionalidad esperada de la aplicación web.

Backend:

Django, un marco web Python de alto nivel, se elige por su facilidad de uso y sus potentes características que facilitan el desarrollo rápido de aplicaciones web. Django REST Framework (DRF) complementa a Django al proporcionar herramientas flexibles y sólidas para crear API. Dadas las sólidas capacidades de análisis de datos de Python y el rico ecosistema de bibliotecas científicas como NumPy y pandas, esta elección se alinea con nuestra necesidad de manejar análisis de datos complejos.

Frontend:

React.js, una potente biblioteca de JavaScript para crear interfaces de usuario, se selecciona para el desarrollo de la interfaz. La arquitectura basada en componentes de React promueve la reutilización y la facilidad de administración, especialmente beneficiosa para aplicaciones web dinámicas como la nuestra. También aprovecharemos Bootstrap, un marco CSS popular, para facilitar la implementación rápida y eficiente de estilos y diseños receptivos. Además, Plotly.js se utilizará junto con React para generar gráficos dinámicos, interactivos y de calidad profesional.

Base de datos:

MySQL, un sistema de administración de base de datos relacional de código abierto ampliamente utilizado, se utilizará para manejar nuestras necesidades de almacenamiento de datos. El rendimiento, la solidez y la compatibilidad de MySQL con varios servicios de hospedaje lo convierten en una excelente opción. El diseño de nuestra base de datos implicará una cuidadosa consideración de los tipos de datos y las relaciones para garantizar un almacenamiento y una recuperación de datos eficientes.

Alojamiento:

Elegimos a *Hostinger* como nuestro proveedor de alojamiento web debido a sus precios asequibles y ofertas de servicios integrales. *Hostinger* proporciona funcionalidades para administrar el servidor, la aplicación web y las bases de datos en un solo lugar. El soporte para MySQL y la disponibilidad de secuencias de comandos del lado del servidor en Python lo convierten en una opción compatible para nuestra pila de tecnología elegida.

A.2.2. Diseño de la arquitectura del sistema

La fase de diseño de la arquitectura del sistema es fundamental para la etapa de planificación del proyecto. Implica definir la estructura de alto nivel de la aplicación, detallar cómo los diferentes componentes del sistema interactuarán entre sí y determinar cómo fluirán los datos a través de la aplicación. Para nuestro proyecto, hemos elegido una combinación de tecnologías y componentes para crear un sistema robusto, eficiente y escalable.

La arquitectura de nuestro sistema está diseñada en torno a un modelo cliente-servidor en el que el *frontend* (cliente) y el *backend* (servidor) son entidades separadas que se comunican entre sí. Esta separación de preocupaciones permite un desarrollo más modular, una escalabilidad más fácil y una capacidad de mantenimiento mejorada.

Backend:

El backend de nuestra aplicación se crea con Django, un marco web Python de alto nivel, y Django Rest Framework (DRF), un potente conjunto de herramientas para crear `API` web. Se elige DRF por su capacidad para crear `API` de manera rápida y sencilla que pueden manejar requisitos de manejo de datos complejos, y su compatibilidad con Django lo convierte en una opción natural para el backend. El *backend* es responsable de administrar la lógica de la aplicación, manejar el procesamiento de datos y comunicarse con la base de datos. Expone una `API` *REST* con la que el *frontend* puede interactuar. La `API` *REST* utiliza métodos HTTP estándar (*GET*, *POST*, *PUT*, *DELETE*) para facilitar la comunicación entre el cliente y el servidor.

Frontend:

En el lado del cliente, usamos *React.js*, una biblioteca popular de *JavaScript* para crear interfaces de usuario. La arquitectura basada en componentes de *React.js* permite una base de código más organizada, lo que facilita el mantenimiento y escalamiento de la aplicación. Para la gestión de estado en nuestra aplicación `React` usamos `Redux`. `Redux` mantiene el estado de una aplicación completa en un único árbol de estado inmutable, lo que puede ayudar a garantizar la coherencia, la previsibilidad y la facilidad de depuración. Cada componente puede acceder al estado según sea necesario, sin tener que pasar *props* a los componentes secundarios o usar funciones de *callback* para enviar datos de regreso a un componente padre.

Para realizar solicitudes HTTP a nuestra `API` *REST* de `Django`, usamos `Axios`, una librería

de JavaScript para hacer peticiones HTTP desde navegadores web. Axios proporciona una API simple y limpia para hacer solicitudes HTTP desde el *frontend* hasta el *backend*, maneja las respuestas y detecta cualquier error que pueda ocurrir durante la solicitud.

Base de datos:

Todos los datos de la aplicación se almacenan en una base de datos MySQL. MySQL es un sistema de gestión de bases de datos relacionales robusto y eficiente, que proporciona todas las funciones necesarias para nuestra aplicación. El *Object Relational Mapper* (ORM) de Django nos permite interactuar con la base de datos, obtener datos o crear nuevos datos, escribiendo código Python en lugar de SQL.

Alojamiento:

Hostinger es elegido como nuestro proveedor de alojamiento web. Proporciona una plataforma para administrar el servidor, la aplicación web y las bases de datos en un solo lugar. El soporte para MySQL y la disponibilidad de secuencias de comandos del lado del servidor en Python lo convierten en una opción compatible para nuestra pila de tecnología elegida.

El diseño de la arquitectura del sistema es fundamental para decidir la comunicación entre el frontend y el backend, cómo fluirán los datos a través de la aplicación y cómo la aplicación interactuará con el usuario y cualquier sistema externo. Con una arquitectura de sistema bien diseñada, estamos asegurando una base sólida para el desarrollo y la implementación del proyecto.

A.2.3. Plazos

Aquí se presenta un cronograma estimado del proyecto basado en la comprensión del proyecto. Es importante tener en cuenta que se trata de estimaciones y que el tiempo real puede variar según varios factores, como la disponibilidad de recursos, la complejidad de las tareas, etc.

Recolección y Análisis de Requerimientos: La duración estimada es de 4 semanas. Esta fase implica comprender los requisitos del proyecto, definirlos y planificar cómo cumplir estos requisitos en las etapas posteriores del proyecto. Planificación: La duración estimada es de 2 semanas. En esta fase, se toman decisiones sobre la pila de tecnología, la arquitectura del sistema

y la estrategia de implementación.

Diseño: La duración estimada es de 4 semanas. Esta fase se centra en el diseño de la interfaz de usuario (UI) y la experiencia del usuario (UX), y tiene como objetivo crear una interfaz intuitiva, fácil de usar y visualmente atractiva.

Desarrollo: La duración estimada es de 12 semanas. Esta es la fase donde ocurre la creación real de la aplicación web. Implica configurar el entorno de desarrollo, estructurar la base de datos, desarrollar el *backend* y el *frontend*, integrar el *frontend* y el *backend* y realizar pruebas.

Prueba: La duración estimada es de 4 semanas. La aplicación se somete a una serie de pruebas para garantizar que funcione según lo previsto. Esto incluye pruebas unitarias, pruebas de integración, pruebas funcionales, pruebas de usabilidad, pruebas de rendimiento, pruebas de seguridad y pruebas de compatibilidad.

Despliegue: La duración estimada es de 2 semanas. En esta etapa, la aplicación web se pone a disposición de los usuarios. Esto incluye elegir un plan de alojamiento, establecer y configurar la cuenta de alojamiento, configurar la base de datos, cargar el sitio web, instalar dependencias, configurar un entorno virtual, configurar `Django` para que se ejecute en modo de producción, iniciar la aplicación, configurar la implementación continua y seguimiento de la aplicación.

Mantenimiento y actualizaciones: este es un proceso continuo que comienza después de la implementación. Será necesario mantener la aplicación web. Esto incluye brindar soporte al usuario, corregir errores, implementar actualizaciones y agregar nuevas funciones basadas en los comentarios de los usuarios.

Esto lleva el tiempo total estimado a alrededor de 28 semanas, excluyendo el mantenimiento y las actualizaciones en curso. Es importante tener en cuenta que estas son estimaciones aproximadas y que el cronograma real puede variar dependiendo de una variedad de factores. Se deben realizar controles y ajustes regulares del proyecto según sea necesario.

A.2.4. Funciones y responsabilidades del equipo

Las funciones y responsabilidades del equipo se detallarán claramente en la fase de planificación. Esto ayuda a establecer un camino claro para una colaboración eficiente y asegura que todos los aspectos del proyecto estén adecuadamente cubiertos.

A.2.5. Consideraciones de seguridad

En términos de seguridad, nos aseguraremos de que la aplicación se desarrolle siguiendo las mejores prácticas. Esto implica autenticación de usuario segura, cifrado de datos y medidas para evitar vulnerabilidades web comunes.

A.2.6. Planificación de escalabilidad

Por último, la etapa de planificación también tendrá en cuenta la escalabilidad futura. A medida que crece la aplicación, puede haber una necesidad de manejar más datos y usuarios, por lo que el diseño y las tecnologías elegidos deberían poder adaptarse a este crecimiento.

La fase de planificación sienta las bases para el desarrollo, las pruebas y la implementación del proyecto, lo que garantiza que todos los aspectos estén bien pensados y que el proyecto esté listo para el éxito.

A.3. Diseño del *frontend*

La fase de diseño de nuestra aplicación web se centrará principalmente en el diseño de la interfaz de usuario (*UI*) y la experiencia del usuario (*UX*), considerando tanto los aspectos estéticos como funcionales. Nuestro objetivo es crear una interfaz intuitiva, fácil de usar y visualmente atractiva, que permita a los usuarios navegar y realizar tareas fácilmente.

El diseño de la aplicación web se compone de cuatro partes principales: Encabezado con Navegación: El encabezado se colocará en la parte superior de cada página. Contendrá el logo de la aplicación a la izquierda, reforzando la identidad de la marca, y una barra de navegación a la derecha. La barra de navegación incluirá enlaces a *Home*, *Login*, *Account*, *Parameter Estimation*, *Process optimization*, *Visualization*, *Experiments*, *Analyses* y *Users*. Esto facilitará la navegación a las secciones clave de la aplicación. Para los usuarios que no han iniciado sesión, solo podrán acceder a la sección de Login, y necesitarán iniciar sesión para acceder al resto de la aplicación.

Panel de la barra lateral: un panel de la barra lateral proporcionará controles y opciones para configurar el análisis estadístico, la visualización, el llenado de formularios y el filtrado de tablas. Se diseñará con controles fáciles de usar, como controles deslizantes, casillas de verificación y

menús desplegables, ofreciendo a los usuarios un buen grado de control sobre los datos con los que están trabajando. El usuario podrá ocultar el panel de la barra lateral si lo necesita. Panel principal: el panel principal mostrará los resultados según la entrada del usuario desde el panel de la barra lateral. Tendrá un diseño flexible para acomodar varios tipos de representación de datos, incluidos gráficos, resultados de análisis estadísticos y tablas con datos experimentales. El diseño se ajustará dinámicamente en función de los datos que se muestren.

Modales: los modales se usarán para mostrar datos adicionales o capturar la entrada del usuario sin interrumpir la estructura general de la aplicación. Se utilizarán para tareas como ingresar nuevos datos de experimentos, confirmar acciones o mostrar información adicional relacionada con los datos presentados en el panel principal.

La aplicación web contará con las siguientes secciones:

Home: la página de inicio de la aplicación, aquí se mostrará información específica del usuario, así como la opción de cerrar sesión. Login: una página de inicio de sesión segura para que los usuarios accedan a sus cuentas antes de tener acceso al resto de la aplicación.

Account: aquí, los usuarios pueden administrar la configuración de su cuenta y ver sus perfiles.

Parameter Estimation: los usuarios pueden ingresar datos de experimentos cinéticos y llevar a cabo la estimación de parámetros cinéticos.

Process optimization: en esta sección el usuario puede introducir datos de diseños experimentales y llevar a cabo análisis estadísticos y de optimización. Visualization: los usuarios pueden introducir datos experimentales y generar gráficos interactivos para analizar sus datos de forma eficiente.

Experiments: una base de datos de todos los experimentos realizados por el usuario, con capacidades de búsqueda, ordenación y filtrado.

Analyses: Una base de datos de todos los análisis realizados por el usuario, con capacidades de búsqueda, ordenación y filtrado.

Users: una sección de administración para que los administradores administren las cuentas y los permisos de los usuarios.

En términos de estética, la aplicación se adherirá a un diseño limpio y moderno, utilizando

un esquema de colores y una tipografía armoniosos para garantizar una experiencia de usuario agradable. Se implementará un diseño responsivo para garantizar la usabilidad en varios dispositivos y tamaños de pantalla.

Elementos interactivos como botones y enlaces proporcionarán información visual para ayudar a los usuarios a comprender la respuesta de la aplicación a sus acciones. Los formularios y las tablas de datos se diseñarán para que sean legibles y fáciles de usar, con la validación adecuada y la retroalimentación de errores.

La fase de diseño implicará la creación de *wireframes* y *mockups* para cada página, seguidas de pruebas de usuario e iteraciones basadas en los comentarios. El objetivo es garantizar que el diseño final esté centrado en el usuario, proporcionando una experiencia eficiente y agradable para todos los usuarios.

A.4. Desarrollo de la aplicación web

La fase de desarrollo marca la etapa de implementación de nuestra metodología de proyecto, transformando los requisitos recopilados meticulosamente, la planificación integral y el diseño cuidadoso en una aplicación web completamente funcional. Esta fase implica varias capas de progresión, desde la configuración del entorno de desarrollo hasta la integración de elementos del *backend*, *frontend* y base de datos en un sistema operativo cohesivo. Las siguientes subsecciones detallan cada etapa en el proceso de desarrollo.

A.4.1. Configuración del entorno

La fase de desarrollo comienza con el establecimiento del entorno de desarrollo, un paso preparatorio crucial que garantiza que todas las herramientas y tecnologías necesarias estén disponibles. Para este proyecto, la configuración de nuestro entorno incluye la instalación de `Django` y `Django Rest Framework` para el desarrollo de *backend*, `React.js` para el desarrollo de *frontend* y `Bootstrap` y `Plotly` para el estilo de *frontend* y la visualización de datos. Además, `MySQL`, nuestro sistema de administración de base de datos elegido, debe instalarse y configurarse.

A.4.2. Estructuración de la base de datos

Después de la configuración del entorno, nos enfocamos en estructurar la base de datos y establecer todas las tablas necesarias. Este paso implica el diseño y la creación de esquemas de bases de datos para almacenar de manera eficiente la información del usuario, los datos experimentales de las cinéticas de fermentación y diseños experimentales, y cualquier dato adicional pertinente a nuestra aplicación.

A.4.3. Desarrollo del *backend*

El desarrollo de *backend* constituye el núcleo de nuestra lógica de aplicación, que abarca el procesamiento de datos, la autenticación de usuarios y otras operaciones del lado del servidor. `Django`, junto con `Django Rest Framework`, forma la columna vertebral de nuestra lógica del lado del servidor. Durante esta etapa, construimos las `API REST` para permitir funcionalidades como la recopilación, el análisis y la visualización de datos, lo que garantiza que estos servicios estén disponibles para el *frontend*.

A.4.4. Desarrollo del *frontend*

El desarrollo *frontend* implica transformar los *wireframes* y *mockups* creados durante la fase de diseño en una interfaz de usuario dinámica, interactiva y responsiva. Usando `React.js`, creamos elementos visuales dinámicos, y con `Bootstrap`, diseñamos estos elementos para una interfaz de usuario atractiva, intuitiva y responsiva. En esta etapa se implementan funciones para la visualización de datos (incluidos diagramas de dispersión y diagramas 3D), análisis estadístico y filtrado de tablas. Además, creamos varias secciones como *Home*, *Login*, *Account*, *Parameter Estimation*, *Process optimization*, *Visualization*, *Experiments*, *Analyses* y *Users* para facilitar la navegación y el uso.

A.4.5. Integración

La fase de integración reúne los componentes del *frontend*, *backend* base de datos de nuestra aplicación. Esta etapa consiste en conectar la interfaz a nuestras `API REST` de `Django` con el

frontend, lo que permite el intercambio de datos entre el cliente y el servidor. El *frontend* inicia solicitudes al *backend*, el cual procesa estas solicitudes y responde en consecuencia, lo que permite actualizaciones dinámicas en la interfaz de usuario.

A.4.6. Pruebas

Tras la integración exitosa del *frontend* y el *backend*, se realizan una serie de pruebas para verificar la funcionalidad, el rendimiento y la seguridad del sistema. La fase de prueba es vital para garantizar que la aplicación se comporte según lo previsto, ofreciendo una experiencia de usuario perfecta. Abarca pruebas de funcionalidad, pruebas de interfaz de usuario, pruebas de rendimiento, pruebas de seguridad, entre otras, según los requisitos del proyecto.

A.5. Pruebas

La fase de prueba representa un paso esencial en nuestra metodología de proyecto, asegurando que la aplicación web funcione según lo previsto en diversas condiciones y escenarios. Esta fase implica una serie de pruebas diseñadas para validar todos los aspectos de la aplicación, desde los componentes individuales hasta el sistema en su conjunto. Las siguientes subsecciones aclaran cada tipo de prueba realizada durante esta fase.

A.5.1. Pruebas unitarias

Las pruebas unitarias implican evaluar componentes individuales o unidades de la aplicación de forma aislada para verificar su correcto funcionamiento. En el *backend*, esto podría significar probar funciones y clases individuales, asegurándose de que devuelvan los resultados esperados para las entradas dadas. En la interfaz, esto podría implicar probar componentes individuales de `React.js`, confirmar que se procesan correctamente y responden a las interacciones del usuario como se esperaba.

A.5.2. Pruebas de integración

Después de garantizar el correcto funcionamiento de los componentes individuales, se realizan pruebas de integración para verificar que estos componentes funcionan correctamente cuando se interconectan. Esto incluye probar las interacciones entre los componentes de *frontend* y *backend*, como garantizar que las llamadas `API` desde el *frontend* activen las funciones apropiadas en el *backend* y devuelvan las respuestas esperadas.

A.5.3. Pruebas funcionales

Las pruebas funcionales tienen como objetivo validar que cada función de la aplicación web se comporte de acuerdo con los requisitos especificados. Implica llevar a cabo tareas que realizarían los usuarios típicos, como iniciar sesión, ingresar datos, realizar análisis y ver resultados, para garantizar que estas funcionalidades funcionen correctamente.

A.5.4. Pruebas de usabilidad

Las pruebas de usabilidad aseguran que la aplicación ofrezca una experiencia fácil de usar y fácil de navegar. Esto incluye verificar la intuición de la navegación de la aplicación, garantizar que los botones y enlaces estén colocados correctamente y funcionen, y confirmar que las visualizaciones son fáciles de entender, interactivas y brindan información significativa.

A.5.5. Pruebas de rendimiento

Las pruebas de rendimiento se realizan para evaluar el rendimiento de la aplicación en cargas de trabajo y condiciones específicas. Esto podría implicar medir la velocidad del procesamiento de datos en el *backend*, el tiempo de carga de las páginas en el *frontend* y la capacidad de respuesta general de la aplicación. La aplicación debe mantener un rendimiento óptimo incluso con mucho tráfico o volumen de datos.

A.5.6. Pruebas de seguridad

Las pruebas de seguridad son fundamentales para garantizar que la aplicación proteja los datos del usuario y sea inmune a amenazas potenciales. Esto implica verificar que la transmisión de datos a través de Internet sea segura, que las contraseñas se almacenen de forma segura y que la aplicación esté protegida contra vulnerabilidades web comunes.

A.5.7. Pruebas de compatibilidad

Las pruebas de compatibilidad garantizan que la aplicación funcione sin problemas en diferentes entornos, incluidos varios navegadores web (como Firefox, Chrome, Safari), dispositivos (como escritorio, tableta, móvil) y tamaños de pantalla. Garantizar la compatibilidad mejora la accesibilidad de la aplicación y la experiencia del usuario. Al identificar cualquier error o problema durante la fase de prueba, se registran y asignan a los desarrolladores para su resolución. Después de la resolución, las pruebas se vuelven a ejecutar para confirmar que el problema se solucionó por completo. Este ciclo de prueba, identificación de errores, reparación de errores y nuevas pruebas continúa hasta que la aplicación cumple con los estándares de calidad establecidos. La fase de prueba es vital para entregar una aplicación web confiable, eficiente y segura.

A.6. Despliegue a producción

La implementación representa la fase en la que la aplicación web desarrollada pasa de un entorno de desarrollo a un entorno de producción en vivo, haciéndola accesible para los usuarios finales. Esta fase implica instalar y configurar el entorno de hospedaje, cargar la aplicación y asegurarse de que se ejecute como se espera en el entorno de producción. Para este proyecto, *Hostinger* es el proveedor de alojamiento elegido. Las secciones siguientes detallan cada paso del proceso de implementación.

A.6.1. Selección del Plan de Alojamiento

Hostinger ofrece una variedad de planes de alojamiento para satisfacer las diferentes necesidades de los proyectos. La selección de un plan adecuado depende de los requisitos específicos de la aplicación web en términos de almacenamiento, tráfico, velocidad y escalabilidad. Como nuestro marco de *backend*, `Django`, está basado en `Python`, es fundamental elegir un plan de alojamiento que admita `Python`.

A.6.2. Configuración de la cuenta de alojamiento

Después de seleccionar y comprar el plan de hospedaje adecuado, el siguiente paso es establecer y configurar la cuenta de hospedaje. Este paso puede implicar la creación de un dominio para la aplicación si aún no existe uno.

A.6.3. Configuración de la base de datos

Hostinger admite bases de datos `MySQL`, que se alinea con la elección de base de datos para nuestra aplicación web. Es necesario crear una nueva base de datos y vincularla a la aplicación `Django`. Este proceso requiere credenciales seguras, que deben almacenarse de forma segura para su uso futuro.

A.6.4. Carga del sitio web

Luego, la aplicación se carga en el servidor de *Hostinger*. Este proceso se puede realizar a través de *FTP* (Protocolo de transferencia de archivos). *Hostinger* facilita la conexión al servidor a través de FTP para la carga de aplicaciones. Tanto el *backend* de `Django` como el *frontend* de `React.js` deben cargarse durante este paso.

A.6.5. Instalación de dependencias

Publique la carga del código en el servidor, se deben instalar todas las dependencias necesarias. Esto incluye las versiones adecuadas de `Python`, `Django`, `Django Rest Framework` y cualquier otra biblioteca de `Python` de la que dependa la aplicación.

A.6.6. Configuración del entorno virtual

Configurar un entorno virtual es una práctica recomendada en el desarrollo de `Python`. Ayuda a aislar la aplicación y sus dependencias de otras aplicaciones de `Python` en el mismo servidor, evitando posibles conflictos. Esta configuración generalmente se puede lograr usando el módulo `venv` de `Python`.

A.6.7. Configuración del modo de producción para Django

`Django` debe configurarse para ejecutarse en un entorno de producción. Esto implica configurar la variable `DEBUG` en `False`, configurar la variable `ALLOWED_HOSTS` para incluir el dominio de la aplicación y configurar un servidor web de producción como `Gunicorn` o `uWSGI`. Además, los archivos estáticos deben recopilarse mediante el comando `collectstatic` de `Django`, y la configuración para el manejo de archivos estáticos y multimedia debe configurarse correctamente.

A.6.8. Lanzamiento de la aplicación

Con todos los pasos anteriores completados, se puede iniciar la aplicación web. Por lo general, esto se hace iniciando el servidor `Django` y luego accediendo al dominio de la aplicación a través de un navegador web.

A.6.9. Configuración de implementación continua

La configuración de la implementación continua puede optimizar las futuras actualizaciones de la aplicación. Con esta configuración, cada actualización de código enviada al repositorio se puede implementar automáticamente en la aplicación en vivo, lo que garantiza que la aplicación se mantenga actualizada con los cambios más recientes.

A.6.10. Supervisión de aplicaciones

Tras la implementación, el monitoreo regular de la aplicación es crucial para garantizar su buen funcionamiento, identificar y corregir cualquier problema y comprender los patrones de uso. Se pueden usar varias herramientas para monitorear la aplicación, lo que proporciona

información que puede ayudar a mejorar la aplicación con el tiempo. La fase de implementación concluye con una aplicación web operativa y en vivo que está lista para los usuarios finales. Esta fase es vital para la transición del proyecto de una etapa de desarrollo a un producto listo para el usuario.

B. ANEXO B. Constancias y Participaciones

CERTIFICADO

Registro Público del Derecho de Autor

Para los efectos de los artículos 13, 162, 163 fracción I, 164 fracción I, y demás relativos de la Ley Federal del Derecho de Autor, se hace constar que la **OBRA** cuyas especificaciones aparecen a continuación, ha quedado inscrita en el Registro Público del Derecho de Autor, con los siguientes datos:

AUTORES: CHÁVEZ PARGA MA. DEL CARMÉN
GONZÁLEZ HERNÁNDEZ JUAN CARLOS
GUTIÉRREZ GARCÍA JUAN MANUEL

TÍTULO: FERMAPP

RAMA: PROGRAMAS DE COMPUTACION (APP POR ANALOGÍA)

TITULARES: CHÁVEZ PARGA MA. DEL CARMÉN
GONZÁLEZ HERNÁNDEZ JUAN CARLOS

Con fundamento en el artículo 30 de la Ley Federal del Derecho de Autor, el titular de los derechos patrimoniales puede, libremente, conforme a lo establecido por esta Ley, transferir sus derechos patrimoniales u otorgar licencias de uso exclusivas o no exclusivas. Toda transmisión de derechos patrimoniales de autor será onerosa y temporal. En ausencia de acuerdo sobre el monto de la remuneración o del procedimiento para fijarla, así como sobre los términos para su pago, la determinarán los tribunales competentes. Los actos, convenios y contratos por los cuales se transmitan derechos patrimoniales y las licencias de uso deberán celebrarse, invariablemente, por escrito, de lo contrario serán nulos de pleno derecho.

Con fundamento en lo establecido por el artículo 168 de la Ley Federal del Derecho de Autor, las inscripciones en el registro establecen la presunción de ser ciertos los hechos y actos que en ellas consten, salvo prueba en contrario. Toda inscripción deja a salvo los derechos de terceros. Si surge controversia, los efectos de la inscripción quedarán suspendidos en tanto se pronuncie resolución firme por autoridad competente.

El presente certificado se expide con fundamento en el Decreto por el que se reforman, adicionan y derogan diversas disposiciones de la Ley Orgánica de la Administración Pública Federal, así como de otras leyes para crear la Secretaría de Cultura, publicado el 17 de diciembre de 2015 en el Diario Oficial de la Federación; artículos 26 y 41 Bis, fracción XVIII de la Ley Orgánica de la Administración Pública Federal; artículos 2, 208, 209 fracción III de la Ley Federal del Derecho de Autor; artículo 69-C de la Ley Federal de Procedimiento Administrativo, de aplicación supletoria de acuerdo con lo establecido por la Ley Federal del Derecho de Autor en su artículo 10; artículo 84 de la Ley General de Mejora Regulatoria; artículos 2, apartado B, fracción IV, 26 y 27 del Reglamento Interior de la Secretaría de Cultura; artículos 103 fracción IV y 104 del Reglamento de la Ley Federal del Derecho de Autor; artículos 1, 3 fracción I, 4, 8 fracción I, 9, 16 y 17 del Reglamento Interior del Instituto Nacional del Derecho de Autor; ACUERDO por el que se establecen los Lineamientos para el uso de la Firma Electrónica Avanzada en los actos y actuaciones de los servidores públicos del Instituto Nacional del Derecho de Autor, publicado en el Diario Oficial de la Federación el 19 de mayo del año dos mil veintiuno; y Acuerdo por el que se establecen las reglas para la presentación, substanciación y resolución de las solicitudes de registro de obras, fonogramas, videogramas y edición de libros en línea ante el Instituto Nacional del Derecho de Autor, publicado el 8 de diciembre de 2021 en el Diario Oficial de la Federación.

V COLOQUIO INTERNACIONAL EN INGENIERÍA QUÍMICA

"La Ingeniería Química en la Sociedad actual"

Universidad Michoacana de San Nicolás de Hidalgo
Facultad de Ingeniería Química

Morelia, Michoacán, 18 de Julio de 2023.

Estimado(a) autor(a) de correspondencia

Agradecemos ampliamente su interés por participar en el **V Coloquio Internacional en Ingeniería Química "La Ingeniería Química en la Sociedad Actual".** Por este conducto nos complace informarle que su trabajo:

SOTWARE PARA LA SIMULACIÓN Y OPTIMIZACIÓN DE PROCESOS DE FERMENTACIÓN MEDIANTE PYTHON

ID: 23 cuyos autores son:

Juan Manuel Gutiérrez-García, Ma. del Carmen Chávez Parga y Juan Carlos González-Hernández

ha sido **Aceptado** para su presentación en la modalidad de **Póster presencial**.

En fechas posteriores podrá consultar el programa completo en nuestra página: https://posgrado.fiq.umich.mx/index.php/coloquio-internacional, para conocer el día y hora precisa de su presentación.

Recuerde que tiene hasta el **09 de agosto de 2023** para enviar su manuscrito en extenso en el formato conforme a la plantilla anexa, al correo: coloquiointernacionaldepfiq@gmail.com

A nombre de la División de Estudios de Posgrado de la Facultad de Ingeniería Química de la Universidad Michoacana de San Nicolás de Hidalgo le agradecemos su valiosa participación y esperamos tener la oportunidad de saludarlo durante el evento.

A t e n t a m e n t e

COMITÉ TÉCNICO

V Coloquio Internacional en Ingeniería Química
"La Ingeniería Química en la Sociedad Actual"

Edificio V1 Ciudad Universitaria
Col. Felicitas del Río, Morelia,
Michoacán México. CP 58060.
(443)327584