UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN

FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

Simulador dinámico para espesadores de relave

Christopher Alejandro Carrasco López

Informe de Habilitación Profesional para optar al título de:
Ingeniero Civil Eléctrico

Profesor Patrocinante:
Dr. Samuel Vergara

Profesores Guía:
Dr. Ricardo Lizana.
Dr. Eduardo Espinosa.

Concepción, Julio 2024

UNIVERSIDAD CATÓLICA DE LA SANTISIMA CONCEPCION
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica

Profesor Patrocinante:
Dr. Samuel Vergara.

Simulador dinámico para espesadores de relave

Christopher Alejandro Carrasco López

Informe de Habilitación Profesional
para optar al Título de

Ingeniero Civil Eléctrico

Concepción, Julio 2024

Resumen

Los espesadores de relave son tanques cilíndricos utilizados ampliamente en el área industrial. Su diseño permite llevar a cabo el proceso de separación solido-liquido mediante el fenómeno físico de sedimentación. Los estándares de calidad a los que se someten los espesadores de relave implican una correcta instrumentación y un óptimo control en su interfaz.

La separación solido-liquido producido al interior de los espesadores de relave se caracteriza por su naturaleza lenta y prolongada. Por esta razón, resulta necesario utilizar herramientas simuladas que posibiliten condensar estos procesos en intervalos más acotados. Las herramientas de simulación desempeñan un papel fundamental al permitirnos comprender y analizar de manera clara la dinámica de un proceso, lo que a su vez facilita la optimización, comprensión y el control de los sistemas en tiempos más reducidos.

Se decide realizar una investigación y aplicación con el propósito de presentar una estrategia para la creación de un simulador para espesadores de relave. El simulador permite representar las características dinámicas del espesador posibilitando la manipulación de las variables de entrada y salida. El proceso al interior de los espesadores de relave se define en función de una secuencia de estudios profesionales que abarcan desde los primeros momentos de funcionamiento hasta sus características dinámicas

En respuesta a esta complejidad, se ha optado por desarrollar un simulador dinámico que aproveche la sinergia entre un lenguaje compilado asociado a Fortran y un lenguaje de programación interpretativo como lo es Python. Este tipo de estrategia permite, condensar y acelerar la representación de estos procesos, permitiendo mejorar los tiempos de respuesta hasta el 90%, al comparar tiempos de ejecución en Python y con su modulo F2py, permitiendo un análisis que refleje la dinámica del espesador en tiempos más acotados y que permita contribuir a entrenamiento de operadores.

A los alumnos del pasado, presente y futuro de la carrera de Ingeniería Civil Eléctrica de la U.C.S.C.

Agradecimientos

En primera instancia agradecer el apoyo incondicional de mis padres, por haberme guiado en este largo camino de mi etapa como estudiante. Agradecer a mi hermano por todos los momentos que me apoyo cuando me encontraba sin ánimos y no dejarme solo.

Agradecer al Profesor Samuel, por brindarme esta oportunidad de trabajar con él y de creer en mis capacidades, por su excelente vocación y por ser un excelente guía ya que siempre ha estado presente para brindarme su ayuda ante cualquier inconveniente y siempre aconsejarme en momentos difíciles. Finalmente agradecer a mis amigos Edgard por ser un excelente amigo y por haberme apoyado siempre a lo largo de la carrera, agradecer a Fidel por todos los momentos agradables vividos y el apoyo incondicional.

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Nomenclatura

Escalares

g : Aceleración de gravedad [m/s^2].
x_r : Distancia entre flujo de alimentación y descarga del espesador.
x_l : Distancia entre el flujo de alimentación y el rebalse del espesador.
d : Diámetro del espesador.
ρ_s : Densidad relativa del sólido.
ρ_l : Densidad relativa del líquido.
C : Parámetro de compactación del sistema.
u_{max} : Velocidad máxima del sistema.
v_{st} : Velocidad superficial del sistema.
α : Fuerza de arrastre de una partícula.
β : Constante de la ley de Stokes.
u_{crit} : Velocidad critica de sedimentación.
Q_F : Caudal de Alimentación del flujo [m^3/hr].
Q_R : Caudal de Descarga del flujo [m^3/hr].
Q_L : Caudal de Rebalse del flujo [m^3/hr].
q_F : Caudal de Alimentación del flujo [m/s].
q_R : Caudal de Descarga del flujo [m/s].
q_L : Caudal de Rebalse del flujo [m/s].
ϕ_F : Concentración volumétrica de alimentación.
ϕ_R : Concentración volumétrica de descarga.
$f(\varphi, t)$: Función densidad de flujo sólidos.
$f_{bk}(\varphi)$: Función densidad de flujo solidos Kynch.
ϕ_R : Concentración volumétrica de descarga.

Abreviaciones

Mayúsculas

$L.A$	: Lazo abierto.
$P.Y$	: Lenguaje de programación Python.
$F.90$	: Lenguaje de programación Fortran.
$F2py$	: Modulo Numpy.
CFL	: Restricción para métodos numéricos.
EDP	: Ecuaciones diferenciales parciales.

Capítulo 1. Introducción

1.1. Introducción General

El sector minero en Chile se encuentra ubicado principalmente en zonas desérticas, caracterizada por la escasez hídrica [8]. En este contexto, se vuelve primordial aumentar esfuerzos con métodos y propuestas que fomenten la reutilización recursos naturales, especialmente la recirculación del agua utilizada en los múltiples procesos industriales.

En el proceso minero, los espesadores de relave son máquinas que desempeñan su rol en el tratamiento de los minerales, formando parte de las últimas etapas del ciclo minero [8]. Estas máquinas se encargan de recibir un flujo de pulpa proveniente de etapas previas, con la finalidad de llevar a cabo el proceso de separación entre solidos-líquidos en su interior. Esto se traduce en la generación de un flujo inferior compuesto por concentrado sólido y un flujo superior de agua clarificada libre de partículas [5][7]. Bajo esta premisa, estas máquinas desempeñan un papel en la recuperación de agua, particularmente por desbordamiento en las zonas periféricas del espesador, esta recuperación, permite reutilizar el agua recuperada, distribuyendo su capacitación hacia las diversas etapas de las operaciones mineras [8].

Los estándares de calidad y seguridad en el funcionamiento de los espesadores de relave deben ser rigurosos, por lo que su regulación normativa debe ser estricta [10]. Los espesadores de relave son maquinas que funcionan de forma continuada, por lo que cualquier falla producida, ya sea por el mal control, instrumentación y componentes asociados al diseño del espesador, podría tener consecuencias severas, tanto en el proceso como el medio ambiente. Estas fallas pueden afectar en la continuidad del proceso y por consecuencia, impedir que el material sólido alcance niveles deseados. Esto no solo afecta al medio ambiente y la población circundante, sino que también puede tener impactos negativos en el proceso productivo en su conjunto.

Los espesadores de relave son máquinas que funcionan de forma continua, esto implica que sean consideradas maquinas críticas dentro del proceso industrial. Cualquier condición de falla producida en uno de estos espesadores, puede ocasionar la paralización de toda la operación del proceso productivo.

El propósito del presente trabajo tiene como finalidad, la creación de un simulador diseñado para entrenamiento de operadores, implementando una nueva estrategia simulada aplicada al proceso de los espesadores de relave. Esta estrategia de programación no solo busca sentar las bases para la creación de un simulador, sino que también permite obtener resultados en diferentes estados de operación, brindando la oportunidad de observar el comportamiento del espesador de relave bajo diversas condiciones.

Dada la naturaleza de las dinámicas internas de los espesadores, que suelen ser lentas en su respuesta, este informe describe la creación de un simulador dinámico utilizando como estrategia la vinculación de distintos lenguajes de programación, específicamente Fortran en complemento con Python mediante el módulo F2py [9]. La finalidad de esta combinación es la creación de un simulador con un rendimiento óptimo en términos de velocidad de procesamiento, aplicado a grandes volúmenes de datos, de tal manera que permita visualizar largas contantes de tiempo en intervalos más acotados.

1.2. Trabajos Previos

En esta sección, se llevará a cabo la revisión bibliográfica que abarca diversos trabajos vinculados al tema de esta investigación. El propósito fundamental de la revisión bibliográfica permite profundizar en la teoría de sedimentación y el modelo matemático que describe el fenómeno.

El documento no pretende ser exhaustivo, sino más bien contribuir como alternativa al modelamiento de espesadores, presentando una herramienta de valor que permita evaluar las dinámicas al interior de los espesadores de relave. En primer lugar, se llevará a cabo un análisis de la documentación, contextualizando el desarrollo histórico del modelo matemático de sedimentación e identificando la instrumentación utilizada por estas máquinas.

En segundo lugar, la revisión se enfocará en el análisis del modelamiento matemático de los espesadores de relave y el fenómeno físico que los rige. En tercer lugar, la revisión se enfocará en el análisis de las técnicas simuladas aplicadas a los espesadores de relave.

En cuarto lugar, se hace referencia a los trabajos relacionados con los modelos matemáticos y se explorara en la vinculación de ambas plataformas de programación, dando a conocer el diseño del simulador dinámico de espesadores. Para validar la respuesta del simulador dinámico, se llevarán a cabo diversas pruebas, que permitan evaluar la estabilidad en el tiempo bajo distintas condiciones de operaciones. Finalmente, se procederá analizar los datos resultantes, con el objetivo de presentar las dificultades y ventajas en la aplicación de la estrategia de programación utilizada, proporcionando consideraciones relevantes y posibles mejoras en el simulador orientado tratamiento de los espesadores de relave.

1.1.1 Simulación numérica para sedimentación floculadas.

♣ R. Bürger, F. Concha, “Mathematical model and numerical simulation of the settling of flocculated suspensions”, International Journal of Multiphase Flow., vol.24, pp. 1005-1023, (1998). [1]

En este documento, se presenta una solución en estado transitorio para el proceso de sedimentación discontinua y continua para suspensiones floculadas. El modelo matemático contempla, la utilización de la ley de conservación escalar, válida para el análisis en interior del espesador. El modelo matemático queda en función de dos ecuaciones complementarias, la ecuación hiperbólica, para el estudio del fenómeno de decantación, y la ecuación parabólica, para la capa de sedimentos compresibles. Para dar forma al modelo matemático, entre algunas de las restricciones consideradas en el análisis, se destacan:

- Se considera un valor crítico cuando los fóculos comienzan a tocarse entre sí.
- La suspensión está totalmente floculada al comienzo del proceso de sedimentación.
- Tanto los componentes sólidos como líquidos se comportan como fluidos elásticos.

El documento también enfatiza en un análisis en profundidad de la teoría de Kynch obteniendo un marco matemático formal que permite clasificación de los procesos de sedimentación.

1.1.2 Simulación numérica del comportamiento transitorio de suspensiones floculadas en un espesador ideal o continuo.

- R.Bürger, M.C. Bustos, F. Concha, "Settling velocities of particulate systems: 9. Phenomenological theory of sedimentation processes: numerical simulation of the transient behavior of flocculated suspensions in an ideal batch or continuous thickener", International Journal of Miner Process., vol. 55, pp. 267-282, (1999). [2]

El artículo aborda un enfoque analítico en la teoría de sedimentación, centrándose en el problema de valores iniciales y de contorno al interior del espesador. La finalidad principal del documento es comparar los resultados con estudios previos y predecir el comportamiento de los espesadores de relave bajo distintas condiciones durante la carga, descarga o al modificar las variables de entrada y salida. Además, el artículo presenta modelos de simulación que permiten visualizar el comportamiento transitorio de un espesador continuo cuando es sometido a distintas condiciones en el flujo de llenado y vaciado, como también validar mediante simulaciones la correcta caracterización de las ecuaciones que representan la teoría de sedimentación en un espesador.

1.1.3 Comparación del proceso de sedimentación con resultados publicados.

- P. Garrido, R. Bürger, F. Concha, "Settling velocities of particulate systems: 11. Comparison of the phenomenological sedimentation-consolidation model with published experimental results", International Journal of Miner., vol. 60, pp.213-227, (1999) [3].

El documento se contextualiza bajo condiciones de simulación para el diseño y control de espesadores modelo, centrándose en dos funciones: *(i) función de densidad* (f_{bk}) y *(ii) función tensión efectiva* (σ_e). Estas funciones desempeñan un papel fundamental en la determinación de los coeficientes necesarios para modelar la teoría de sedimentación mediante la ecuación hiperbólica-parabólica.

La novedad destacada es la consideración del floculante como factor relevante en la formación de flóculos. En este sentido, el modelo matemático incorpora como variable independiente el efecto del floculante, la que puede ser ajustada en el controlador.

1.1.4 Métodos numéricos fiables y no fiables para simulación en tratamientos de aguas residuales.

- ♣ R. Bürger, Stefan Diehl, Sebastian Faraz, Ingmar Noppen's, "On reliable and unreliable numerical methods for the simulation of secondary settling thanks in wastewater treatment", Computers and Chemical Engineering., vol. 41, 93-105, (2012). [4].

El documento expone la metodología de análisis mediante simulación, cuyo propósito permite evaluar el rendimiento del espesador en dos escenarios: *(i) Método de convergencia (G)* y *(ii) Metodología que permite reducir errores números en la operación de plantas de tratamiento de aguas residuales (EO).* El enfoque principal de la investigación consiste en realizar un análisis comparativo entre el método (G) y método (EO), con el objetivo de validar la confiabilidad del método (G) propuesto.

Como parte del análisis, se busca determinar la precisión con la que los tiempos requeridos en la simulación concuerdan con la realidad, al mismo tiempo que se busca evidenciar las limitaciones y validaciones del nuevo método (G) y la obtención de resultados.

1.1.5 Controladores de balance de masa para sedimentación continua.

- ♣ Fernando Betancourt, Fernando Concha, Daniel Sbárbaro, "Simple mass balance controllers for continuous sedimentation", Computers and Chemical Engineering., vol. 54, pp. 34-43, (2013). [5].

El documento propone una solución simplificada, comparada con los extensos modelos matemáticos que detallan el proceso de sedimentación en espesadores, como también, maximizar la recuperación de agua en las zonas periféricas del espesador. La investigación se centra en la implementación de un controlador PI no lineal, capaz de estabilizar el funcionamiento del espesador, mediante una estructura de control simple.

Con tal de validar la estructura de control, se retoman las suposiciones realizadas en la teoría cinemática Kynch, que describe la sedimentación de una suspensión ideal para pequeñas esferas rígidas del mismo tamaño. Asumiendo que todas las variables de flujo dependen de la profundidad del espesador y del tiempo. El diseño matemático se sostiene en la base en los modelos visto previamente [3].

El análisis divide principalmente al espesador en cuatro zonas de operación: *(i) zona de espesamiento (*$0 < z < z_R$*), (ii) zona de clarificación (*$z_L < z < 0$*), (iii) zona flujo inferior (*$z > z_R$*), (iv) zona desbordamiento (*$z > z_L$*).*

El flujo de alimentación al interior del espesador comienza en la condición $z = 0$, con una velocidad de alimentación Q_F y una concentración de sólidos ϕ_F. El flujo inferior se caracteriza por Q_R cuya concentración ϕ_D. Finalmente, el desbordamiento producido en los costados del espesador se Q_L y su concentración ϕ_E.

Las condiciones que se buscan alcanzar mediante el controlador PI son: *(i)* Producir una concentración de desbordamiento baja (muy cercano al valor 0) *(ii)* Producir una concentración de descarga superior a un valor mínimo y por debajo de un valor máximo *(iii)* ser robusto a variaciones en las variables de alimentaciones.

1.1.6 Modelo y control para espesadores alimentados por suspensiones con propiedades dependiente del tiempo.

♣ Fernando Betancourt, Raimund Bürger, Stefan Diehl, Sebastian Faraz, “Modeling and controlling clarifier-thickeners fed by suspensions with time-dependent properties”, Minerals Engineering., vol. 62, pp. 91-101, 2014. [6].

El documento, permite ampliar el modelo de espesador-clarificador conocido, considerando los cambios en las propiedades físicas de los sólidos, que ingresan al espesador. El modelo contempla la velocidad de sedimentación, que depende del factor de floculantes de las partículas sólidas. Para ello, se propone un modelo unidimensional capaz de variar las propiedades de floculación en función del tiempo. Este modelo se describe como un sistema de dos ecuaciones diferenciales parciales no lineales que se ajusta mediante una variable independiente *(k)* que permite controlar la dosificación de floculantes.

La importancia del control de partículas se basa en que la velocidad de sedimentación depende del tamaño de las partículas y de la diferencia de densidad entre sólidos y líquidos que a su vez se rige por la velocidad Stokes.

El nuevo modelo aborda las posibilidades y limitaciones del control manual mediante el ajuste parcial de las dosis de floculantes. Además, se detallan los pasos en el proceso de floculación, que incluye la etapa de desestabilización de las partículas suspendidas, la formación y crecimiento de flóculos y la degradación de floculas, controlados principalmente por la variación de pH.

1.1.7 Manual de filtración.

- Fernando Concha, “Manual de filtración & separación”, Talcahuano, Chile: Universidad de Concepción, 2001. [7].

Este libro, tiene su enfoque en la etapa del proceso mineral, específicamente en el proceso de separación liquido-solido, basándose en la teoría de diseño, operación y optimización de los procesos involucrados. El libro hace referencia a la teoría evolutiva en el diseño de espesadores, presentando los principales actores en la teoría de sedimentación, como también en el refinamiento del modelo matemático a través de los siglos.

1.1.8 Uso eficiente de aguas en la industria minera y buenas prácticas.

- Subsecretaría de economía consejo nacional de producción limpia, “Uso Eficiente de Aguas en la Industria Minera y Buenas Prácticas”, Gobierno de Chile, Ministerio de Minería Subsecretaría de Minería, Noviembre 2002. [8].

Este manual tiene como objetivo dar a conocer la problemática de los escases hídrica en Chile y el impacto que tienen las empresas en la utilización de este recurso natural. También permite tener una herramienta practica para orientar una gestión responsable en el consumo hídrico. Mejorar las prácticas en la faena minera y dejar en evidencia la necesidad de una gestión de responsable de los recursos hídricos que permitan la conservación del medio ambiente y el uso racional de ella.

1.1.9 Guía del usuario y manual de referencia de F2py.

- NumPy. (16 de Septiembre de 203). F2py Users Guide and Reference Manual. https://numpy.org. [9].

La página se presenta como un manual detallado sobre como generar el vínculo entre los lenguajes de programación Fortran y Python. Se inicia con las instrucciones específicas para la instalación de librerías NumPy, clave para realizar de manera exitosa la integración. Numpy ofrece una herramienta completa con la finalidad guiar al usuario en la construcción del módulo F2py. La guía incluye comandos y ejemplos prácticos que reflejan cada paso del proceso, facilitando la comprensión y aplicación de las técnicas descritas. Esta guía no solo se limita a la aplicación de un solo método, sino que proporciona diversas mejoras con tal de optimizar el rendimiento y la confiabilidad de las simulaciones.

1.1.10 Informe de tendencia del mercado del cobre.

- Comisión Chilena del cobre, “Informe de tendencias del mercado del cobre, proyecciones para los años 2024 y 2025”, Gobierno de Chile, Ministerio de Minería, Diciembre 2023. [10].

El documento tiene como objetivo analizar la demanda minera en Chile y presentar una proyección detallada del balance mundial del cobre. Para responder a la demanda de producción minera, se proyecta la demanda para los años 2024 y 2025. Además, se contextualiza y respalda el balance de producción de cobre, destacando la influencia de factores políticos, energéticos y tecnológicos, y como estas variables pueden afectar la demanda del cobre a nivel y su exportación.

1.1.11 Discusión

Los espesadores de relave son máquinas industriales de vital importancia en el proceso de separación sólido-líquido. Este proceso comienza cuando la pulpa, proveniente de la etapa de flotación, ingresa al espesador de relave a través de una bandeja de alimentación [8]. Una vez que la pulpa se encuentra en el interior del estanque, las partículas suspendidas tienden sedimentarse gradualmente en el fondo del contenedor, lo que da lugar al fenómeno de sedimentación. La revisión bibliográfica permite describir matemáticamente la teoría de sedimentación, en particular analizando las ecuaciones hiperbólicas para el asentamiento de partículas y parabólicas para la consolidación de los sedimentos [1] [2][**7**]. El análisis se puede llevar a cabo bajo dos suposiciones, que abordan tanto la sedimentación continua como la discontinua [2] [4] [7]. Para validar el modelo de sedimentación, se realizan una serie de simulaciones en torno a la teoría matemática bajo diversas condiciones de estabilidad. Esto permite evaluar los parámetros óptimos para el flujo de alimentación y descarga, con el fin determinar los cambios necesarios para acercarse a un equilibrio en estado estacionario. Históricamente se han realizado múltiples estudios y diversas ecuaciones, que confirman que el modelo de sedimentación representa fielmente el comportamiento físico producido al interior de un espesador de relave [2][3].

Aunque muchos, actores describen del mercado de las plantas de concentrados “en un limbo” debido a su lentitud, desorden y a las constantes interrupciones en la producción de concentrado, ya sea por eventos significativos (sociales, medioambientales, políticos, económicos e instrumentación) que contribuyen a una inestabilidad en el mercado [10]. Mas de la mitad de estas interrupciones se pueden atribuir a problemas técnicos y a la lenta puesta en marcha de las operaciones mineras [10], dejando en evidencia la necesidad de implementar estrategias confiables, capaces de mejorar las condiciones en las plantas concentradoras. Con tal de generar una contribución al área, se busca implementar métodos de simulación que permitan optimizar respuestas de manera precisa y cuyos tiempos de procesamiento de datos se realicen en lapsos más acotados de tiempo (CPU). Esto es especialmente útil cuando se buscan probar distintas estrategias de control y métodos de análisis [3].

El análisis simulado no solo permite comparar tiempos de procesamiento de datos, sino que también proporciona una solución numérica más exacta mediante la manipulación de grandes volúmenes de datos, siempre y cuando se mantenga por debajo de cierto umbral de error y se cumplan restricciones de estabilidad cruciales para garantizar la convergencia en la utilización de métodos numéricos [4][5].

El controlador PI no lineal, permite representar la estabilidad en el proceso de sedimentación bajo el criterio de estabilidad CFL para esquemas numéricos [4][5], manteniendo al mismo tiempo una estructura de control simple [5]. La elección de aplicar estrategias de control depende completamente de las necesidades del proceso industrial en cuestión [5].

Dado que los espesadores de relave son máquinas diseñadas para la separación sólido-líquido, el control del contenido sólido no es tarea trivial. Factores como la diferencia de densidades y la variación en el tamaño de las partículas sólidas pueden impedir que el proceso alcance un estado óptimo. Es por ello que la adición de químicos, como los floculantes, se convierte en una práctica habitual para acelerar el proceso de sedimentación. El control preciso de la dosificación de floculante en los estanques de relave es primordial, aunque también un desafío complejo. Por lo tanto, se vuelve necesario desarrollar nuevas estrategias de control que permitan manipular la dosificación de floculantes, si lo que se busca es obtener resultados lo más cercano a la realidad [5] [6].

Se decide finalmente tomar como prioridad de estudios los documentos [6] y [**7**] para la creación del simulador para espesadores de relave, utilizando la estrategia de programación mediante la aplicación del módulo F2py, que permite vincular dos lenguajes de programación Fortran y Python [9], cada uno de estos lenguajes cumple un rol distinto en la programación ya que Fortran corresponde a un lenguaje compilado, a diferencia de Python [9] que corresponde un lenguaje interpretativo, por lo que para la creación del simulador dinámico se busca utilizar lo mejor ambos mundos.

1.3. Hipótesis de Trabajo

Mediante el vínculo de dos lenguajes de programación, es posible lograr una optimización significativa en los tiempos de procesamiento, en conjunto con el manejo de grandes volúmenes de datos, permitiendo la manipulación a conveniencia en las variables de entrada y salida del sistema. Este enfoque permite establecer las condiciones en la creación del simulador dinámico que tenga prestaciones en cuanto eficiencia y versatilidad, para el análisis del proceso de sedimentación y al mismo tiempo, optimizar los tiempos de respuesta del proceso.

1.4. Objetivos

1.1.12 Objetivo General

Se propone un simulador dinámico que permita optimizar los tiempos de respuesta en el proceso de sedimentación, mediante el tratamiento de grandes volúmenes de datos, utilizando el vínculo entre un lenguaje compilado Fortran y un lenguaje interpretativo Python.

1.1.13 Objetivos Específicos

- Conocer el proceso de espesamiento y su importancia en el proceso productivo industrial y en el aspecto social / medioambiental.
- Comprender la caracterización dinámica de espesadores según bibliografía existente.
- Comparar velocidad de cómputo entre Fortran y Python.
- Desarrollar simulador dinámico vinculando código Fortran y lenguaje Python.
- Evaluar desempeño del simulador dinámico.

1.5. Alcances y Limitaciones

El propósito del proyecto es fomentar el desarrollo de habilidades de simulación, implementación y comprensión del proceso de estudio. El documento pretende ser una alternativa de simulación destinada a mejorar los tiempos de respuesta en un proceso en el que la instrumentación y control son críticos. Para ello, se realizarán pruebas simuladas, con tal de verificar la calidad del simulador.

- El simulador dinámico propuesto tiene como objetivo abordar el problema de las respuestas lentas en el procesamiento de datos para el control de espesador.
- Las pruebas llevadas a cabo en el simulador se realizarán bajo condiciones propuestas en los documentos utilizados como referencias para proyecto.
- De acuerdo a la literatura la dosificación de aditivos químicos se considerará como una variable ideal y no se forma parte del modelo.
- Se idealizará el tamaño de las partículas sólidas que ingresan al espesador, considerando un estado de uniformidad entre ellas.
- Debido a la magnitud de los espesadores y la falta de instrumentación, no es factible emplear parámetros a escala, por lo que el proyecto se restringe a la utilización de parámetros simulados.

El simulador representa el comienzo de una serie de proyectos futuros en los cuales se buscará la oportunidad de expandir y optimizar el modelo para presentar una representación más precisa de los estados reales con la que operan estos espesadores. La incorporación de rastras, sensores, válvulas y toda la instrumentación correspondiente son parámetros que están fueran del estudio, como también la implementación de dosificación de floculantes.

1.6. Temario y Metodología

El presente informe se organiza en varios capítulos, cada uno con un enfoque especifico que contribuye a la comprensión y desarrollo del simulador dinámico para espesadores. A continuación, se resume el contenido de los capítulos que engloban la investigación.

Para llevar a cabo esta investigación, la metodología de trabajo comienza con la revisión de diversos recursos, tales como, trabajos de investigación, charlas, conferencia, y documentos relacionados con los espesadores de relave y la teoría del proceso de sedimentación. Este apartado tiene como objetivo introducir al lector en la importancia que tiene los espesadores de relave en el área industrial y los impactos ambientales que conlleva la falta de instrumentación y control deficiente. Luego, se profundiza en el diseño y la instrumentación utilizados en la creación de estos espesadores.

El siguiente capitulo se centra en una revisión del estado del arte en relación en cuanto a las topologías tradicionales de los espesadores de relave. Este análisis aborda conceptos fundamentales, incluyendo los procesos físicos y químicos involucrados en el proceso de sedimentación y presta especial atención a los principales modelos matemáticos.

A continuación, se presenta la propuesta del simulador dinámico para espesadores, destacando la utilidad mediante la combinación de dos lenguajes de programación complementarios: Python y Fortran. Esto se logra mediante un módulo para la conexión fluida entre ambas plataformas. Esta etapa sienta las bases del simulador dinámico, con el objetivo de utilizar el modelo matemático validado en la literatura para el modelamiento de espesadores. Para ello, se detallan todos los pasos recorridos en el trascurso de esta investigación, desde su concepción en la teoría de espesamiento y la dinámica del proceso, que se produce en los espesadores de relave.

La estructura del presente documento ha sido diseñada de tal manera que permita dar a conocer la importancia de los espesadores de relave y los desafíos que enfrentan debido a falta de instrumentación y el mal control. Como propuesta, se da pie a la creación de un simulador dinámico que se beneficia del vínculo entre dos plataformas complementarias. A lo largo de este informe, se destaca la utilidad y ventajas potenciales que esta solución puede ofrecer.

Capítulo 2. Espesadores de Relave

2.1. Introducción

El presente capitulo, se centra en exponer la problemática asociada a los escases hídrica en las zonas mineras y la manera en que las industrias toman acciones y contribuyen a la conservación de los recursos naturales, presentando el enfoque en los principales centros de recuperación de agua denominados espesadores de relave y su contribución al tratamiento de agua y concentrado de minerales. Los espesadores de relaves son considerados maquinas críticas en los procesos industriales ya que funcionan de forma continua, cuya función principal es facilitar el tratamiento de minerales mediante el proceso de separación sólido-liquido a través de métodos mecánicos.

Bajo esta premisa, se analiza el estado del arte de los espesadores de relave, abordando principalmente su diseño y características, con especial enfoque en el proceso de separación sólido-líquido y la recuperación de agua producida. El propósito es profundizar en la topología de los espesadores de relave y su funcionamiento como también dar a conocer la teoría que rigen estos procesos e identificar las principales variables involucradas que intervienen, con tal de comprender como estos sistemas contribuyen al tratamiento de aguas y concentrados de minerales.

2.2. Importancia del tratamiento de aguas

La actividad minera, por su naturaleza productiva involucra el tratamiento de grandes volúmenes de material, para ello se instalan plantas de procesamiento, tranques de relave, estanques y piscinas de almacenamiento de agua. Las grandes empresas de la minería del cobre producen alrededor de un 94% de cobre fino nacional [10] esta producción implica que existan condiciones en el consumo hídrico que ayudan a contribuir en el uso eficiente del agua.

- Un mayor consumo de agua y por consiguiente una reducción en la disponibilidad del mismo recurso natural.

- Contribuir en el uso racional y eficiente del agua con la finalidad de reducir el consumo por tonelada tratada, incluso ante un aumento en la producción.

En el proceso industrial el tratamiento de minerales queda reflejado en el diagrama de flujo simplificado Fig. 2.1. Una vez tratado el concentrado en el proceso de flotación, es enviado posteriormente a la etapa de espesamiento para luego ser enviado al proceso de filtración, con la finalidad de reducir la cantidad de agua contenida del concentrado [7]. En la zona de espesamiento donde trabajan los espesadores de relave se produce la recuperación de agua. Si bien cuando es posible la recuperación de agua, una porción es destinada al uso industrial y el resto se devuelve al medio ambiente, bajo condiciones controladas [8].

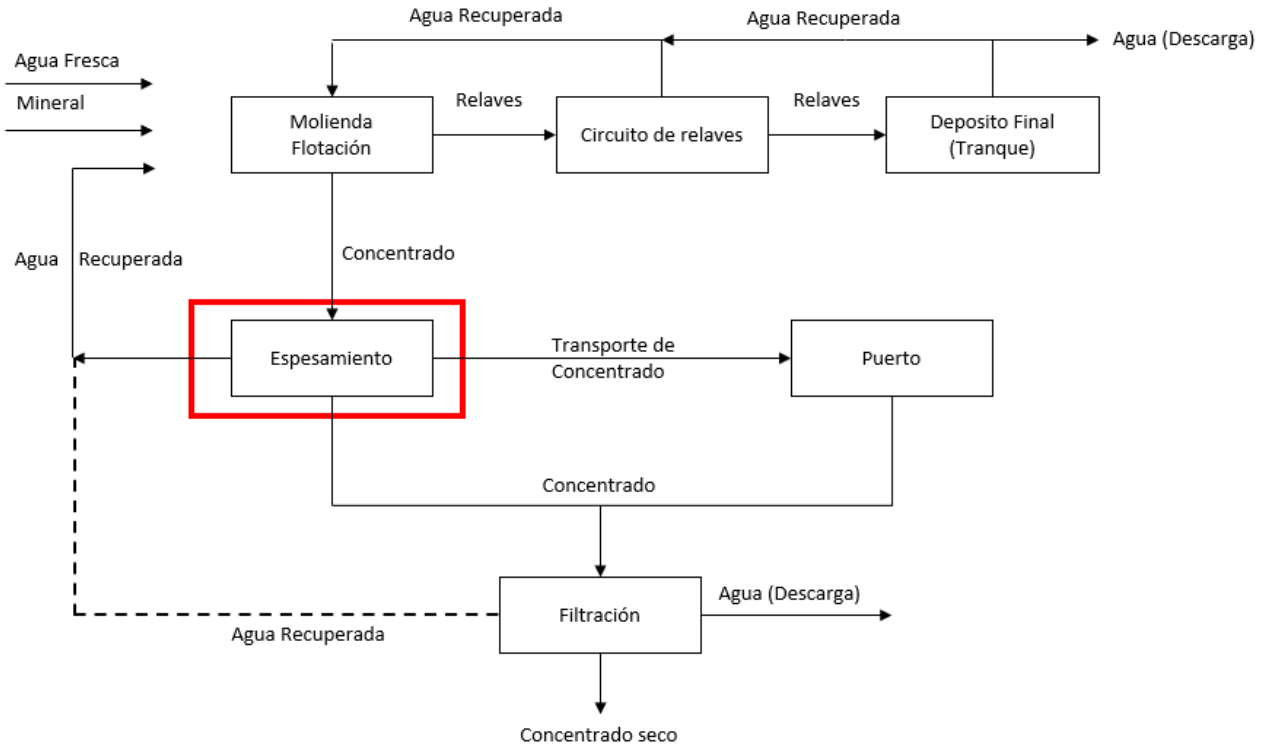

Fig. 2.1 – Diagrama de flujo básico de una planta concentradora [8].

La mezcla resultante del proceso de flotación es enviada a la etapa de espesamiento, donde se separa una porción del agua contenida en la pulpa Fig. 2.1. Una vez completo el fenómeno de sedimentación, el material sólido se extrae y se envía a depósitos finales, como los tranque se relave o depósitos simulares.

2.3. Espesadores de relave

Generalmente estos estanques tienen una estructura con cuerpo cilíndrico y fondo cónico, diseñados para contener grandes volúmenes de residuos provenientes de etapas previas del procesamiento mineral comúnmente denominados como relaves Fig. 2.1. En el interior de estos estanques, se encuentra una mezcla compuesta por un porcentaje de sólidos y un porcentaje de líquido. La particularidad de estos sistemas reside principalmente en su función de separar eficientemente el contenido solido del líquido. Esto permite que el agua recuperada sea recirculada hacia otros procesos del ciclo minero o que sea liberada de manera segura y controlada al medio ambiente Fig. 2.1. El contenido sólido almacenado en la parte inferior cuando alcanza cierto nivel de compactación debe ser transportados y evacuados de manera segura. Estos pueden destinarse a instalaciones de disposición final, como relleno de minas o directamente ser depositados en tranques de relave. Estos últimos principalmente son estructuras diseñadas para el almacenamiento seguro de sólidos residuales. La importancia del control de los espesadores de relave contribuye en evitar contaminación del medio ambiente por lo que su diseño y mantenimiento debe garantizar la seguridad medio ambiental. La estructura básica de un espesador se ilustra en la Fig. 2.2.

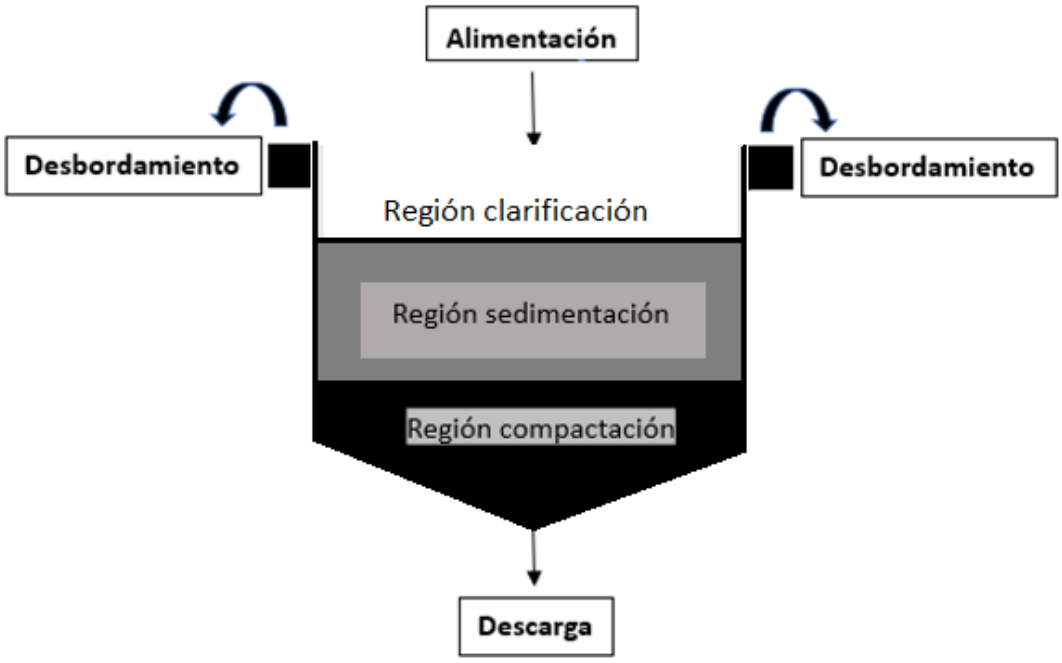

Fig. 2.2 – Estructura tradicional de un espesador de relave.

La estructura tradicional que tienen los espesadores de relave se presenta en la Fig. 2.2 y principalmente están constituidos por una bandeja de alimentación, sistema de descarga y sistema de recuperación por desbordamiento o rebalse en las zonas periféricas del diseño y se detallan a continuación.

- Bandeja de Alimentación:

La bandeja de alimentación de acuerdo con su diseño característico se ubica en la zona centro del estanque Fig. 2.2. Este diseño presenta beneficios en cuanto a facilitar y mejorar la distribución de la pulpa que ingresa al espesador, como también facilita que la mezcla contenida en el estanque sea distribuida uniformemente. La fuente de alimentación tiene como objetivo crear un canal entre la etapa de flotación y la zona de espesamiento Fig. 2.1. El enfoque integral de la fuente de alimentación ubicada en la zona centro del espesador y contribuye a mejorar la eficiencia en el manejo e integración de la suspensión, asegurando un flujo controlado y uniforme para el proceso.

- Sistema de Descarga:

La zona de descarga de un espesador de relave se sitúa en la parte inferior de la estructura, desempeñando un rol esencial en la extracción eficiente del contenido sólido aglutinado en la parte inferior por efecto de la sedimentación. Esta región, está diseñada con una forma cónica que facilita y agrupa las partículas sólidas en el centro del espesador, formando niveles de capa de sólidos. En consecuencia, la forma cónica proporciona un gradiente descendente que contribuye en la separación eficaz entre la interfaz sólido y líquido.

- Sistema recuperación por desbordamiento:

La zona de recuperación de agua por desbordamiento está diseñada por canaletas que facilitan la recuperación eficiente de agua clarificada. Este diseño tiene en cuenta la acumulación de sólidos almacenados en la parte inferior del espesador. La presión ejercida entre el contenido sólido aglutinado permite comprimir los poros de los sólidos y por consiguiente un aumento en el nivel de agua del espesador, generando un desbordamiento en las zonas periféricas y por ende la recuperación de agua clara.

La gestión cuidadosa del flujo agua a través de las canaletas es esencial, ya que permite garantizar que el agua recuperada cumpla con estándares establecidos, bien sea para la reutilización de agua clara en el proceso minero o para su liberación de manera segura.

- Rastras:

El sistema mecánico proporcionado por las rastras tiene la función de desplazar el contenido sólido acumulado en el fondo del espesador hacia la abertura de descarga, gracias al movimiento giratorio ejercido en el interior del espesador. La labor desempeñada por las rastras es esencial, especialmente considerando la alta concentración de sólidos en la parte inferior del espesador. Los espesadores al compactar un gran contenido de sólidos pueden ocasionar problemas ya que cuando acumula una gran cantidad estos pueden exceder el torque de las rastras y por consiguiente que exceda la resistencia mecánica.

2.4. Proceso de sedimentación

Una vez que el lodo proveniente de etapas previas como lo es la etapa de flotación Fig. 2.1, es depositado al interior del espesador de relave dando comienzo al fenómeno físico de sedimentación. Las partículas sólidas que se encuentran suspendidas en el líquido tienden a precipitar hacia el fondo del espesador y por consiguiente dando comienzo al fenómeno de sedimentación, como se muestra en la Fig. 2.3.

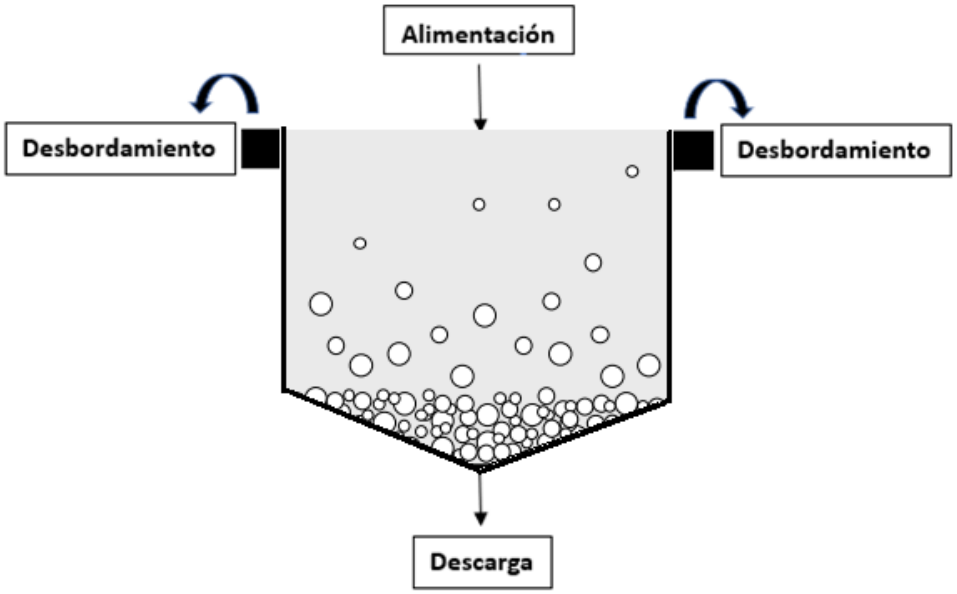

Fig. 2.3 – Proceso de sedimentación del sólido.

El proceso de sedimentación no es un asunto trivial y dependiendo del sistema que se esté utilizando se puede categorizar bajo la actuación de distintos fenómenos físicos, el espesador se encuentra en la categoría de la sedimentación gravitacional Fig. 2.4. Factores como la presencia de partículas de distinto tamaño, el peso, las distintas densidades en la interfaz de sólidos-líquido y la viscosidad del fluido, representan desafíos significativos para lograr una eficiencia y control óptimo en estos centros de almacenamiento [7]. Partículas extremadamente finas por efecto de la gravedad tienden ralentizan el proceso de sedimentación, ocasionando la lenta precipitación del contenido sólido hacia el fondo del tanque. Para abordar este desafío, se recurre a diversas estrategias como la incorporación de aditivos químicos denominados floculantes.

Estos aditivos permiten aglutinar partículas sólidas en cadenas de polímeros, formando conglomerado de sólidos de mayor tamaño y por consiguiente dando uniformidad a la concentración y por ende aumentando la velocidad con la que precipitan hacia el fondo del espesador. Este proceso contribuye a la velocidad de sedimentación y a la uniformidad en la separación de fases solido-líquido y como se depositan en el fondo del espesador. No obstante, la dosificación de floculante debe ser precisa y ajustada en función de la capacidad y diseño del espesador.

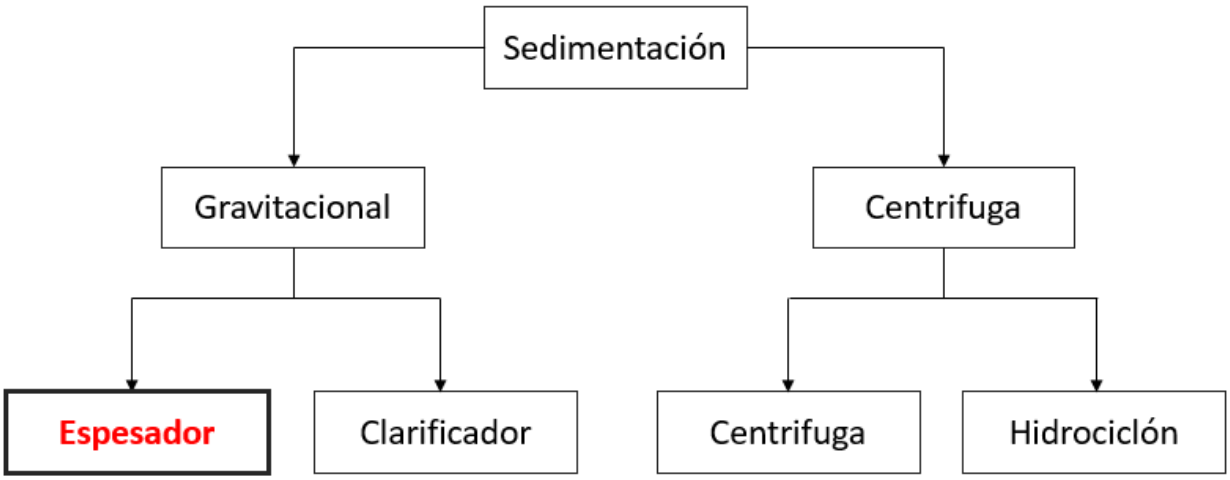

Fig. 2.4 – Equipos para el tratamiento de sedimentación.

Históricamente desde la invención del espesador existen limitaciones en cuanto instrumentación disponible por lo que modelar el sistema condujeron a la realización de diversas pruebas y experimentos que contribuyeron al entendimiento del fenómeno de sedimentación [7].

La invención del espesador Dorr en 1905, puede ser considerado el punto de partida de la era moderna de espesamiento [7]. Inicialmente, los análisis fueron realizados en un sistema de *batch* para sedimentación, el cual representaba un proceso semi continuo. Sin embargo, con la introducción del espesador Dorr, se dio paso a una evolución en la teoría hacia procesos continuos. Esto se logró mediante la implementación de una bandeja de alimentación capaz de suministrar de forma continua el contenido sólido y evacuado por una zona descarga.

A su vez los primeros modelos matemáticos del proceso de sedimentación se introdujeron en el análisis para la *teoría de sedimentación mediante el Kynch* en 1952 [**4**].

Este modelo sentó las primeras aproximaciones para describir matemáticamente que la sedimentación está basada en una ecuación diferencial parcial hiperbólica de primer orden. Kynch demostró que, conociendo la concentración inicial de la suspensión y la densidad de flujo de sólido, se puede obtener una solución de la ecuación por el método de características, segmentando distintas zonas de en qué la concentración varía en forma continua [7] [5][3].

2.4.1 Sedimentación continua

El proceso de sedimentación continua fue estudiado con el propósito de dar una explicación al comportamiento de los espesadores a nivel industrial y como las partículas se asientan de forma continuada en el fondo del espesador. El análisis se enfoca en comprender de manera exacta el proceso en donde las partículas sólidas decantan de manera constante y sin interrupciones en un periodo de tiempo. La particularidad del proceso radica en que la sedimentación continua a diferencia de otros procesos de separación ocurre de manera ininterrumpida en un sistema idealizado, esto implica que el efecto de roce ejercido en las paredes del espesador, como también el tamaño de partículas sólidas contenidas al interior sean simplificadas a características ideales, estas condiciones se cumplen mientras que las partículas sólidas sedimentan gradualmente hasta consolidarse en el fondo del espesador.

Un aspecto clave en este proceso es la aplicación de la teoría de Kynch, que proporciona un margen teórico que permite comprender el movimiento de partículas en un sistema. La teoría de Kynch utiliza los principios de la mecánica de fluidos y la variación en las concentraciones de partículas sólidas en un determinado tiempo durante del proceso. Gracias a las ecuaciones diferenciales se puede proveer el comportamiento dinámico de la sedimentación para proceso industriales permitiendo modelar con precisión la variación en las concentraciones de partículas sólidas en función del tiempo.

2.4.2 Relevancia Teoría de Kynch

El modelo de Kynch tiene una profunda influencia en el desarrollo de la teoría de sedimentación ya que permite comprender la velocidad de sedimentación en un medio continuo. Para el análisis, las partículas sólidas sedimentan a una velocidad constante.

Kynch demostró que en función de la concentración y la velocidad de sedimentación se puede obtener una ecuación por el método características. Esto implica que los resultados sean reflejados en distintas zonas con variaciones en la concentración a medida que el contenido particulado se asienta de forma continua en el fondo del espesador Fig. 2.5.

Esta subdivisión permite comprender de manera efectiva el funcionamiento del proceso y la separación producida entre la interfaz sólido y líquido.

Para el análisis de los espesadores de relave, se distinguen cuatro zonas de operación [5]. Estas zonas describen el movimiento transitorio de las partículas al interior del espesador de relave, como se observa en la Fig. 2.5.

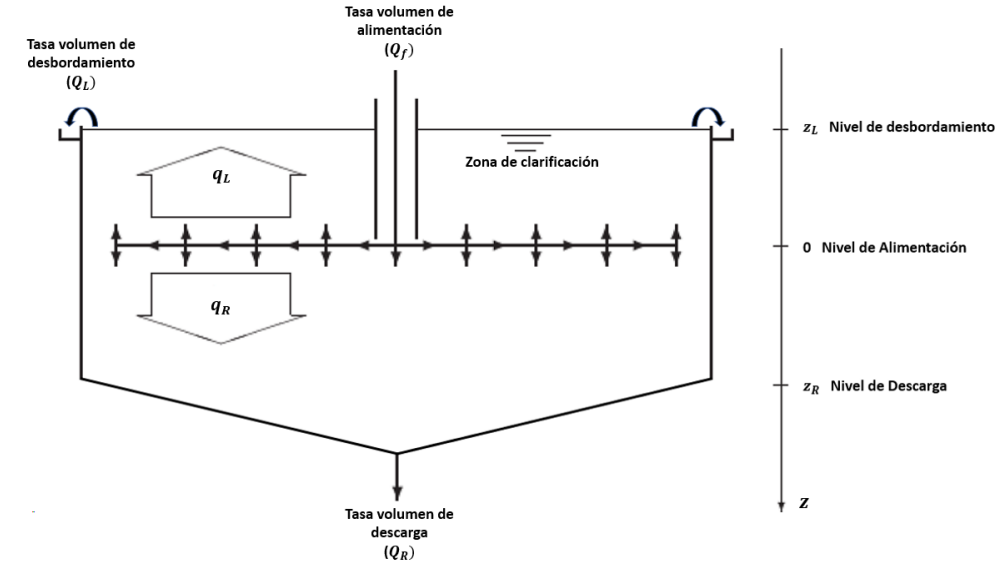

Fig. 2.5 – Modelo segmentado de un espesador [5].

- Zona de Clarificación

Esta fase marca el punto de inicio del proceso de sedimentación, donde las partículas sólidas, en conjunto con el contenido líquido, se sitúan en la parte superior del espesador de relave. En esta etapa, las partículas sólidas se encuentran próximas a la superficie del espesador, dando comienzo al proceso de sedimentación.

- Zona de sedimentación

La zona de sedimentación comprende el movimiento de las partículas sólidas, a medida que comienzan a descender.

Este fenómeno se produce por la influencia de la gravedad sobre las partículas sólidas, como también las distintas densidades existentes entre la interfaz sólido-líquido, además de la convección inducida por la velocidad de flujo. En esta etapa del proceso, el material solido contenido comienza el proceso de asentamiento decantando hacia el fondo del espesador.

- Zona de transición

Esta zona comprende el punto de transición que representa el punto de cambio entre la zona de sedimentación y la zona de compresión. Aquí, las partículas sólidas han completado el proceso de sedimentación y se aproximan hacia el fondo del espesador.

- Zona de compresión

En esta región del espesador, se encuentra la concentración máxima de partículas sólidas. El contenido solido se asienta en el fondo del espesador y se comprime, formado capas compactas que en su conjunto se denomina cama de sólidos.

En consecuencia, de acuerdo con Fig. 2.5, el sistema se abastece en el nivel $z = 0$ a través de un caudal constante de alimentación (Q_F) y una concentración de alimentación (ϕ_F). Por otro lado, en el nivel z se ubica la zona de descarga del espesador (Q_R) con una concentración de (ϕ_R). Además de la interacción entrada-salida, en los flancos del espesador se sitúa el nivel de rebalse en z_L cuya tasa volumétrica esta representada (Q_L) [5].

2.5. Discusión

La limitada instrumentación y los análisis realizados en medios semi continuos dieron lugar a teoría y diseños que se acercaban más a la realidad de las aplicaciones prácticas. Inicialmente, el análisis se centraba en medios semi continuos, lo que implicaba la realización de experimentos en lotes individuales donde el material tratado, en lugar de ser un flujo continuo como lo es en el caso industrial, estudiaban el comportamiento de pequeñas cantidades de material en cada experimento, en lugar de visualizar el comportamiento del proceso en su totalidad y de manera continua. Si bien esta metodología era viable pero limitada para procesos industriales.

En cuanto al modelamiento matemático, la teoría de Kynch supuso un avance significativo al describir el comportamiento de partículas y su proceso de sedimentación. Sin embargo, existen limitaciones en su aplicación al modelo real en el tratamiento de minerales, especialmente en lo que respecta a las diferentes densidades de las concentraciones de sólidos y despreciar el efecto producido de las paredes del espesador, como influencia de la interfaz sólido-liquido. Sin embargo, existen simplificaciones en el modelo propuesto realizado principalmente en los modelos teóricos para facilitar el modelo matemático y su comprensión en la teoría del proceso de sedimentación. Dicha idealización, fue una limitación en los modelos de los espesadores, ya que no reflejaban condiciones reales del sistema, ya que las partículas sedimentan a diferentes velocidades debido a su tamaño y densidad. Esto condujeron a la idealización del análisis, asumiendo que todas las partículas sólidas contenidas al interior del espesador tienen el mismo tamaño y densidad.

Si bien, para abordar dichas limitaciones el análisis en la inyección de químicos que permiten aglomerar partículas sólidas formado un conglomerado de mayor tamaño, esto proporciono beneficios en cuanto a la velocidad de sedimentación estandarizando el material en elementos de mayor tamaño.

Capítulo 3. Modelación Dinámica de Espesadores

3.1. Introducción

El siguiente capitulo tiene como finalidad central explorar la modelación dinámica de los espesadores, para ello se explora en como los principios matemáticos influyen en el control de un espesador. Esta combinación de acuerdo a su diseño se presenta como una herramienta para análisis y aplicación en la comprensión en relación del fenómeno de sedimentación y la estrategia utilizada para este tipo de sistema.

A lo largo del capítulo se detallarán y profundizará en el análisis del modelo teórico, cómo como también sentar las bases en el modelamiento en la creación del simulador para espesadores. Se pretende generar una de información que permita dar a conocer matemáticamente el modelamiento de los espesadores, como también tener una visión detallada sobre la estrategia de control que tienen estos equipos.

3.2. Instrumentación de un espesador de relave

El correcto monitoreo de las variables de entrada y salida implica la supervisión de un conjunto de equipos cuyo objetivo es proporcionar una respuesta en tiempo real sobre las condiciones de operación del espesador. La instrumentación permite garantizar la continuidad del proceso, ya que permite identificar los tradicionales sistemas de medición, asegurando condiciones de trabajo aceptables y evitando posibles fallos en el proceso.

3.2.1 Sistema de medición

- Presión de la cama

Para medir la presión en la zona inferior del espesador, se utiliza una premisa basada en un indicador de gravedad especifica global al interior del equipo, proporcionando el volumen de partículas que se aglomeran en la zona inferior. La gravedad es prácticamente constante y se correlaciona con la cantidad de solidos depositados en la parte inferior del espesador.

Para llevar a cabo la medición de la presión de la cama de sólidos, se emplean sensores con el objetivo de compensar posibles variaciones en la presión debido a la compactación. El diseño característico del espesador, cuya forma tradicional es cónica, contribuye a prevenir la obstrucciones o acumulación excesiva de sólidos, asegurando que los sólidos puedan sedimentar.

- Nivel de cama

El nivel de cama es medido por sensores encargados de monitorear la altura aproximada de un nivel especifico de solidos en el espesador. Esta medición modela el fenómeno de sedimentación, que crea un perfil de concentración de partículas que se extiende desde la superficie hasta el fondo del espesador.

- Válvula descarga

La válvula de descarga es un componente crucial del diseño. Se encarga de regular el flujo de material que sale por la parte inferior del espesador de relave, enviado hacia el siguiente proceso o destino. Esta válvula permite controlar la velocidad y la cantidad de relave que se descarga, lo que es fundamental para mantener un flujo constante y controlado del material a lo largo del proceso.

A medida que las partículas se depositan en el fondo del espesador, se forma una cama de concentrado. El monitoreo continuo del nivel de cama permite optimizar y garantizar la eficiencia del espesador, evitando la compactación excesiva al interior del espesador de relave.

3.3. Modelamiento del espesador de relave

3.3.1 Ecuación hiperbólica parabólica fuertemente degenerada a la ley de conservación

El movimiento del contenido sólido al interior del espesador se ve afectado por fuerzas dinámicas, ya que la transición de las partículas suele decantar de forma gradual Fig. 2.3. Para modelar este fenómeno, se deben definir las ecuaciones que gobiernan el sistema, la cual se expresa mediante una ecuación parabólica fuertemente degenerada que evoluciona hacia una ley de conservación hiperbólica de primer orden cuando $\phi < \phi_{critica}$ [5]. Esta ecuación, se denota mediante (3.1) y proporciona una descripción simplificada del transporte de partículas en el interior del estanque de relave.

$$\frac{\partial_\phi}{\partial_t} + \frac{\partial}{\partial_z}(q\phi + f_{bk}(\phi)) = 0 \tag{3.1}$$

donde,

ϕ : Concentración volumétrica de sólidos.

f_{bk} : Función densidad de flujo de sólidos.

De acuerdo (3.1), se expresa en función de la densidad de flujo de sólidos (f_{bk}), de acuerdo a esta expresión se puede desprender la ecuación de Kynch para el análisis sedimentación.

3.3.2 Flujo densidad de sólidos (f_{bk})

El fenómeno de sedimentación, expresado entre la relación de la concentración de sólidos, en función de la velocidad volumétrica y el flujo de densidad de sólidos. Estos modelos se fundamentan en la teoría cinemática de Kynch, para describir el comportamiento de pequeñas partículas sólidas suspendidas en un medio líquido. Este enfoque permite profundizar en el comportamiento de las partículas y como interactúan en un entorno acuoso basado en la ley de conservación. Las características de un sólido mediante la función de densidad flux se ve expresado en la (3.2).

$$\phi_t + f_{bk}(\phi)_z = 0 \tag{3.2}$$

Donde $\phi(t)$ representa la fracción volumétrica de sólidos en función de la profundidad del espesador *z* y del tiempo [**5**].

3.3.3 Estado de equilibrio

El comportamiento de una suspensión en un sistema con un área transversal constante requiere del análisis de las ecuaciones de balance, especialmente cuando en el sistema confabulan dos medios de distintas densidades. En el contexto de un estado de equilibrio para la interfaz de sólido-líquido la importancia que tiene el movimiento de las partículas sólidas en función de la velocidad decantamiento que ingresan al espesador. Para ello es necesario remitirse a la expresión que asume la ecuación de conversación de masas en (3.3).

$$\phi_t + (q(t)\phi + \phi(1-\phi)v_r)_z = 0 \tag{3.3}$$

donde,

v_R : Velocidad relativa entre fase sólida y liquida.

$q(t)$: Velocidad de la suspensión en función del tiempo.

En este contexto, $q(t)$ es la velocidad aparente de la suspensión, mientras que v_r se denota como la velocidad relativa. Dicho esto, se puede analizar el comportamiento de la concentración de sólidos suspendidos en un medio liquido a lo largo del tiempo y el espacio. De acuerdo, con la "*Ecuación de Balance*" resultante, la teoría cinemática de Kynch, que postula la existencia en la superposición de ambas velocidades $v_r = v_r(\phi)$, expresada en términos de la función flux de Kynch. Esta condición se expresa en (3.4) y (3.5).

$$v_r = \frac{f_{bk}(\phi)}{\phi(1-\phi)} \tag{3.4}$$

Reemplazando la (3.2) en la expresión (3.4), se obtiene la siguiente expresión (3.5).

$$\phi_t + (q(t)\phi + f_{bk}(\phi))_z = 0 \tag{3.5}$$

3.4. Control para espesador

3.4.1 Definición del Controlador Proporcional Integral (PI)

En el contexto del controlador Proporcional Integral (*PI*) es un método utilizando ampliamente en el control de sistemas industriales, incluyendo los espesadores de relave. Su principal función es ajustar las señales de control en tiempo real, cuya finalidad es mantener una variable especifica dentro de los márgenes aceptables de estabilidad. En el contexto del espesador de relave, el control *PI* se aplica para mantener la concentración de sólidos lo más cerca posible a los valores deseados. De acuerdo, a la Fig. 3.1 el punto de ajuste se ve representado por $\phi_R(t)$ indica la cantidad de sólidos que se desea obtener en un determinado tiempo.

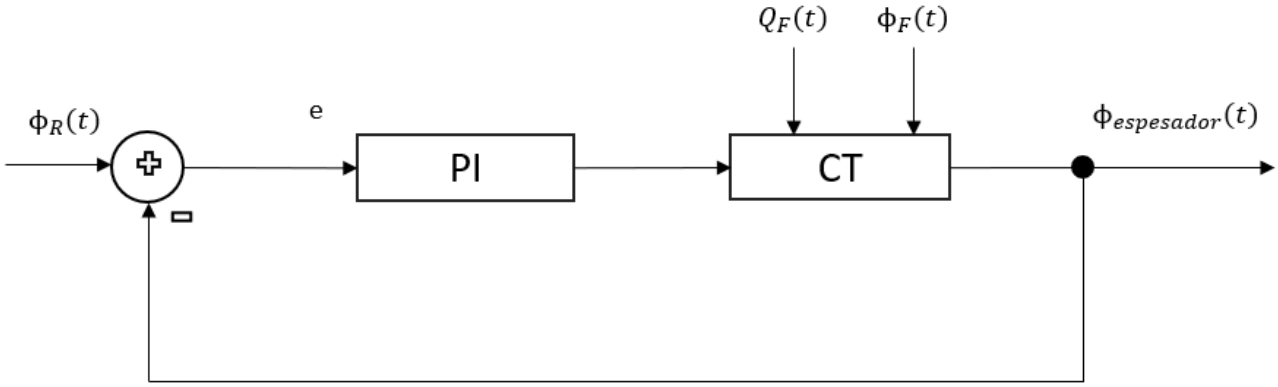

Fig. 3.1 – Modelo de control de un espesador convencional [5].

Al trabajar solamente con el enfoque Proporcional-Integral el controlador PI, permite que el sistema pueda eliminar el error en estado estacionario. Si bien al trabajar únicamente con un controlador P, este puede alcanzar un estado estacionario (e.e), pero no garantiza que el error sea nulo.

Para lograr un control óptimo del sistema, existen dos aspectos relevantes asociados al $error$ del control [5]:

- Aspecto Proporcional

Bajo estas condiciones el sistema se ajusta la señal de control para la parte proporcional que se asume como el error actual del sistema, entre la salida deseada y la salida real del proceso. Esta diferencia se de acuerdo a la Fig. 3.1 se ve representada como la concentración de sólidos requerido $\phi_R(t)$ y la concentración de sólidos medida del sistema $\phi_{espesador}(t)$, bajo estas condiciones se controla mediante la variable Proporcional (K_p).

- Aspecto Integral

Para la condición integral de control (K_i), es considerada como la acumulación de errores en tiempo pasado. Las condiciones mencionadas para el error de control medida en el tiempo t, se define mediante (3.6).

$$e_{(t)} = \phi_R - \phi_{espesador} \tag{3.6}$$

donde,

$e_{(t)}$: Error de control en el tiempo.
ϕ_R : Variable de control deseada.
$\phi_{espesador}$: Variable de control medida.

La salida del controlador *PI* considerado en cada instante de tiempo, se calcula como la suma de ambos términos mencionados el aspecto proporcional y el aspecto integral. Ya que de acuerdo a lo mencionado la parte proporcional ajusta la señal de control de acuerdo al error actual y a su vez la parte integral considera la acumulación de errores pasados a lo largo del tiempo (t). Para ello se refleja mediante (3.7).

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau \tag{3.7}$$

donde,

$u(t)$: Señal controlada en el tiempo.

K_p : Variable proporcional.

K_i : Variable integral.

Por lo tanto, para mantener el control de las concentraciones en el espesador, se utiliza la estructura que contempla un controlador *PI* esto permite mantener la concentración descarga $\phi_R(t)$ de sólidos en valores deseados de estabilidad.

3.4.2 Sintonización del controlador PI

Si bien para la sintonización del controlador Proporcional – Integral (PI), no se utilizó un método formal de ajuste. En su lugar, se adoptó un enfoque empírico, probando distintos valores para los parámetros K_p y K_i hasta lograr la estabilización deseada del sistema. El método utilizado, aunque no cumple con una estrategia propia, permite iterar y observar el comportamiento del sistema en tiempo real, ajustando los parámetros hasta lograr minimizar el error y obtener una respuesta estable. Dado que el sistema es no lineal, y contiene ecuaciones diferenciales parciales es difícil linealizar, por lo que la sintonización está hecha a prueba y error.

3.5. Discusión y Conclusiones

Los principales sistemas de control empleados en los espesadores de relaves a nivel industrial están asociados a sensores que permiten medir y modelar el comportamiento del sedimento [**7**]. Sin embargo, debido a que estos sistemas dependen de múltiples dispositivos de medición, pueden ser ineficaces en caso de fallas. Por ello, la implementación de un simulador dinámico integrado permite reducir la instrumentación necesaria y mejorar el modelamiento bajo diversas circunstancias.

Además, se presenta el modelo matemático centrado en el balance de masas y el análisis basado en la teoría de Kynch, se detalla el diseño de controlador Proporcional-Integral (PI). Este controlador se implementa con el propósito de regular las concentraciones (ϕ) en niveles predeterminados. El objetivo es lograr que el control de las concentraciones se mantenga estable a lo largo del tiempo en respuesta a variaciones en la señal controlada asociado a la descarga. Para ello, el controlador PI utiliza la diferencia entre la concentración deseada y la concentración real como error de entrada, ajustando así el flujo de descarga con tal de minimizar esta discrepancia.

El controlador PI al corregir el error en estado estacionario (e.e) permite ajustar la salida del sistema a lo largo del tiempo, actuando al eliminar cualquier diferencia entre el valor deseado de concentraciones y el valor actual de las concentraciones, una vez que el sistema ha alcanzado un estado estable o estacionario igual a 0.

Debido a que el sistema es no lineal y contiene ecuaciones diferenciales parciales, su linealización resulta un problema complejo. Los sistemas no lineales no responden de manera proporcional a las entradas, esto implica que pequeñas variaciones en los parámetros de control puedan generar grandes cambios en la respuesta. Además, el sistema al estar regido por ecuaciones diferenciales parciales, las variables involucradas en la descripción del sistema dependen de múltiples factores que cambian con el tiempo, en el caso del espesador el nivel de las concentraciones. Por ello, se optó por un método de ensayo y error para la sintonización del controlador PI, ajustando las variables K_p y K_i de forma empírica.

Capítulo 4. Evaluación de Lenguajes de Programación para Simulación

4.1. Introducción

La aplicación de análisis simulado en procesos como los espesadores, ofrece una herramienta valiosa que busca mejorar la comprensión de estos procesos. El enfoque simulado permite la evaluación de lenguajes de programación con tal de crear una reconstrucción del modelo matemático con la finalidad de modelar la respuesta de un proceso evaluando problemas operativos y mejorar la continuidad del diseño. Mediante el proceso simulado se permite identificar variaciones en los estados de carga en función de caudal de alimentación (Q_F) y una concentración de alimentación (ϕ_F) y por consiguiente la caudal de descarga (Q_R) y la concentración de descarga (ϕ_R). El análisis por trozos correlacionado con el movimiento de partículas busca una aproximación que garantice la correcta continuidad del proceso.

Se explora en la influencia simulada en los espesadores, centrándose en la evaluación de los distintos lenguajes de programación comúnmente utilizados como *Fortran*, *Python*, *C++*, *C*, entre otros. A través del análisis cualitativo, se destacan las ventajas y desventajas en su implementación con tal de justificar la opción seleccionada para el desarrollo del simulador. El simulador busca ofrecer prestaciones en cuanto a manejo de grandes volúmenes de datos, como también la velocidad de ejecución de estos en tiempos más acotados.

La particularidad del simulador pretende utilizar y seleccionar un lenguaje de programación accesible y simplificado. Si bien la programación puede funcionar de forma independiente, también se explora en la posibilidad de vincular dos lenguajes de programación complementarios.

4.2. Lenguaje Programado Fortran

El código fuente utilizado en el simulador se basa en el uso del entorno de programación aplicado para el diseño del espesador de relave en Fortran. Conocido por su enfoque en la programación científica y el cálculo numérico es una herramienta de programación de alto nivel. Este lenguaje de programación caracterizado principalmente, por la potencia en la resolución de cálculos matemáticos, se convierte en una elección natural para aplicaciones que requieren un rendimiento óptimo en términos de precisión y eficiencia computacional. La capacidad para manejar grandes volúmenes de datos y realizar cálculos numéricos de manera eficiente.

Fortran aplicado al diseño de espesadores, se beneficia en la capacidad de realizar cálculos matemáticos aplicados a ecuaciones diferenciales. Esto permite predecir el rendimiento del equipo bajo diferentes condiciones operativas como lo son la variación en estados de alimentación (Q_F), descarga (Q_R) o desbordamiento (Q_L) del sistema. Las principales ventajas y desventajas, orientado a la creación del simulador dinámico útil, para en análisis de espesadores, se esquematiza en Fig. 4.1.

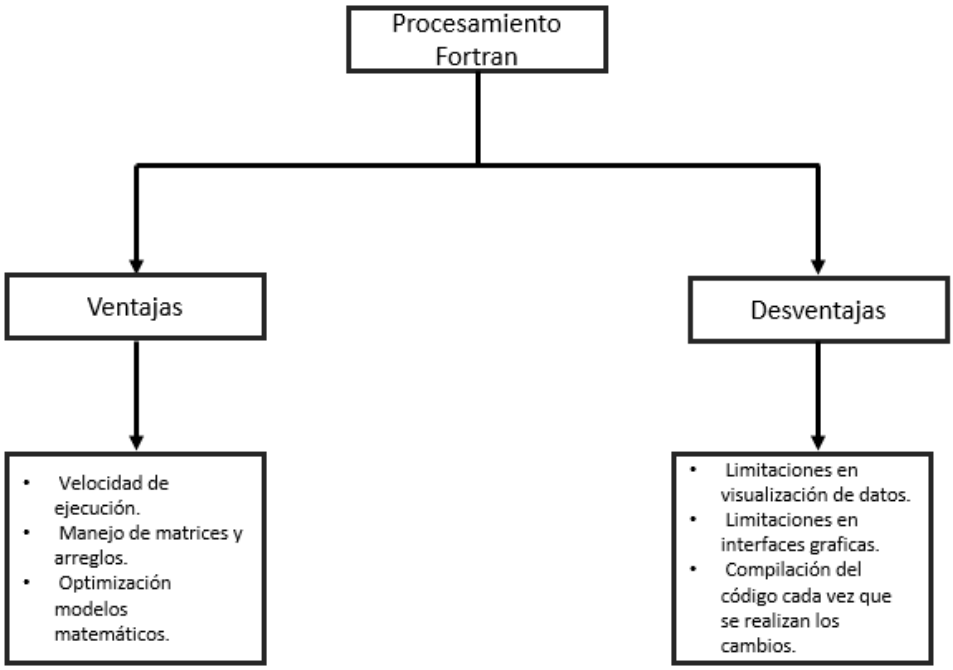

Fig. 4.1 – Ventajas y desventajas en la aplicación Fortran.

En la actualidad existen varias versiones normalizadas del lenguaje Fortran. ANSI 198-1992, que tiene sus bases en “Programan Lenguaje Fortran Extended” más bien conocido como Fortran.90 ($f90$), esta versión se considera como una evolución del lenguaje Fortran tradicional incorporando módulos y sobrecarga de operadores junto con nuevos tipos de datos.

- Ventajas entorno Fortran

La eficiencia de Fortran destaca por su naturaleza optimizada para cálculos numéricos, lo que se traduce en obtener resultados en tiempos de ejecución más acotados en comparación con otros lenguajes. Un punto para destacar es el control y manipulación de grandes volúmenes de datos en forma de matrices multidimensionales. Esta característica permite una programación más concisa y legible, al mismo tiempo que optimiza el rendimiento en operaciones matriciales complejas.

Fortran y su capacidad de manejar ecuaciones diferenciales permite la creación de modelos precisos, particularmente relevantes en aplicaciones como el diseño y la optimización asociados a espesadores de relave.

- Desventajas entorno Fortran

En cuanto a las limitaciones en la visualización de datos y en las interfaces gráficas son aspectos relevantes para considerar la utilización de Fortran. Este entorno de programación carece de la capacidad de integración directa para la creación de gráficos y construcción de interfaces, lo que limita la visualización de los resultados y la interacción con el sistema, si lo que se busca es que la herramienta sea dinámica y capaz de adaptarse a diversas circunstancias. Además, la necesidad de volver a compilar el código generando un nuevo archivo cada vez que se realizan cambios ya sea por la variación de parámetros o ecuaciones se considera una desventaja significativa lo que puede ralentizar significativamente el análisis del sistema. Este proceso repetitivo implica un tiempo adicional y una mayor complejidad en el desarrollo y depuración del código, lo que ralentiza el proceso de iteración cada vez que se requiera realizar nuevas iteraciones en cada nueva modificación requerida.

4.3. Lenguaje Programado Python

El entorno de programación Python, es ampliamente reconocido por su versatilidad y facilidad de uso, especialmente en el ámbito de la programación científica y de ingeniería. Este lenguaje proporciona bibliotecas interactivas y herramientas que facilitan el análisis de datos, la implementación de algoritmo complejos y la creación de modelos matemáticos. Python, destacada por su enfoque simple y legible, lo que se convierte en una excelente opción para modelar el comportamiento de un espesador de relave. Su capacidad para manejar grandes volúmenes de datos lo convierte en una herramienta beneficiosa a la hora del análisis en la estabilidad del espesador, considerando su visualización grafica. Para su aplicación las bibliotecas especializadas, como "*Numpy"* y *"SciPy"*, proporciona funciones y métodos optimizados para realizar análisis matemáticos de manera eficiente. La integración de Python con herramientas de visualización de datos como "*Matplotlib"* permite visualizar el rendimiento del espesador bajo diferentes condiciones operativas y por consiguiente optimizar su diseño que permita garantizar la estabilidad del sistema.

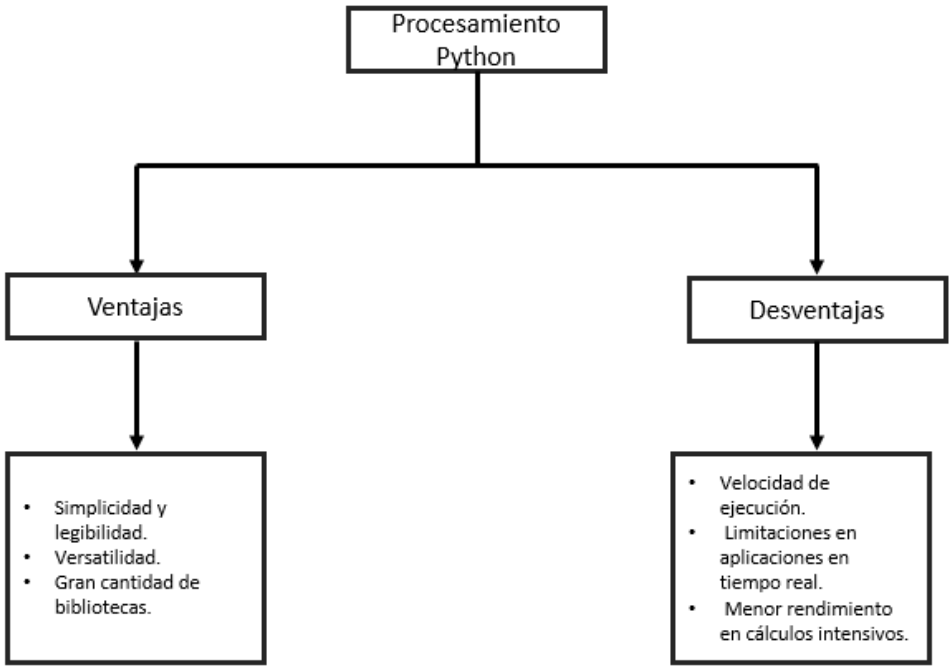

Fig. 4.2 – Ventajas y desventajas en la aplicación Python.

- Ventajas entorno Python.

En primer lugar, su simplicidad y legibilidad inherentes en $Python$ lo convierte en un lenguaje accesible para la programación lo que facilita la estructura y comprensión del código. Estas características, combinada con su versatilidad, permite que $Python$ sea utilizado en la optimización de tareas en procesos industriales de manera simulada. Además, la gran cantidad de bibliotecas disponibles especializada para realizar tareas de manera efectiva que permite visualizar el comportamiento de los espesadores de relave en tiempo real.

- Desventajas entorno Python.

La utilización del código en Python también conlleva algunas desventajas que deben tenerse en cuenta. En primer lugar, la velocidad de ejecución de Python puede ser más lenta en comparación con lenguajes compilados como lo son C, C^{++} o *Fortran* lo que puede ser un factor limitante en aplicaciones que requieren un alto rendimiento en tiempos de respuesta rápidos. Además, Python puede mostrar limitaciones en aplicaciones que garantices simulaciones en tiempo real, ya que su naturaleza puede afectar la capacidad en el procesamiento de datos en sistemas dinámicos. Python de acuerdo a la aplicación puede mostrar un rendimiento inferior a la hora de ser comprado con otros lenguajes optimizados, disminuyendo la estabilidad y eficiencia para analizar el proceso en tiempo real.

4.4. Lenguaje Programado Matlab

En el desarrollo del simulador, pretende tener un enfoque en la optimización de procesos productivos. En este contexto, Matlab se presenta como un entorno de programación principalmente dedicado al cálculo numérico y análisis científico.

El entorno Matlab está diseñado para facilitar la manipulación de matrices y vectores de una manera eficiente, convirtiéndolo en un entorno ideal para el procesamiento de datos y cálculo numérico. Su simplicidad en el desarrollo de cálculos permite una herramienta accesible y rápida, agilizando el proceso de diseño y manipulación de datos.

Matlab destaca por su capacidad en la visualización de datos, con representaciones tanto en gráficos $2D$ y $3D$. La capacidad en la visualización de datos es permite interpretar y comunicar los resultados del proceso.

Matlab, cuenta extensas biblioteca que cubren el área principalmente del análisis matemático, las curvas de nivel, las estadísticas, la optimización y procesamiento de señales. En el contexto, surge como opción para el tratamiento de grandes volúmenes de datos, resolución de ecuaciones diferenciales y generar visualizaciones detalladas que permiten comprender mejor el comportamiento dinámico de un espesador.

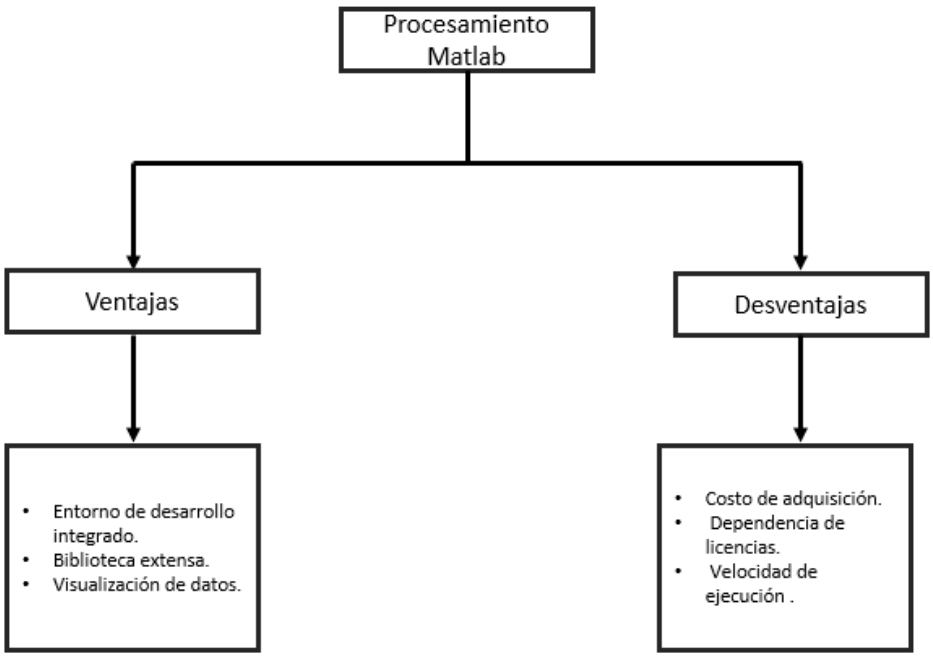

Fig. 4.3 – Ventajas y desventajas en la aplicación Matlab.

- Ventajas entorno Matlab

Matlab ofrece un entorno de desarrollo integrado que facilita la depuración misma del código. Este entorno incluye un editor de texto y la capacidad de visualización de datos que permiten una comunicación fluida para en estudio de los resultados. Además, Matlab cuenta con una extensa variedad de bibliotecas predefinidas que cubren el análisis numérico, facilitando la implementación de distintos algoritmos para la realización de tareas complejas.

- Desventajas entorno Matlab

Adquirir este entorno de programación resulta costoso ya que cuenta con licencia pagada, por lo que adquirirlo puede ser una barrera costosa, especialmente si tiene un contexto de investigación que contribuye a presentar una herramienta de libre acceso. Al ser de uso pagado, existentes limitaciones en cuanto al intercambio de conocimientos.

Si bien Matlab tiene aplicaciones en diversas áreas científicas, en términos de velocidad de ejecución en comparación con otros lenguajes de programación puede verse reducido, especialmente cuando se pone a prueba bajo una extensa cantidad de datos. En cuanto a la información compartida Matlab es de uso cerrado, lo que significa que la información se encuentra limitada a las funcionalidades que proporciona $MathWorks$.

4.5. Comparación Cualitativa de los Lenguajes

A continuación, se desarrolla un análisis cualitativo, entre los lenguajes de programación que potencialmente pueden ser útiles en la creación del simulador. Para ello identificar la estrategia de programación que permita validar y justificar la utilización del lenguaje de programación seleccionado. Para este propósito se analizará diversas características en cuanto aspectos y capacidades de los lenguajes de programación estudiados enfocándose principalmente en características cualitativa y pruebas cuantitativas.

El estudio conlleva, descartar aquellos lenguajes de programación que no sean adecuados para el desarrollo del simulador, ya que cada lenguaje presente sus propias fortalezas y limitaciones, sin embargo, pueden ser significativas a la hora de reflejar el comportamiento de un sistema y sus distintas dinámicas.

- Usabilidad

Se evalúa la facilidad de uso con la que pueden trabajar estos lenguajes de programación. Para ello el análisis se enfoca en aspectos relevantes como la claridad en la sintaxis, la facilidad para comprender, simplicidad en escribir el código. Este enfoque también puede discriminar en cuanto a la cantidad de líneas y recursos que faciliten el proceso de desarrollo. El uso de lenguajes de programación adquiere importancia cuando se aplica a la productividad y desarrollo del simulador, donde un entorno de programación claro y accesible puede acotar considerablemente los tiempos de desarrollo y depuración del código, mejorando la eficiencia ante eventuales fallas.

- Velocidad de Ejecución

La velocidad de ejecución en el marco general del sistema analizado implica evaluar la capacidad del simulador para el manejo de grandes volúmenes de datos y la efectividad en la resolución de cálculos complejos en tiempos de ejecución razonables. Considerando que el sistema analizado, como lo son los espesadores, tiende a tener respuestas transitorias prolongadas para el fenómeno físico de sedimentación. Por lo que el análisis en cuanto a la velocidad de ejecución asegura que el simulador pueda procesar rápidamente la información obteniendo resultados precisos.

- Disponibilidad de Bibliotecas y Funcionabilidad

La disponibilidad de bibliotecas es un punto relevante dentro de los lenguajes de programación mencionados, ya que determina la calidad y variedad de las herramientas disponibles para la lectura de datos y representación gráfica del sistema. Es necesario utilizar el lenguaje de programación que tengan la capacidad de acceder a bibliotecas especializadas que faciliten el análisis de datos y la visualización de los resultados ya que pueden mejorar significativamente la eficiencia del proceso dinámico, permitiendo una mayor versatilidad a la hora de su implementación.

- Costos y Licenciamiento

Se requiere discriminar entre plataformas por su accesibilidad, dado que se evalúa la accesibilidad de un lenguaje de programación esto puede implicar inversiones considerables en termino de licencias de software y recursos adicionales. En el contexto del proyecto, se prioriza una plataforma de uso gratuito, lo que garantiza un acceso universal y elimina la necesidad de incurrir en costos adicionales. Estos factores permiten que el simulador dinámico aplicado en espesadores pueda ser implementado de manera amplia y accesible, sin limitaciones financieras que puedan dificultar la aplicación del simulador.

Alguna de las aproximaciones que se pueden realizar en cuanto al análisis comparativo entre *Fortran, Python, C++, C y Matlab*, se discrimina en función de la capacidad que tiene para responder ante errores, como también la eficiencia y optimización.

- Localización de errores

Los lenguajes compilados como lo son Fortran y C++ tienen la capacidad de detectar errores de forma estática. Esto implica que pueden identificar errores durante la fase de compilación, antes de que se ejecute el programa. A su vez los softwares de interpretados, como Matlab y Python, son capaces de detectar errores de forma dinámica, es decir, a medida se ejecuta el programa.

- Generalizaciones del software

Fortran es conocido por sus especificaciones altamente detalladas y explícitas, especialmente en el ámbito de la computación numérica. Además, tanto vez tanto Fortran como Python ofrecen módulos que permiten la integración de datos, funciones e interfaces de manera conjunta, lo que facilita el desarrollo de aplicaciones complejas.

Para discriminar entre plataformas, se evalúa las cualidades de cada lenguaje de programación en los puntos mencionados previamente. Esto puede implicar inversiones considerables en termino de licencias de software y recursos adicionales. En este contexto, se prioriza una plataforma de uso gratuito, lo que garantiza un acceso universal y elimina la necesidad de incurrir en costos adicionales. Para ello, de acuerdo al repositorio GitHub, dos aspectos claves en el proceso de desarrollo y mejora continua de los proyectos están las solicitudes de extracción y los problemas reportados.

Las solicitudes de extracción representan la propuesta de cambias realizadas por colaboradores externo o internos del proyecto. Esto aborda la corrección de errores hasta la implementación de nuevas características. Dicha revisión contribuye a la colaboración y el intercambio de conocimiento entre los desarrolladores de la comunidad y su proyección se puede observar Fig. 4.4.

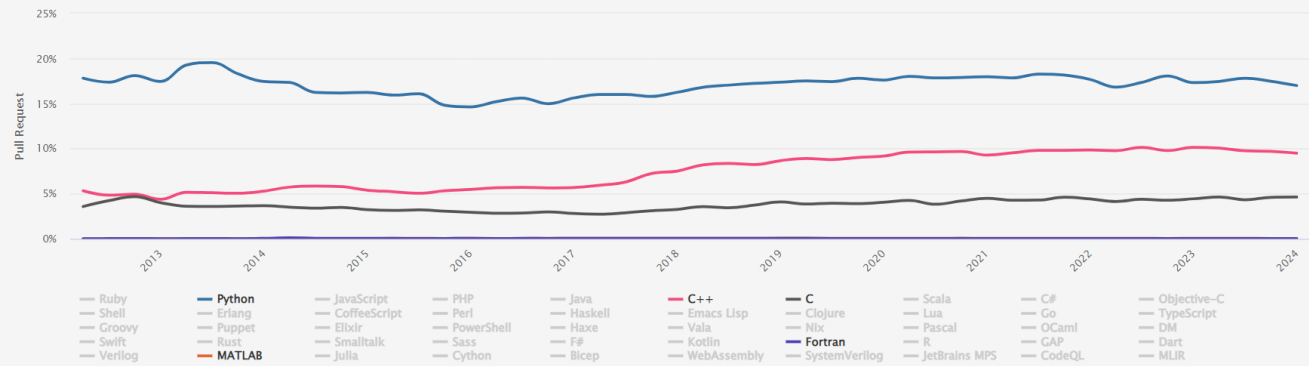

Fig. 4.4 – Repositorio GitHub – Solicitudes de extracción por lenguaje (Python, Matlab, C++, C, Fortran).

A su vez, los problemas reportados son informes de posibles errores, que permiten identificar soluciones y mejoras por los usuarios. Esto puede abordar una variedad de aspectos, como errores de funcionamiento, problemas de usabilidad, solicitudes de nuevas características Fig. 4.5.

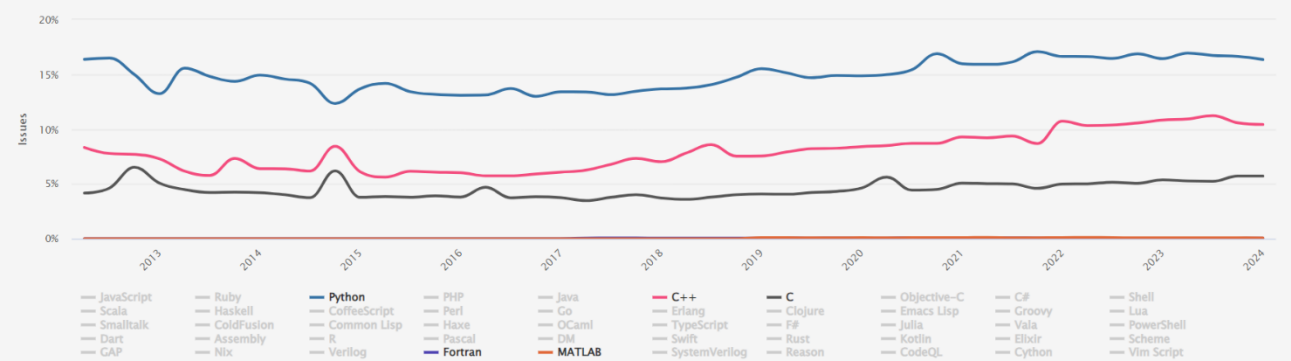

Fig. 4.5 – Repositorio GitHub – Problemas detectados por lenguaje (Python, Matlab, C++, C, Fortran).

En su conjunto las solicitudes de extracción y los problemas reportados permiten una visión amplia de la evolución de un proyecto de software en GitHub. En este contexto se observa el análisis para diversos lenguajes de programación. Se destaca que Python tiene una curva mayor, con valores que oscilan entre el 15% y 20% para ambos aspectos del análisis. Esto puede implicar que Python tenga una comunidad más activa en cuanto a propuestas de cambios e identificación de problemas en el software.

Por otro lado, los lenguajes C++ y C se encuentran en un rango más moderado, entre un 5% y 10% para ambos aspectos Fig. 4.4 y Fig. 4.5, lo que indica que existe una contribución menor por parte de los usuarios en comparación a Python.

En cuanto Fortran y Matlab, se observa 0% en ambos aspectos. Si bien esto puede deberse a diferentes factores, como el tamaño de la comunidad o el enfoque particular en el desarrollo de estos lenguajes. Sin embargo, es importante tener en consideración que Matlab es una plataforma de pago, lo que podría limitar su aplicación.

4.6. Estudio desempeño tiempo de procesamiento de datos utilizando la serie de Fibonacci

La estrategia de programación tiene como objetivo crear una herramienta que ofrezca un alto rendimiento en cuanto a la velocidad de procesamiento de datos y la dinámica eficiente ante los posibles cambios en variables de entrada y salida. Para ello aspectos como la eficacia y el modelo del sistema se ven directamente afectados por aspectos como el rendimiento y el tiempo. Para su comprobación uno de los métodos usualmente utilizados para evaluar el rendimiento de la estrategia de programación implementada es el análisis del tiempo de ejecución mediante pruebas de referencia, para ello se pone en práctica la aplicación de la serie de Fibonacci.

La serie de Fibonacci es una secuencia de números la que permite evaluar el sistema como un conjunto de valores en el que cada número es la suma de los dos anteriores. Esta serie se utiliza para estudiar el rendimiento y tiempo de procesamiento de datos, explorando como varían estos aspectos en función de las técnicas y algoritmo utilizados. Se lleva a cabo un estudio comparativo entre el uso independiente del lenguaje de programación Python y el uso del módulo *F2py.* El objetivo es validar la efectividad en implementación del simulador dinámico y si puede ser considerado como una herramienta valiosa para el análisis del comportamiento de los sistemas de almacenamiento.

Se realizaron pruebas en función de una serie de Fibonacci y se llevó a cabo en una *AMD Ryzen 5 2500U Radeon Vega Mobile Gfx 2.00 GHz* con un sistema operativo de 64 bits. Los softwares de compiladores y entornos de ejecución fueron usados.

- Fortran: 32 Bits.
- Matlab: Versión 2015a
- Python: Versión 3.10.11 64-bit

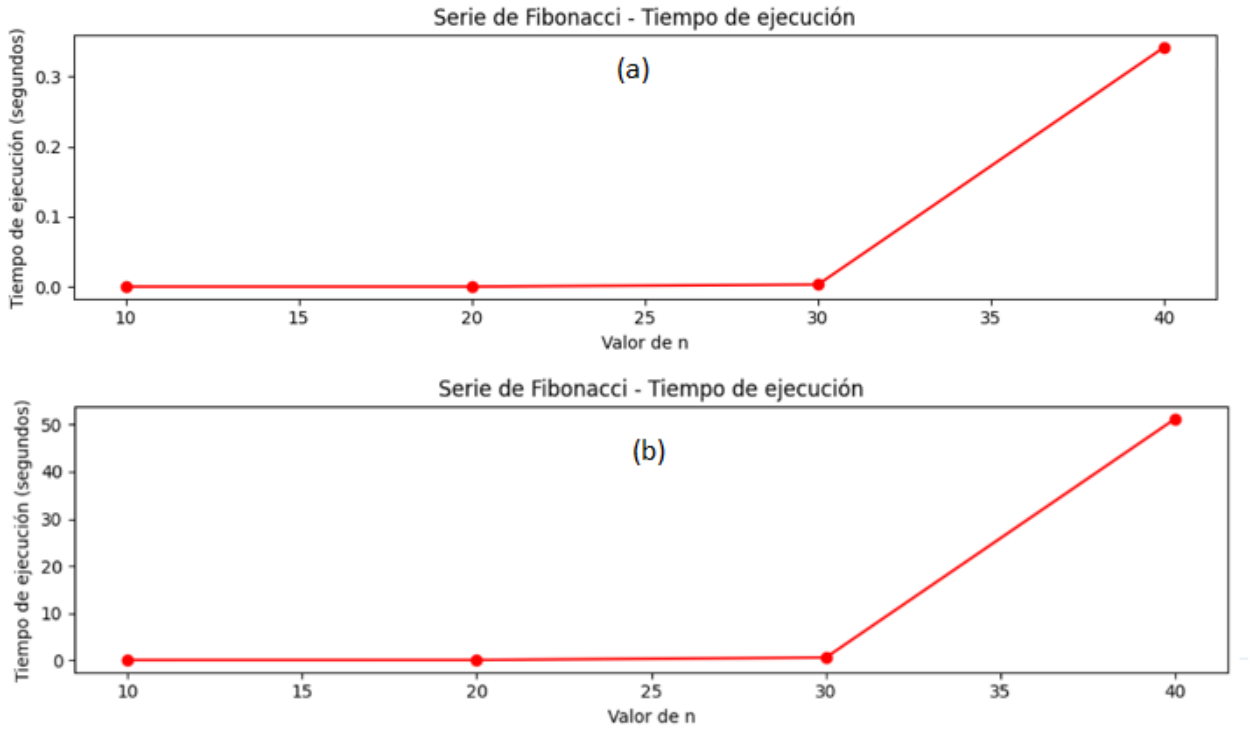

Fig. 4.6 – Simulación comparativa mediante serie de Fibonacci

(a) Aplicación F2py (b) Aplicación Python.

Basándose en Fig. 4.6, se realizó una serie de Fibonacci para analizar la velocidad de ejecución de 40 puntos y el tiempo que demoran en procesarlos. Dado que Fortran tiene sus limitaciones en la visualización de datos, se optó por utilizar el módulo *F2py* que pertenece a una extensión Python y que facilita la vinculación entre un código fuente Fortran con Python.

Tabla 4.1. Análisis cuantitativo simulado.

Software	Cantidad de puntos	Tiempo [s]
Python	40	50
F2py	40	0,3

En este contexto, la implementación del módulo F2py surge como la opción más favorable. Esto se debe a su capacidad para procesar una cantidad de datos en un tiempo considerablemente menor en comparación con otras alternativas. La Tabla 4.1, representa una mejora de hasta 50 veces en la velocidad de procesamiento de datos para la misma cantidad de puntos evaluados.

4.7. Discusión

Por consiguiente, en la implementación del simulador se utiliza el módulo *F2py* que surge como solución para abordar los desafíos asociados a la optimización misma del procesó. Al integrar un código Fortran especializado en modelos de simulación dentro del entorno de programación Python Fig.4.4 se genera una herramienta que facilita la implementación y ejecución de simulaciones detalladas y precisas utilizando la potencia de ambos lenguajes de programación. Dicho vinculo permite la integración para generar una mayor flexibilidad y capacidad de personalización en el desarrollo de modelos, proporcionando un lenguaje de programación diseñado específicamente para la computación científica y de ingeniería, esto implica la manipulación de extensos y complejos códigos. Por lo que el uso del módulo *F2py* no solo simplifica el proceso modelado y la simulación, sino que también proporciona una oportunidad para obtener resultados en tiempos más acotados, esto es beneficioso ya que el proceso de sedimentación es lento.

Capítulo 5. Implementación F2PY

5.1. Introducción

El desarrollo del simulador implica la integración de dos lenguajes de programación complementarios: Fortran y Python. Dicha integración se lleva a cabo mediante el módulo F2py, el cual forma parte de una extensión de Python proporcionada por la biblioteca Numpy. En este contexto, se presenta la utilidad del módulo F2py, abordando aspectos relacionados con su implementación, requisitos y el comienzo de diversas pruebas simplificadas y destinadas a validar la correcta comunicación entre lenguajes.

A lo largo del capítulo se presenta un antecedente de las dificultades y ventajas en la implementación del módulo permitiendo generar una base en cuanto a la instalación, implementación y desarrollo del simulador.

5.2. Módulo F2PY

Python es un lenguaje de programación de alto nivel, este se categoriza dentro de los lenguajes interpretados y multipropósito. Debido a sus características, puede ser utilizado como lenguaje de control para el manejo de programas existente. Dentro de sus múltiples utilidades Python ofrece una extensa gama de herramientas exportadas como bibliotecas, que facilitan el desarrollo y ejecución en diversos campos aplicados al análisis de datos. El módulo F2py proporciona una herramienta que permite vincular dos lenguajes de programación, por una parte, tenemos Fortran que forma parte del conjunto de programas compilados y por otra parte se encuentra Python, utilizado como lenguaje de interpretación. Este tipo de módulos nos proporciona una herramienta sólida a la hora de ejecuciones de grandes volúmenes de datos, un aspecto crucial en el estudio de los espesadores de relave. Debido a que el diseño y simulación analiza el comportamiento del contenido sólido, demostrando, como estas decantan a lo largo del estanque hasta aglomerarse en el fondo del espesador.

5.3. Instalación módulo F2py

5.3.1 implementación desde software

El módulo F2py debe tener instalado previamente la biblioteca NumPy que se encuentra implementado entre las bibliotecas que proporciona Python. Para realizar la instalación se puede recurrir a la utilización del comando “pip”, esta herramienta permite la administración y gestión de paquetes y bibliotecas. Este comando se puede utilizar directamente desde el terminal del sistema Unix asociado a plataformas Linux y MacOS o símbolo del sistema para Windows. Para ello se debe utilizar el siguiente comando.

$$pip\ install\ numpy$$

Las opciones de instalación dependen de la plataforma que se esté utilizando. En este caso se utilizó la plataforma Visual Studio Code (VS), esta plataforma se utilizó considerando la capacidad de soportar múltiples lenguajes de programación, esto implica que pueda estar utilizando Fortran en paralelo con Python, sin tener que recurrir a otros lenguajes o entornos del sistema Windows por separado, ya que este tipo de plataformas mantiene toda su interfaz integrada.

Una vez instalado la biblioteca NumPy, se puede verificar la correcta instalación y la versión asociada al módulo F2py que se está utilizando, para ello desde el mismo terminal se requiere utilizar el siguiente comando.

$$f2py - -versión$$

Es importante reconocer las características con las que se está ejecutando el programa ya que es necesario estandarizar una versión tanto en Python como Fortran. Comprobando las características con las que se desarrollaron las simulaciones, se tiene lo siguiente:

- Versión: 1.24.2
- Numpy Version: 1.24.2
- Requires: Python 3.5 or higer
- License: Numpy license.

Una vez instalado el módulo F2py y comprobado su estandarización en la misma versión que sus tanto para Python como para Fortran se debe recurrir a la compilación del código Fortran, para ello se debe utilizar el siguiente esquema de compilación. Este paso es relevante ya que permite generar el vínculo directo entre ambos lenguajes, la estructura que tiene es la siguiente:

$$f2py - c - m\ mymodule\ funciones.f90$$

El comando anteriormente mencionado, permite principalmente compilar el código Fortran y crear un módulo de extensión que puede ser importando y utilizado en Python.

- $f2py$: Permite generar una interfaz de Python a partir de un código Fortran, esto se incluye al interior de la biblioteca NumPy.
- $-c$: La siguiente opción permite que el módulo F2py debe compilar el código Fortran. Con esta instrucción se crea el módulo en forma de archivo.
- $-m\ mymodule$: Con la asignación "$-m$" se envía la instrucción que especifica el nombre del módulo que se creara. En este caso de ejemplo, se está especificando que el módulo en formato de archivo tenga el nombre "$mymodule$". Si bien el nombre elegido depende del ejecutor este debe seguir las mismas convenciones cuando sea llamado nombrado desde Python.

- $funciones.f90$: En esta última asignación se hace referencia al nombre del archivo fuente contenido en el código Fortran y que F2py va a compilar y convertir en un modulo de Python.

Una vez ejecutado el comando F2py, mediante la codificación expuesta anteriormente, F2py genera un archivo compilado a partir de "$funciones.f90$". Este archivo compilado se convierte en un módulo que puede ser importado en Python. El archivo generado tendrá una extensión "$.so$" en sistemas Linux y MacOS, a su vez para Windows, la extensión generada corresponde a "$.pyd$".

5.4. Pruebas realizadas

Una vez realizado el vínculo entre ambos lenguajes de programación a continuación se exponen ejemplos que permitan determinar la correcta implementación.

Inicialmente se presenta lo que corresponde al código Fortran, en donde se refleja una estructura de la siguiente manera.

```
module mi_modulo
    implicit none

    contains
    subroutine mi_subrutina(x, y)
        real, intent(in) :: x
        real, intent(out) :: y
        y = 2.0 * x
    end subroutine mi_subrutina

end module mi_modulo
```

Las distintas sentencias mencionas en el código Fortran, se define de la siguiente manera:

- "$module\ mi\_modulo$" : Principalmente permite definir el módulo llamado "mi_modulo" .

- "$implicit\ none$" : Permite que las variables sean declaradas explícitamente. Este tipo de declaraciones asegura que las variables sean definidas correctamente como de tipo entero (INTEGER) o de tipo real (REAL), evitando errores en la interpretación.

- "$subroutine\ mi\_subrutina(x, y)$" : Se crea una subrutina llamada "$mi\_subrutina$" que toma dos argumentos.
    - "x": Corresponde a un número real de entrada (input).
    - "y": Corresponde a un número real de salida (output).
    -

- "$real, intent\ (in) :: x$" : Esta asignación es totalmente relevante ya que permite definir las variables que entraran en el sistema, declarando como "x" la variable real de entrada proveniente del código Python.

- "$real, intent\ (out) :: y$" : Se declara "y" como como una variable real de salida.

- "$y\ =\ 2.0\ *\ x$" : Calcula "y" como el doble de "x".

Posteriormente se requiere utilizar el módulo F2py para obtener una vinculo efectivo entre Fortran y Python. Por lo que es necesario remitirse al punto anterior para realizar la siguiente conexión.

Suponiendo que el archivo Fortran se llama "$mi\_modulo.f90$" puede ser compilado utilizando el siguiente comando en el terminal.

$$f2py - c - m\ mi\_modulo\ mi\_modulo.f90$$

Después de completar el proceso de compilación y generar el archivo que será llamado desde Python, es necesario definir las siguientes bibliotecas y sentencias.

```
import numpy as np
import mi_modulo
print(dir(mi_modulo) )

x = np.array(2.0, dtype = np.float(32)
y = mi_modulo.mi_modulo.mi_subrutina(x)
print("Resultado:", y)
```

La definición de cada una de las sentencias se describe a continuación.

- "$import\ numpy\ as\ np$" : Principalmente permite definir el módulo llamado "mi_modulo" . Inicialmente se declara la Liberia Numpy, que tiene dos objetivos principalmente utilizar el módulo F2py como también manejar arreglos tipo de datos números en Python.

- "$import\ mi\_modulo$" : Liberia con la finalidad de importar el módulo Fortran "mi_modulo".

- "$print(dir(mi\_modulo))$" : Se crea una subrutina llamada "$mi\_subrutina$" que toma dos argumentos. Permite imprimir los nombres definidos en el módulo. Con esta determinación se permite comprobar la correcta comunicación entre ambos lenguajes.

- "$x\ =\ np.array\ (2.0, dtype = np.float32)$" : Crea un array NumPy del tipo flotante (float32) con el valor 2.0.

- "$y\ =\ mi\_modulo.mi\_modulo.mi\_subrutina(x)$" : Se llama a la subrutina "$mi\_subrutina$" del módulo creado en Fortran con "x" como argumento.

- "$print("Resultado:", y)$" : Imprime el resultado devuelto por la subrutina Fortran..

Si bien este ejemplo permite demostrar como integrar un código Fortran se vincula con un código Python utilizando el módulo F2py. Aprovechando las ventajas de cada lenguaje para operaciones numéricas.

5.5. Discusión

Este apartando se proporciona como una guía que permite demostrar cómo se realiza de manera efectiva la vinculación entre un código Fortran y un código Python, presentando un ejemplo que detalla dicha interacción. Es importante destacar que uno de los conceptos relevantes es la identificación del tipo de variable que se están declarando, esto implica que puedan ser declarados números enteros o decimales dependiendo de las necesidades del proyecto.

Un punto relevante en la implementación de esta vinculación es definir de manera correcta las variables que serán exportadas desde Python y recibidas en Fortran. Algunas variables que pueden ser definidas internamente en Fortran, mientras que otras pueden ser asignadas desde Python y luego recibidas en Fortran, creando un bucle de comunicación eficiente entre ambos lenguajes.

Como se menciona, es de total importancia verificar y asegurar de que las versiones de Python, NumPy y Fortran, que se esté utilizando son compatibles entre sí. Si bien las diferencias de versiones pueden causar incompatibilidades como también errores al ejecutar el código.

Uno de los puntos particulares en la descripción del código es la sentencia "$implicit\ none$", implementando en Fortran, esto ayudara al código a estipular que las variables sean definidas explícitamente, evitando errores.

Se debe verificar que los tipos de datos en Fortran corresponden a los datos esperados en Python. Esto es de total importante y se vincula netamente con el tipo declaración que se este utilizando, si son enteros o flotantes. La errónea declaración de las variables puede ocasionar errores en la interpretación.

Capítulo 6. Simulador dinámico para Espesadores

6.1. Introducción

El capítulo expone el diseño del simulador mediante el vínculo entre dos lenguajes de programación complementarios Python *(.py)* y Fortran *(f90)*, la estrategia consiste en utilizar el módulo F2Ppy con tal de generar dicha conexión. Se profundizará en el modelo matemático y su implementación con el lenguaje de programación con tal de mantener un control en las concentraciones del sistema. El análisis implica someter el simulador bajo dinámicas que influyan en su operación, con tal de profundizar en el entendimiento y probar la factibilidad de su implementación bajo diferentes condiciones del sistema.

El modelo del simulador está orientado para la realización de pruebas bajo distintos estados de carga. El enfoque evaluado para el simulador tiene como finalidad realizar pruebas mediante dos condiciones experimentales *i) Simulador sin controlador ii) Simulador utilizando controlador PI discreto*. Estos enfoques permitirán justificar la correcta operación del espesador de relave y la capacidad que tiene para ofrecer y mantener un control estable, preciso y eficiente.

De esta manera, se abordará en detalle el proceso de vinculación entre ambos lenguajes de programación Fortran *(. f90)* y Python *(.py)* enfatizando las ventajas y consideraciones que respaldan la utilización de dicha estrategia. Para su implementación se utilizará el módulo F2py, cuyo propósito es generar de manera efectiva la interacción entre ambos entornos de programación, aprovechando la capacidad y funcionalidad que ofrece cada plataforma.

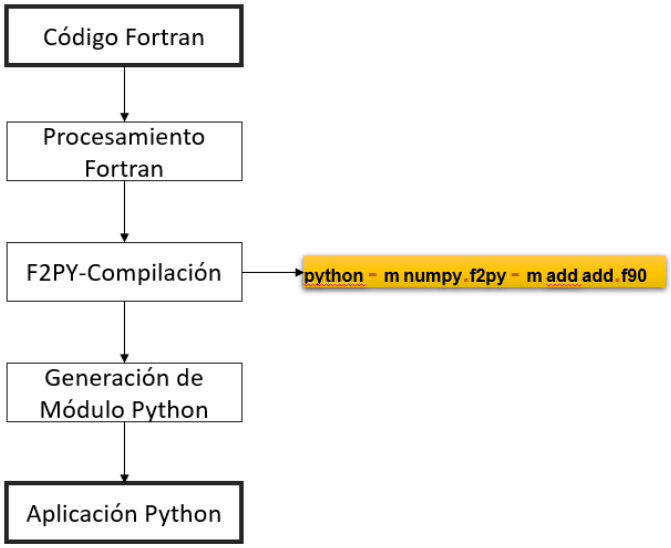

Fig. 6.1 – Comunicación Fortran-Python mediante F2py.

Bajo este contexto, el simulador sigue la estructura de la Fig. 6.1 esta proporciona una representación visual de la comunicación existente entre ambos lenguajes de programación. Principalmente se escriben dos códigos con enfoques distintos, pero al mismo tiempo complementarios, por una parte el código fuente es proporcionado por Fortran *(. f90)* en donde se realizan las funciones principales para el modelo del espesador. Python*(.py)* por su parte es principalmente utilizado para obtener una representación visual del diseño como también el desarrollo del controlador que busca mantener la estabilidad en las concentraciones del sistema. Por lo tanto, el capítulo proporciona, una visión detallada del diseño destacando las estrategias involucradas en la implementación.

6.2. Simulación de Espesadores

Optimizar el rendimiento de estos equipos puede ser un desafío debido a la complejidad y diseño. La variabilidad en la alimentación que entra al espesador, como también las condiciones de operación y la geometría misma del espesador de relave, puede suponer grandes problemas a la hora de mantener el proceso estable.

En este sentido, los modelos simulados ofrecen una herramienta valiosa, permitiendo comprender y mejorar el rendimiento de los espesadores de relave. El modelo puede simular el comportamiento del espesador bajo diferentes escenarios, proporcionando una información detallada sobre la distribución de sólidos y líquidos, la eficiencia de la sedimentación y otros parámetros relevantes.

Al incorporar modelos simulados en el proceso, se puede prever el rendimiento del espesador de relave, bajo diferentes condiciones lo que permite evaluar el impacto e identificar posibles mejoras. Estos procesos simulados no solamente permiten optimizar el rendimiento, sino que también permite maximizar la eficiencia y rentabilidad de los espesadores de relaves, a su vez también permite minimizar los riesgos en el manejó de relaves mineros.

6.2.1 Condición estabilidad para métodos numéricos CFL (Número de Courant-Friedrichs-Lewy)

La condición CFL, aplicada a métodos numéricos permite resolver ecuaciones derivadas parciales, es esencial en simulaciones donde los procesos dinámicos consideran largas constantes de tiempo como la sedimentación continua u fenómeno modelado matemáticamente. La adecuada implementación asegura la estabilidad y convergencia de los métodos numéricos utilizados en intervalos de tiempo $[0, t]$ aplicado a simulaciones, lo que resulta primordial para obtener resultados precisos y confiables.

Esta condición se utiliza específicamente en simulaciones matemáticas aplicadas a la sedimentación continua 2.4.1, donde se modelan las variaciones en las concentraciones de sólido a lo largo del tiempo. La condición CFL, actúa como una restricción fundamental para realización y análisis condicionada largas constantes de tiempo, permitiendo simular el comportamiento de las partículas sólidas en un espesador de relave en el transcurso del tiempo. Es por ello por lo que para asegurar una correcta convergencia en un determinado intervalo de tiempo debe implementarse una correcta condición CFL.

6.2.2 Incorporación desde Fortran

Para el simulador dinámico, se requiere discretizar el área de estudio en segmentos igualmente espaciados a lo largo del eje x. Para ello es esencial determinar la precisión y la resolución del simulador dinámico, lo que se refleja en (6.1).

$$\Delta_X = \frac{x_r - x_l + 3.0}{Cantidad\ de\ puntos\ requeridos} \tag{6.1}$$

donde,

x_r : Distancia desde el punto de alimentación hasta el punto de descarga del espesador.

x_l : Distancia desde el punto de alimentación hasta el punto desbordamiento del espesador.

Δ_X : Tamaño de cada segmento en el eje X.

Esta ecuación permite establecer el cálculo del intervalo entre cada punto discretizado en eje x. Como complemento, el simulador dinámico se debe acotar bajo un intervalo de tiempo en el eje temporal. El intervalo de tiempo utilizado se define como Δ_t y calcula en función del tamaño de discretización presentado en (6.1).

Dicho esto, el cálculo discretizado en el eje del tiempo viene condicionado por la restricción *CFL*, particularmente esta condición es utilizada en simulaciones para ecuaciones diferenciales parciales (EDP) y se detalla en el apartado (3.3.6), para efectos prácticos de simulación la condición de estabilidad se establece en $CFL = 500$, ya que impone una restricción de acuerdo al tamaño del paso en el tiempo en relación con el tamaño espacial mencionado en (6.1). Esta condición es fundamental ya que permite mantener una estabilidad numérica del sistema.

Es por ello que la relación entre el tamaño del paso de tiempo y el tamaño espacial se define mediante (6.2).

$$\Delta_t = 500 \cdot \Delta_x \cdot 2.0 \tag{6.2}$$

Esta relación garantiza y sienta las primeras bases para que el simulador dinámico funcione dentro de los límites de estabilidad, siempre y cuando se mantenga dentro de los rangos aceptables. A su vez la condición *CFL* viene condicionado por el coeficiente de Courant (λ), ya que establece la cantidad de pasos que se realizarán en un intervalo de tiempo dado, como se describe en (6.3).

$$\lambda = \frac{\Delta_t}{\Delta_x} \tag{6.3}$$

donde,

Δ_t : Pasos en el tiempo.

λ : Coeficiente de Courtant.

Bajo estas premisas el coeficiente Courant (λ) permite ajustar el tamaño del paso de tiempo en función del tamaño espacial, garantizando la estabilidad numérica del simulador dinámico.

El comportamiento del fluido contenido en el espesador se ve directamente influenciado por la viscosidad del mismo. Es por ello que para determinar la viscosidad del fluido (μ) se determina mediante (6.4).

$$\mu = \frac{\lambda}{\Delta_x} \tag{6.4}$$

Para el desarrollo del simulador requiere determinar las condiciones de diseño del espesador que se aplicaran. Estas condiciones incluyen determinan la distribución inicial respecto a la altura con la que se está diseñando el espesador, así como los factores de corrección que puedan aplicar a cada zona del sistema.

Para las condiciones iniciales, con la que se diseña la altura del espesador. El espesador debe tener una distribución equidistante en espacios igualmente segmentados tomando como condición inicial el espaciamiento dado en $x_l - 1.0$ y avanzando en incrementos de Δ_x. Además, se establece un factor de corrección (F_c) cuando el valor considerando en el eje x supera el valor de 4.0. Esta condición permite calcular el área transversal para el diseño del espesador.

Cabe mencionar que la distribución inicial en cuanto a la altura del espesador y las correcciones aplicadas afectan directamente la evolución temporal del sistema en lo que respecta a la sedimentación del contenido sólido y la formación de capas en el fondo del espesador.

Luego de describir los parámetros de diseño del espesador, se debe resolver el problema de las variables de entrada y salida del sistema. El cálculo es clave ya que permite describir y comprender las dinámicas internas del espesador de relave y cómo el flujo de entrada afecta en el asentamiento del contenido sólido en el fondo del espesador. La Fig. 2.4 representa la distribución por zonas y el diseño en cuanto a la estructura del espesador, reflejando puntos clave en el caudal de alimentación (Q_F), la caudal descarga (Q_R) y desbordamiento (Q_L). Para el entendimiento en cuanto a los flujos dentro del sistema, es necesario considerar la entrada y salida del sistema (Q_F, Q_R), con ello se puede tener una relación directamente en cuando al flujo desbordamiento del sistema (Q_L).

Mediante la bandeja de alimentación que inyecta una mezcla preveniente de etapas previas del ciclo minero y que contiene un porcentaje sólido y líquido. Como parámetros de entrada al sistema la alimentación que ingresa al espesador se ve representa por la velocidad a la que los sólidos ingresan al sistema por unidad de área respecto a la sección transversal del espesador. El caudal de entrada se define en (m^3/hr) sin embargo para normalizar el sistema se trabaja (m/s), que indica la cantidad de solidos por unidad de tiempo. Dicha taza de alimentación se normaliza considerando el área transversal del espesador de relaves y se expresa mediante (6.5).

$$q_F = \frac{Q_F}{\pi \cdot (d \cdot 2.0) \cdot 0{,}25 \cdot 3600.0} \tag{6.5}$$

donde,

q_F : Caudal de sólidos en la entrada al espesador, en (m/s).

d : Diámetro del espesador.

Q_F : Caudal de sólidos en la entrada al espesador, en (m^3/hr).

De acuerdo al apartado (6.5) y siguiendo la misma lógica utilizada para modelar el caudal de entrada. Esta normalización en cuanto a velocidad de salida que influye directamente en la eficiencia del proceso de sedimentación y la concentración final de sólidos en el efluente. Por lo que al igual que el caudal de alimentación, se debe normalizar el caudal de salida, esto proporciona una medida estandarizada en su conjunto para en análisis del sistema y dado por (6.6).

$$q_R = \frac{Q_R}{\pi \cdot (d \cdot 2.0) \cdot 0{,}25 \cdot 3600.0} \tag{6.6}$$

donde,

q_R : Caudal de sólidos en la salida del espesador, en (m/s).

d : Diámetro del espesador.

Q_R : Caudal de sólidos en la salida del espesador, en (m^3/hr).

Por consiguiente, para evaluar la recuperación de agua, representada por el flujo desbordamiento en las zonas periféricas del espesador. Se define como la diferencia entre el caudal de alimentación (q_F) y el caudal de salida (q_R). La evaluación de esta recuperación es primordial y es de suma importancia ya que permite comprender el rendimiento y eficiencia en el proceso de sedimentación. Por consiguiente, retomando (6.5) y (6.6) se calcula el flujo de desbordamiento dado por (6.7).

$$q_L = q_R - q_F \tag{6.7}$$

donde,

q_L : Caudal de rebalse del espesador, en (m/s).

Una vez establecidas las variables de entrada y salida del sistema, se debe realizar el análisis del comportamiento interior del espesador. Esto implica evaluar el desempeño en función de los parámetros de entrada y salida.

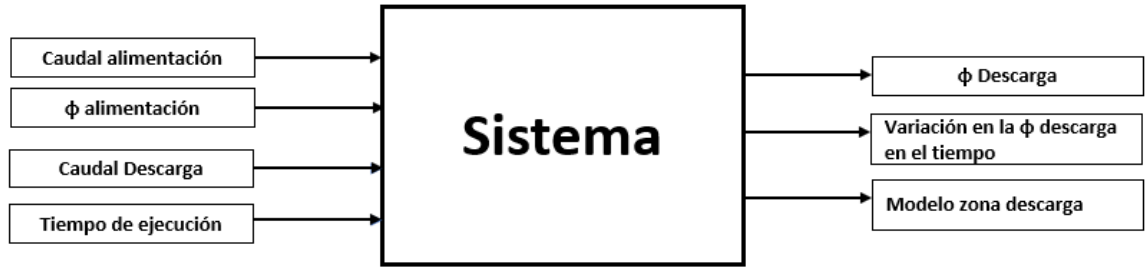

Fig. 6.2 – Modelación simplificada entrada-salida del sistema.

Una vez identificadas las variables que actúan activamente en el proceso de sedimentación, es necesario simular el comportamiento dinámico del espesador. El sistema esta principalmente diseñado para el tratamiento de minerales con tal de mantener estable la concentración de descarga (ϕ_R). El simulador se enfoca en cómo cambia la concentración de sólidos en la descarga (ϕ_R) a lo largo del tiempo, observando el comportamiento para las diferentes zonas del espesador.

Para el modelo del espesador, la función "$flux$" desempeña un rol fundamental, ya que permite calcular la tasa de flujo de sólidos entre dos segmentos adyacentes. Esto es relevante ya que permite comprender el comportamiento del contenido sólido través de las diferentes zonas del equipo. Los segmentos adyacentes vienen dados por secciones contiguas como puntos discretos, lo que facilita la evaluar la concentración de sólidos en el interior del sistema y como esta distribución evoluciona en el tiempo.

De acuerdo (6.8) permite modelar cómo se desplaza el contenido sólido de un segmento a otro, considerando múltiples factores, como lo son el caudal de alimentación (q_F) y caudal de descarga (q_R), así como el tamaño de cada segmento en el espacio, determinado en la (6.8).

$$aux_1 = flux\left(\phi(i), \phi(i+1), x(i) + \frac{\Delta_x}{2.0}, q_R, q_L\right) - flux\left(\phi(i-1), \phi(i), x(i) - \frac{\Delta_x}{2.0}, q_R, q_L\right) \quad 1 < i < pasos_x \tag{6.8}$$

El cálculo de la variable "aux_1" mediante la condición "$flux$" realiza cálculos para cada segmento del espesador para un determinado tiempo de muestreo. Aquí la concentración de sólidos en el tiempo viene dada por $\phi(i)$ para un determinado tiempo de muestreo, mientras que $x(i)$ representa la altura del espesador. Por lo tanto, la función "$flux$" evalúa el comportamiento de las concentraciones de sólidos entre los segmentos que vienen dado por $(i, i+1, i-1)$.

A su vez la contribución en la propiedad del fluido entre segmentos adyacentes, de acuerdo al cambio de concentración de sólidos en el espesador, viene representado por la (6.9). Dicha contribución se calcula mediante la diferencia en las propiedades del fluido en cada segmento por la diferencia en la concentración de sólidos entre ellos. Esta aplicación evalúa el impacto que tiene los cambios de propiedades del fluido que afectan en el transporte y distribución de sólidos suspendidos en el espesador. La ecuación que define esta contribución viene dada por (6.9).

$$aux_2 = \alpha_1(x(i) + \Delta_x \cdot 0{,}5) \cdot (A(\phi(i+1) - A(\phi(i)) - \alpha_1(x(i) - \Delta_x \cdot 0{,}5) \cdot (A\left(\phi(i) - A(\phi(i-1))\right) \;\; 1 < i < pasos_x \quad (6.9)$$

Donde "α_1" representa las propiedades del fluido en cada segmento, $x(i)$ corresponde a la altura del espesador y "Δ_x" corresponde al tamaño del segmento. Así, la ecuación (6.9) refleja los cambios en las propiedades de los fluidos y como impacta en la distribución de sólidos a largo del espesador.

6.2.3 Incorporación desde Python

La implementación en *Python* ($.py$) se convierte en una pieza clave en el estudio y análisis del espesador, ya que ofrece una interfaz interactiva en conjunto con herramientas de visualización. La estrategia programada involucra la interacción de ambas plataformas ya que mediante la integración de Python ($.py$) permite la creación de una biblioteca cuyo propósito permite ajustar los parámetros operativos del sistema en tiempo real y controlar la ejecución del espesador, además de proporcionar una visualización grafica de los resultados. Esta dinámica facilita la interacción y análisis del espesador bajo diferentes escenarios operativos, lo que conduce a mejoras en cuanto a la eficiencia y el rendimiento del sistema.

La implementación en *Python* ($.py$) se convierte en una pieza clave en el estudio y análisis del espesador, ya que ofrece una interfaz interactiva en conjunto con herramientas de visualización. La estrategia programada involucra la interacción de ambas plataformas mediante el módulo *"F2py"* ya que mediante la integración de Python ($.py$) permite la creación de una biblioteca cuyo propósito permite ajustar los parámetros operativos del sistema en tiempo real y controlar la ejecución del espesador.

En el contexto del lenguaje programado, *Python* $(.py)$ es una herramienta valiosa que permite realizar simulaciones detalladas mediante las bibliotecas *Numpy.np, Matplotlib.pyplot, mpl.toolkits. mplot3d.*

Para la integración del código *Fortran* $(.f90)$ se requiere llamar las funciones definidas en *Python* $(.py)$. Este proceso tiene como finalidad de generar la interacción fluida entre ambos lenguajes de programación una vez el módulo *F2py* ha realizado el vínculo. Para lograr este objetivo, desde *Python* $(.py)$, se debe llamar al nombre de la función desarrollada en *Fortran* $(.f90)$ mediante la definición "*def"*, cuyo propósito es esencial para definir las variables que se utilizaran como entrada y salida definidas como (In) (Out), ya que cualquier error en esta definición puede generar errores en el vínculo. Además, se debe incluir la declaración "*return"* para garantizar que el ciclo de ejecución se repita correctamente ante cualquier modificación en las variables con las que se diseña el sistema.

Una vez establecidos los parámetros, se definen como parte de la biblioteca las variables que influyen en el diseño del espesador y que podrán ser modificadas como lo son Q_F, Q_R, ϕ_F, ϕ_R . Además, para simular dinámicamente el comportamiento del espesador, se emplea el enfoque para un tiempo discreto, ya que es fundamental determinar la frecuencia con la que se repetirá el bucle principal del sistema.

Esta estrategia permite ajustar la resolución del modelo y evaluar el comportamiento de las concentraciones contenidas bajo diferentes escalas temporales.

Para el almacenamiento de los datos, se inicializa una matriz de resultados, con la cantidad de dimensiones declarados en *Python* $(.py)$.

$$Matriz\ resultados = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

Esta matriz es fundamental y se utiliza principalmente para almacenar los resultados de la simulación ya que se actualiza con cada iteración del bucle. Dicha matriz contiene las concentraciones de alimentación y descarga y como varían a lo largo de las distintas zonas del espesador permitiendo visualizar el comportamiento de acuerdo a la variación en el tiempo.

Para cada iteración, se llama a la función principal que contiene el modelo del espesador. Los resultados de la simulación se asignan a cada fila de la matriz, donde cada fila representa un paso en el tiempo y cada columna representa la sección del espesador. Además, desde Python se ajustan los caudales de entrada y salida para simular cambios en las condiciones operativas del sistema.

Antes de imprimir las concentraciones (ϕ) obtenidas en la matriz de resultados, se utiliza "$plt.pause(0.01)$" para detener la ejecución del programa. Este procedimiento permite que las visualizaciones generadas hasta el momento se muestren y almacenen correctamente en la interfaz gráfica, antes de continuar con la siguiente iteración del bucle.

6.3. Discusión

El capítulo presenta la metodología utilizada para el desarrollo del simulador de espesadores. Se utilizo Fortran (*f90*) para escribir el código fuente. Esto implica la creación de un bucle principal, que de acuerdo las condiciones y el análisis por zonas al interior del espesador Fig. 2.5. La vinculación entre Fortran y Python se logró gracias al módulo *F2py*, que corresponde a una extensión de Python.

Python fue adaptado para crear una biblioteca capaz de variar las condiciones del espesador, el cual fue adaptado con la finalidad de crear una biblioteca capaz de modificar sus variables de estado, como el caudal de alimentación (Q_F), el caudal de descarga (Q_R), la concentración de alimentación (ϕ_F) y la concentración de descarga (ϕ_R). Además, desde Python es posible ajustar la cantidad de pasos utilizados para la simulación.

Una de las dificultades detectadas durante la simulación fue definir el nivel de precisión de los softwares. Es necesario especificar la sensibilidad con la que se realiza la simulación, para ello se debe estandarizar en ambos softwares la misma precisión, definiendo los puntos flotantes en 32 o 64 BITS. Al trabajar con 64 BITS, tiene ciertas prestaciones que pueden beneficiar en el tratamiento de datos, ya que se puede una mayor precisión en los cálculos numéricos, puede conducir a divergencias debido al redondeo de datos muy pequeños.

El sistema manejar grandes volúmenes de datos sin redondeos al utilizar una precisión en 32 BITS. Aunque es posible mezclar niveles de precisión, es crucial definir claramente si se está trabajando con 32 o 64 BITS, lo cual tienen que ser idénticos en ambos programas Fortran y Python.

Capítulo 7. Casos prácticos del simulador

7.1. Introducción

El capítulo tiene presenta los resultados prácticos obtenidos desde el simulador. En este contexto, se explora en profundidad el comportamiento dinámico del sistema bajo variaciones de condiciones operativas. Además, se presentará un estudio en los tiempos de respuesta del sistema para los diversos escenarios. Se analizará la velocidad con la que el espesador responde a cambios en la condiciones operativas o ajuste en el controlador para la concentración de sólidos deseada. Este análisis permitirá determinar la presencia de oscilaciones o inestabilidades en el proceso y se identificaran posibles factores que puedan afectar en la estabilidad. Para este análisis se expondrá al simulador bajo dos condiciones.

(i) Estabilidad del espesador bajo distintas condiciones sin controlador

En primera instancia, se pone a prueba la estabilidad del espesador de relave en ausencia del controlador. Se realizan simulaciones bajo diversas condiciones de alimentación *(Q_F)*. Esta aplicación permite analizar cómo varía la respuesta del espesador de relave frente a cambios en el flujo de alimentación, también se obtiene un análisis en los tiempos de respuesta que tiene el espesador.

(ii) Estabilidad del espesador bajo distintas condiciones de S.P con controlador.

En esta segunda etapa del análisis, se explora en la estabilidad del espesador de relave bajo la presencia de un controlador *PI,* considerando distintos puntos de ajuste determinado por las variaciones en el *S.P* y variaciones en la concentración de alimentación *(ϕ_F).* Se llevan cabo diversas simulaciones para evaluar cómo responde el sistema cuando se ajusta el *S.P* aplicado a la concentración de sólidos deseada. Este enfoque permite realizar un análisis primordial, para obtener una eficacia del controlador y mantener la estabilidad del proceso, con tal de cumplir con la calidad de la simulación.

7.2. Resultados simulación

A continuación, se presentarán los resultados de la simulación, considerando dos condiciones globales. En primer lugar, se llevará a cabo un estudio sin intervención de un controlador, con variaciones en los caudales de alimentación del sistema (Q_F). El objetivo es obtener respuestas que indiquen la estabilidad del simulador y validar la correcta vinculación proporcionada por el módulo *F2py*.

El segundo análisis, se implementará un controlador proporcional-integral (PI). El enfoque principal es obtener una respuesta que permita mantener las concentraciones de descarga (ϕ_R) dentro de los límites determinados por el *S.P*. Se busca, generar variaciones en el *S.P* con tal de que el espesador sea capaz de estabilizarse frente a variaciones del sistema. Para validar el comportamiento del simulador, se expondrán cambios en el *S.P*, caudal de alimentación (Q_F) y concentraciones de alimentación (ϕ_F).

7.2.1 Caso 1: Caudal Alimentación(Q_F)

Primeramente, se presenta el caso para las condiciones nominales del espesador de relave, este análisis pretende observar el comportamiento en función de sus valores nominales, que caracterizan las condiciones sistema e identificar si la conexión se realiza exitosamente y si tiene problemas de estabilidad.

La Tabla 7.1 muestra el diagrama de flujo que representa el proceso utilizado para el diseño del simulador. Los parámetros de entrada se definieron mediante una biblioteca en Python, como se detalla en la Tabla 7.1. Una vez establecido el vínculo entre ambos lenguajes de programación se determinan los parámetros de entrada al sistema como también la cantidad de pasos que se ejecutara el proceso, esto es definido mediante Python (*.py*). Durante la ejecución del ciclo, los valores de concentraciones objetivos en cada paso se almacenan en una matriz previamente vacía, acumulando así los resultados según sea necesario.

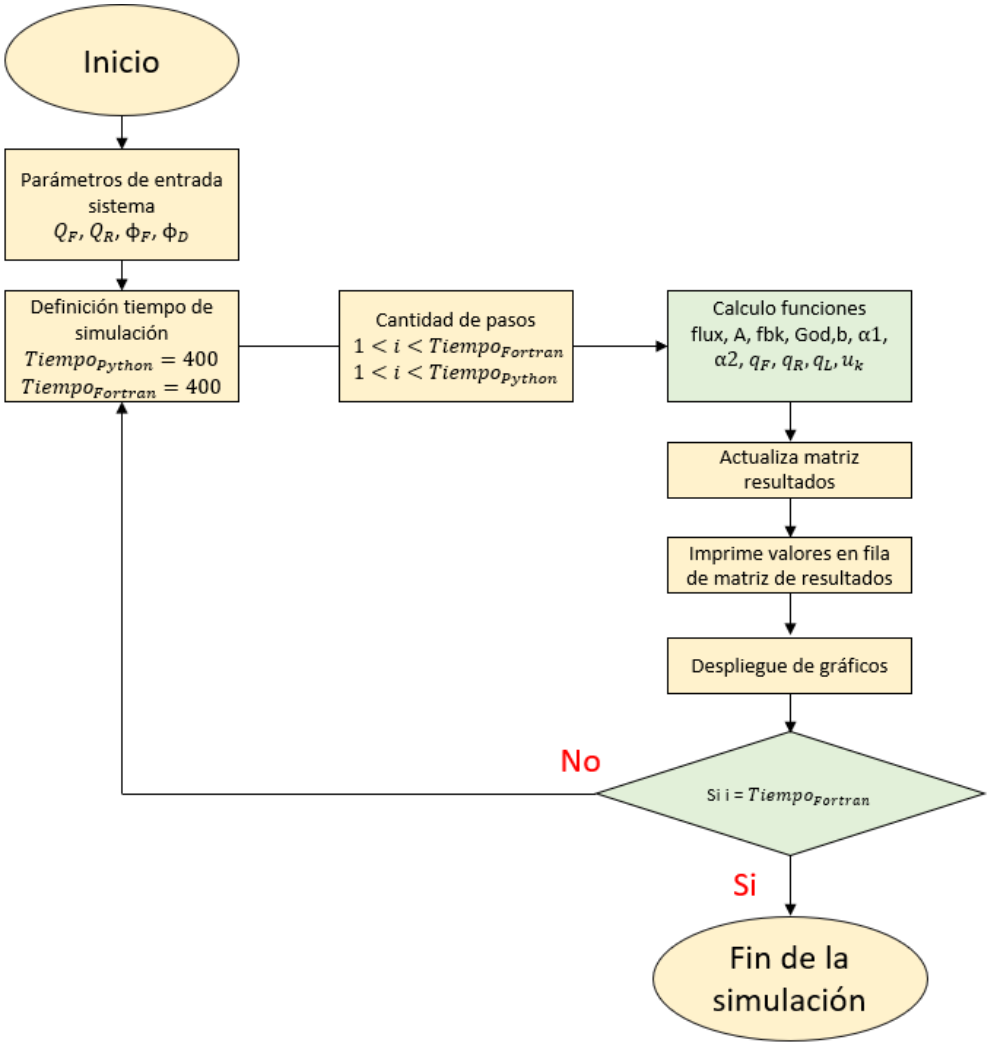

Fig. 7.1 – Simulador para espesadores de relave - sin controlador.
(a) Sistema modelado en 400 pasos.

De acuerdo a los datos presentados en la Tabla 7.1 el color verde presenta la aplicación de Fortran y a su vez el color amarillo la aplicación Python. Se mencionan los parámetros nominales del proceso que se utilizaron en la simulación sin condiciones de cambio. Estos parámetros son permiten comprender el comportamiento del sistema durante el proceso de espesamiento. La Tabla 7.1. incluye los caudales entrada y salida del sistema, así como también las concentraciones correspondientes.

Tabla 7.1. Parámetros nominales del sistema evaluado en 400 pasos

Parámetros	Simbología	Unidad	Valor
Flujo alimentación	Q_F	m^3/hr	1600.13
Flujo descarga	Q_R	m^3/hr	1177.00
Concentración alimentación	ϕ_F	-	0,304
Concentración descarga	ϕ_R	-	0,413

Del proceso de espesamiento se definen las condiciones iniciales y los parámetros de operación del sistema. El caudal de alimentación (Q_F) indica la cantidad de liquido que entra al espesador en un periodo de tiempo, mientras que la concentración de alimentación (ϕ_F), estos parámetros en su conjunto determinan la composición de la pulpa que entra en el espesador de relave.

A continuación, Fig. 7.2 el caso para los valores nominales del sistema y el comportamiento del espesador. El primer caso el espesador está trabajando en condiciones nominales sin variación simulado para *t = 400 [s]*.

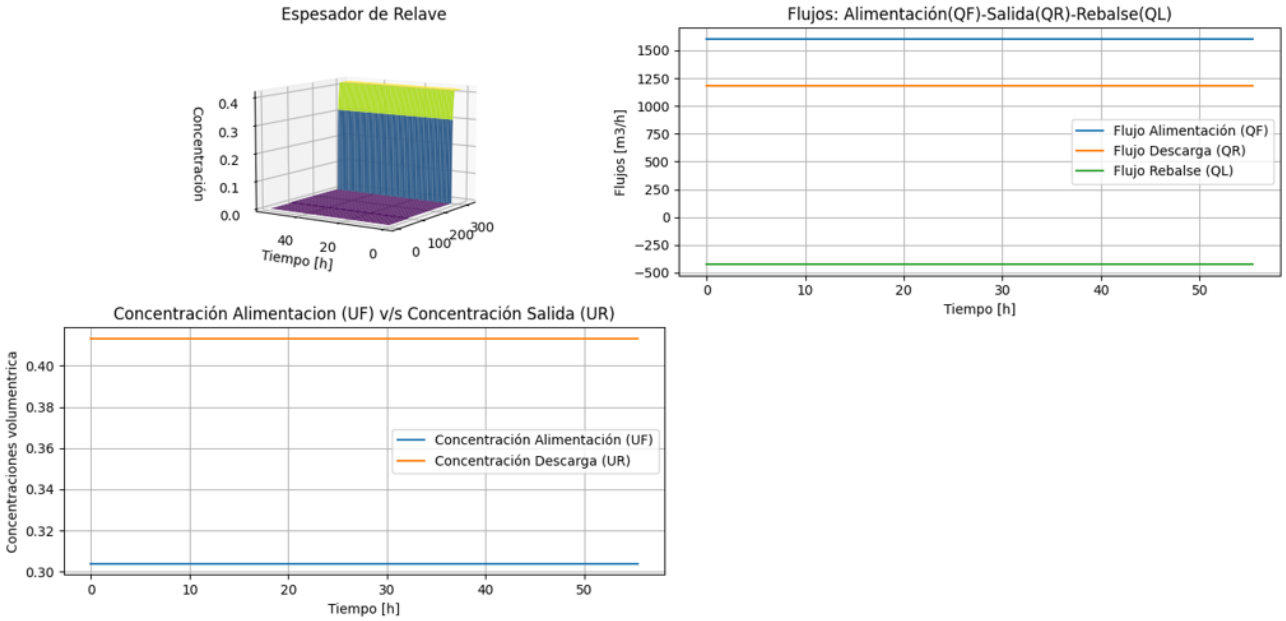

Fig. 7.2 – Simulador para espesadores de relave - sin controlador.
(a) Sistema modelado en 400 pasos (b) Flujo de alimentación (Q_F), Flujo de descarga (Q_R), Flujo de rebalse (Q_L) (c) Concentraciones de alimentación (ϕ_F), Concentraciones de descarga (ϕ_R).

Según se muestra en le Fig. 7.2, se puede deducir la duración de la simulación realizada. Se observa que el simulador pudo mantenerse estable con una comunicación efectiva y fluida entre la interfaz Fortran (*.f90*) y Python (*.py*), dado que el caudal de alimentación como también el caudal de descarga no sufran variaciones en el tiempo por lo que el espesador se considera que trabaja en un estado de equilibrio masico. Esto implica que el simulador fue capaz de visualizar modelar 50 [hrs] de funcionamiento del espesador de manera efectiva.

7.2.2 Caso 2: Caudal Alimentación(Q_F) al 80% del valor nominal

El diagrama de flujo Fig. 7.3. representa el caso para variaciones en el caudal de alimentación (Q_F) correspondiente a la porción del 80% de los valores nominales del sistema. A diferencia del Caso 1, se agrega un salto de escalón que permite variar el caudal de alimentación (Q_F), que permite someter al simulador a variaciones.

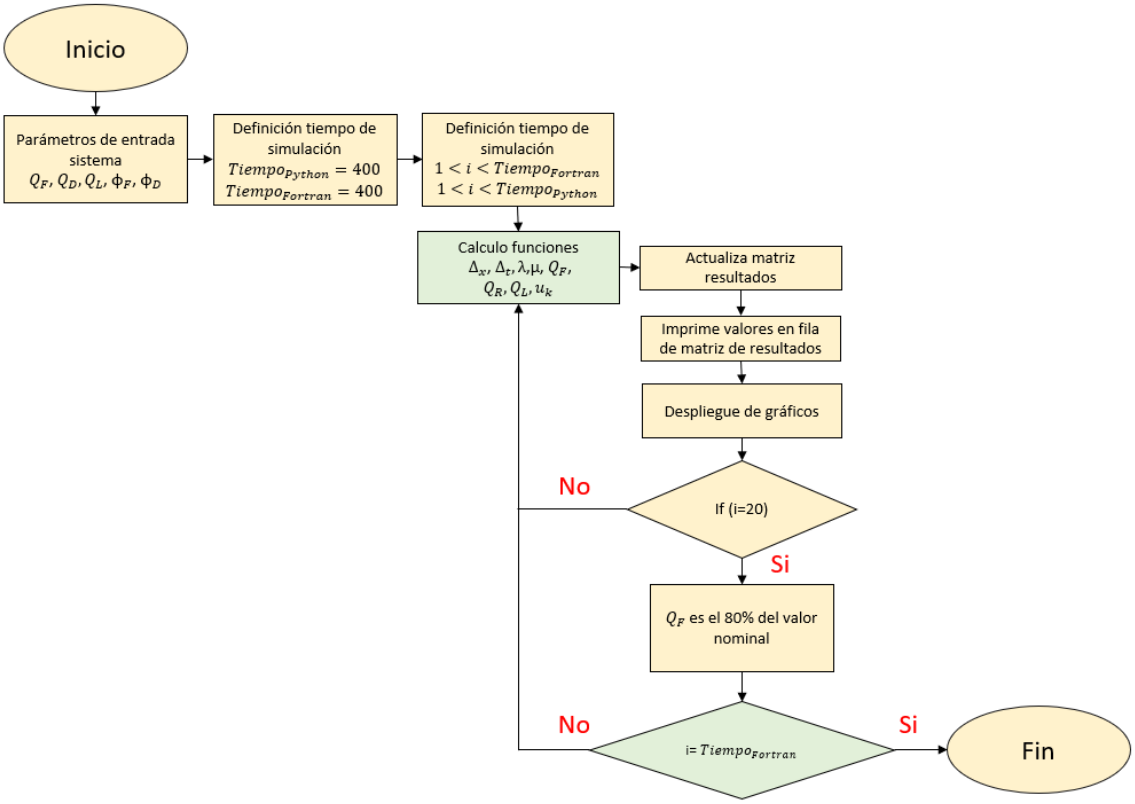

Fig. 7.3 – Simulador para espesadores de relave - sin controlador.
(a) Sistema modelado en 400 pasos (b) Sistema escalón 80% del valor nominal.

Para Caso 2 el espesador está trabajando con variaciones del 80% del caudal de alimentación (Q_F) en el tiempo simulado en 400 pasos de Fortran.

Tabla 7.2. Parámetros al 80% del valor nominal simulado en 400 pasos Fortran.

Parámetros	Simbología	Unidad	Valor
Flujo alimentación	Q_F	m^3/hr	1280.10
Flujo descarga	Q_R	m^3/hr	941,6
Concentración alimentación	ϕ_F	-	0,304
Concentración descarga	ϕ_R	-	0,413

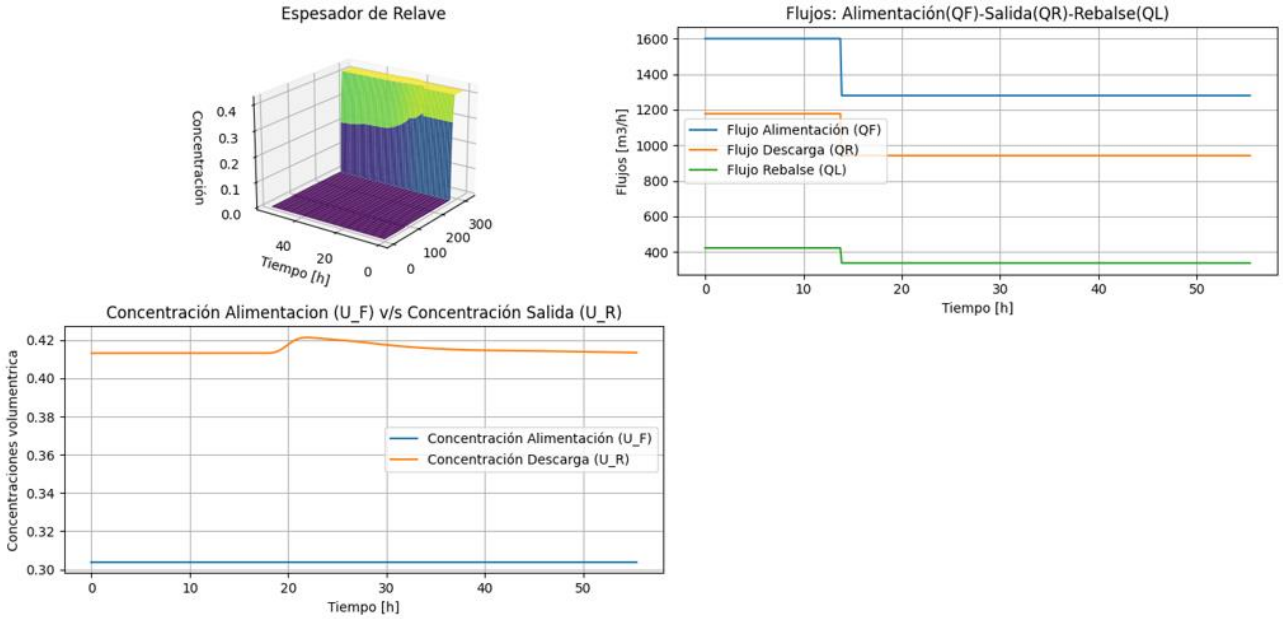

Fig. 7.4 – Simulador para espesadores de relave - sin controlador.
(a) Sistema modelado en 400 pasos (b) Sistema escalón 80% del valor nominal (c) Flujo de alimentación (Q_F), Flujo de descarga (Q_R), Flujo de rebalse (Q_L) (d) Concentraciones de alimentación (ϕ_F) Concentraciones de descarga (ϕ_R).

La Fig. 7.3, induce la duración de la simulación realizada. Se observa que el simulador es capaz de mantenerse estable, a pesar del cambio de escalón ocurrido a las 15 [hr] del proceso simulado. Además, según se observa en la Fig.(c), se aprecia que, a pesar de los cambios de escalón generados, el simulador experimenta un aumento en la concentración de alimentación entre las 20 y 30 [hr]. Sin embargo, el espesador logra estabilizar el nivel de concentraciones hasta alcanzar un estado estacionario. Por otro lado, debido a l cambio de escalón, disminuye el flujo de alimentación (Q_F) y, por consiguiente, de acuerdo con el balance de masas, el flujo de descarga (Q_R) también tiende a disminuir. A pesar de estos cambios, el espesador se encuentra operando en condiciones óptimas, ya que se observa un desbordamiento positivo en todo momento de la simulación.

7.2.3 Caso 3: Caudal Alimentación(Q_F) al 60% del valor nominal

Al igual que el caso 2, se realiza un nuevo cambio de escalón al 60% de los valores nominales del espesador.

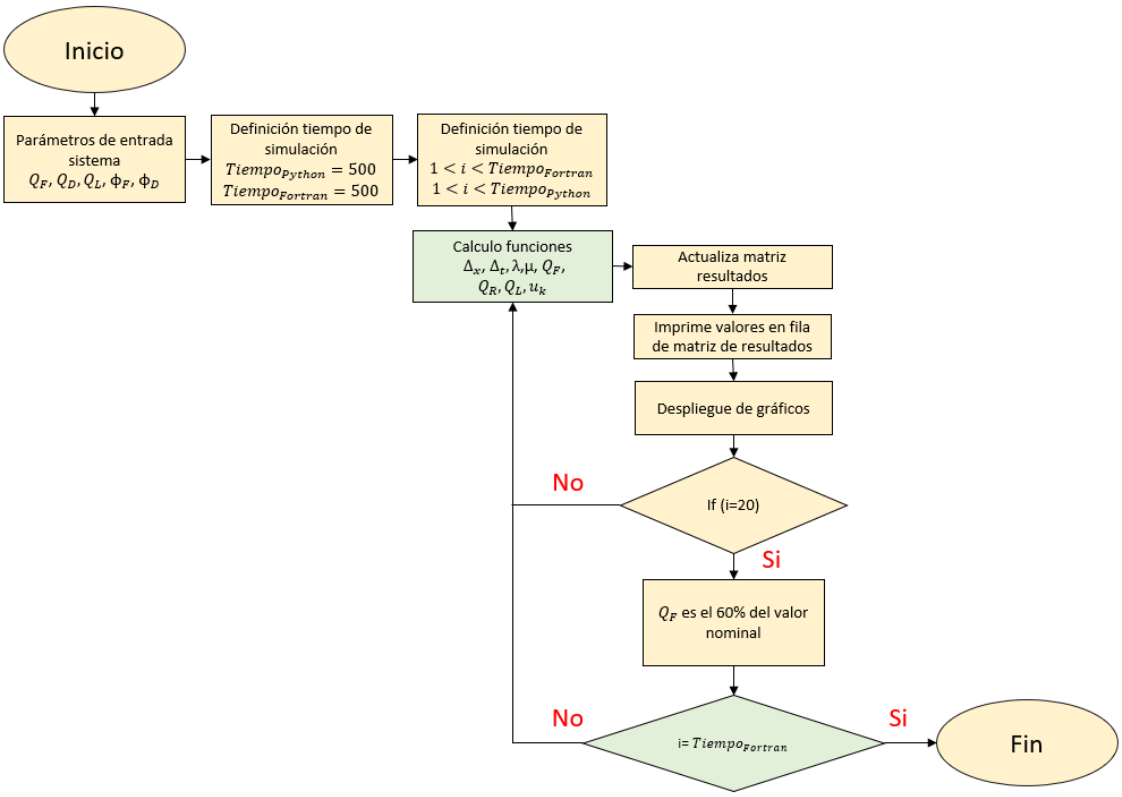

Fig. 7.5 – Simulador para espesadores de relave - sin Controlador.
(a) Sistema modelado en 500 pasos (b) Sistema escalón 60% del valor nominal.

Para caso 3, el espesador opera con variaciones del 60% en el caudal de alimentación (Q_F). Estas variaciones se producen en paso 20 de la simulación, que comprende una cantidad total de 500 pasos de Fortran equivalente a una evaluación de 85 [hrs] del proceso.

Tabla 7.3. Parámetros al 60% del valor nominal simulado en 500 pasos Fortran.

Parámetros	Simbología	Unidad	Valor
Flujo alimentación	Q_F	m^3/hr	960,078
Flujo descarga	Q_R	m^3/hr	706,2
Concentración alimentación	ϕ_F	-	0,304
Concentración descarga	ϕ_R	-	0,413

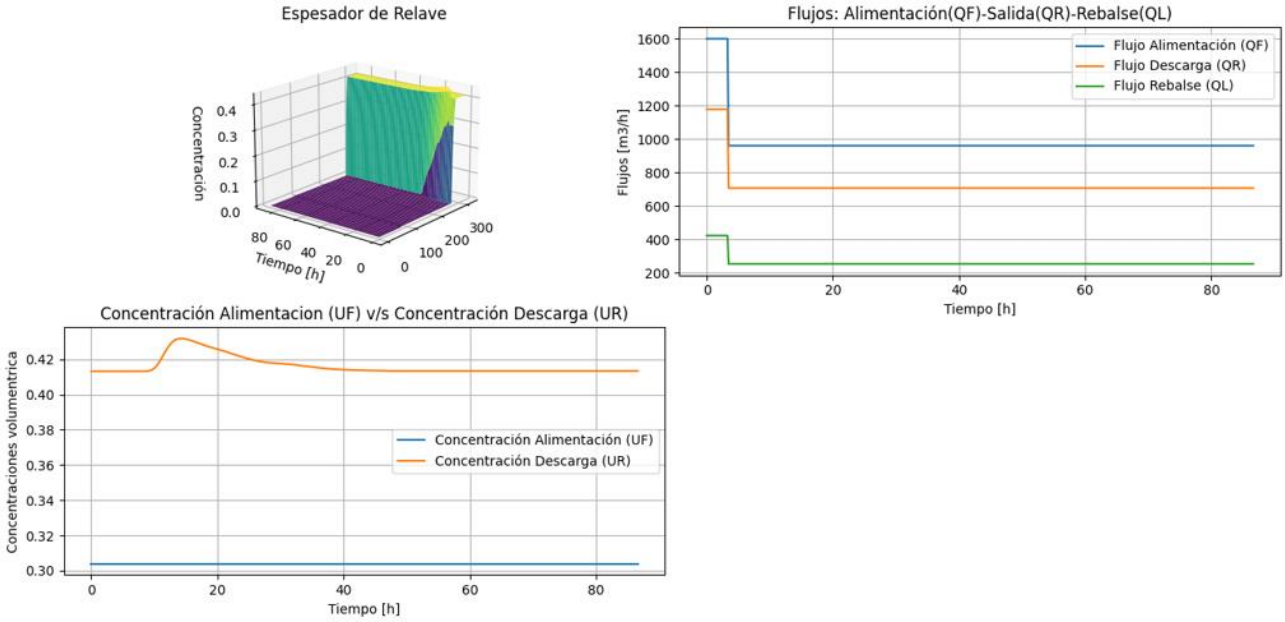

Fig. 7.6 – Simulador para espesadores de relave - sin controlador.
(a) Sistema modelado en 500 pasos (b) Sistema escalón 60% del valor nominal (c) Flujo de alimentación (Q_F), Flujo de descarga (Q_R), Flujo de rebalse (Q_L) (d) Concentraciones de alimentación (ϕ_F) Concentraciones de descarga (ϕ_R).

Similar al caso 2 mencionado, en el caso 3 no se observan variaciones en la estabilidad del espesador. Se mantiene el equilibrio de masas, lo que resulta en un desbordamiento continuo dentro del espesador. Estas condiciones reflejan una operación estable y consistente del sistema, incluso antes las variaciones del 60% introducida en el paso 20 de la simulación.

7.2.4 Caso 4: Caudal alimentación(Q_F) desde un 20% al 40% del valor nominal

El diagrama de flujo Fig. 7.7, que representa el escenario para variaciones en el caudal de alimentación (Q_F). Inicialmente, comienza en los valores nominales del sistema para luego reducir su alimentación al 40% y 20% posteriormente.

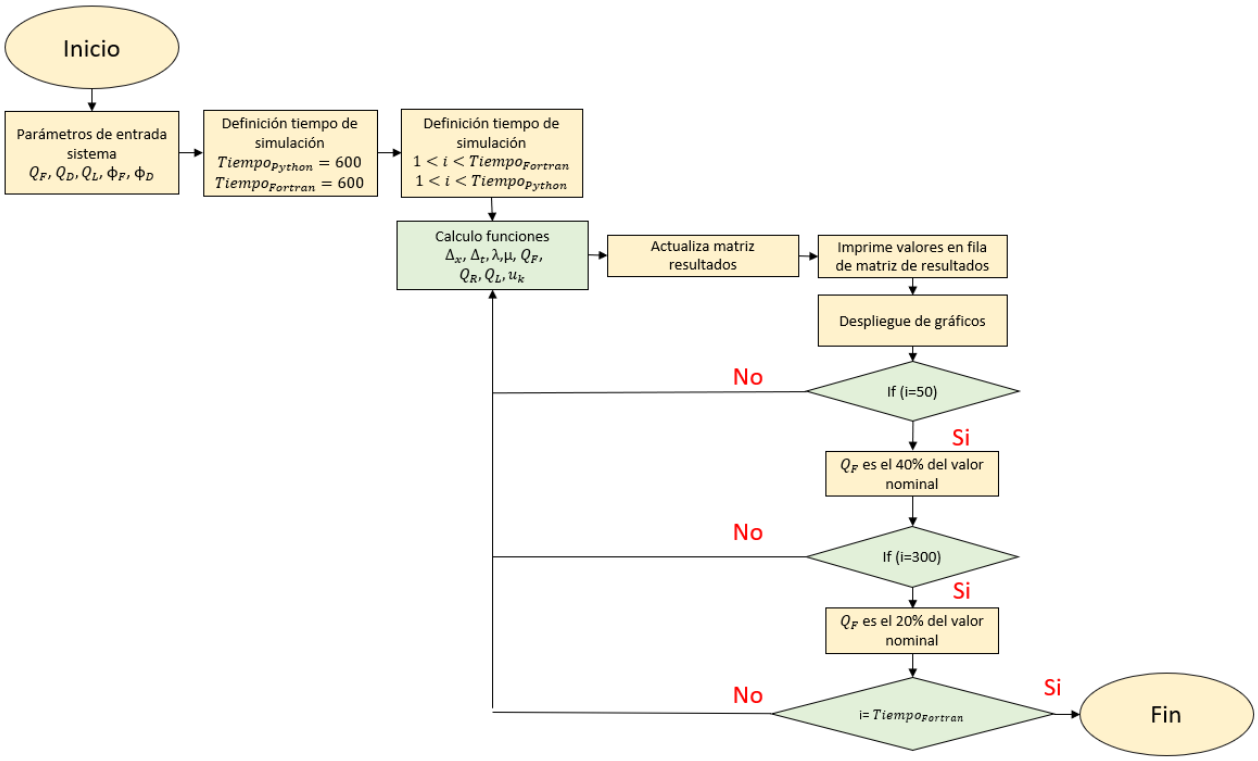

Fig. 7.7 – Simulador para espesadores de relave - sin Controlador.
(a) Sistema modelado en 600 pasos (b) Sistema escalón 40% al 20% del valor nominal.

En el caso 4, el espesador opera inicialmente con un caudal de alimentación del valor nominal indicado en la Tabla 7.1, para un momento de t = 0 [s]. Luego, se produce un aumento de escalón en caudal de alimentación (Q_F) al 40% en 50 pasos de Fortran. Finalmente, se aplica otra disminución en el flujo de alimentación del 20% del flujo alimentación nominal (Q_F) cuando se cumplen *300* pasos de Fortran.

Tabla 7.4. Parámetros con variaciones escalón desde hasta 40% y 20% del valor nominal.

Parámetros	**Simbología**	**Unidad**	**Valor 20%**	**Valor 40%**
Flujo alimentación	Q_F	m^3/hr	320,025	640,052
Flujo descarga	Q_R	m^3/hr	235,4	470,8
Concentración alimentación	ϕ_F	-	0,304	0,304
Concentración descarga	ϕ_R	-	0,413	0,413

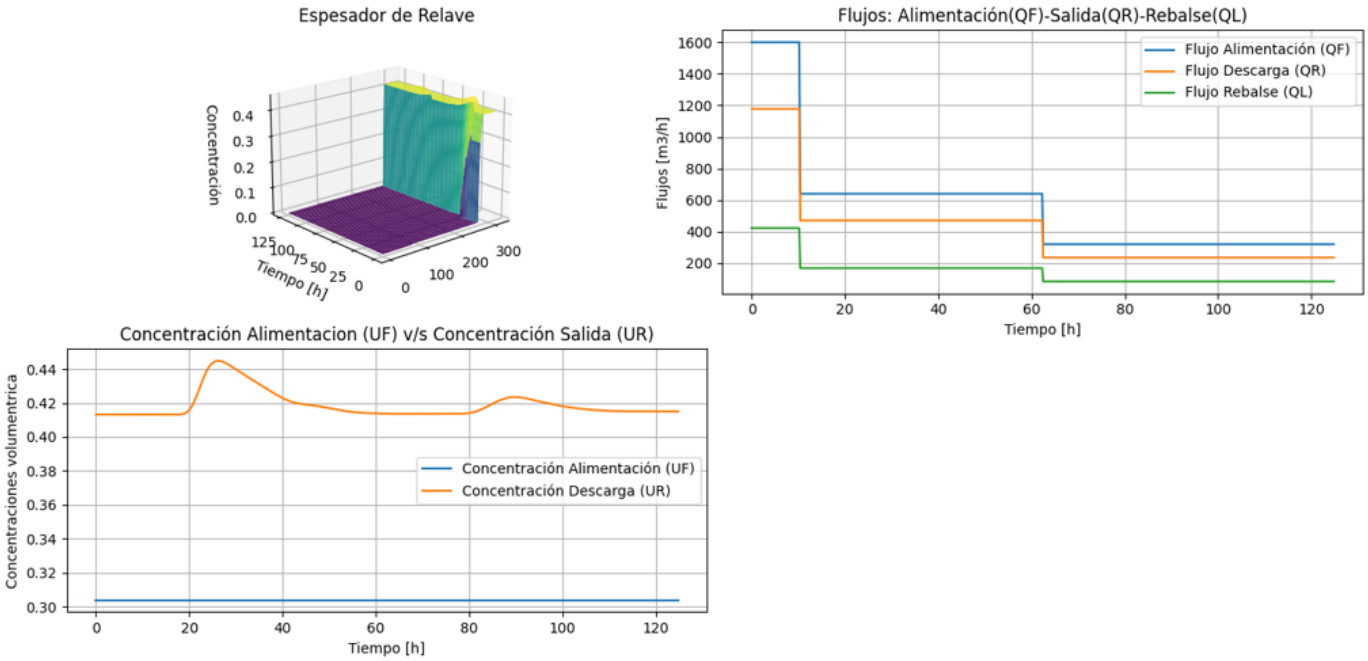

Fig. 7.8 – Simulador para espesadores de relave - sin controlador.
(a) Sistema modelado en 600 pasos (b) Sistema escalón 40% al 20% del valor nominal (c) Flujo de alimentación (Q_F), Flujo de descarga (Q_R), Flujo de rebalse (Q_L) (d) Concentraciones de alimentación (ϕ_F) Concentraciones de descarga (ϕ_R).

En el caso 4, el espesador experimenta dos cambios en los parámetros del flujo de alimentación $(\boldsymbol{Q_F})$. Inicialmente, parte del valor nominal y luego se produce el primer cambio de escalón después de 50 pasos, equivalente a 50 [hr] de simulación del proceso del espesador en Fortran. Durante este período, se observa un incremento en las concentraciones, seguido de una estabilización hasta alcanzar un estado estacionario. A pesar de los cambios en los escalones, se mantienen el equilibrio de masas, lo que resulta en un rebalse constante en las zonas periféricas del espesador.

7.2.5 Caso 5: Variaciones en el Setpoint 0,41 al 0,42

Para el caso 5, se evalúa un controlador (PI) diseñado con el objetivo de analizar su respuesta ante variaciones en el S.P y su capacidad para mantener la estabilidad de sistema para una señal de control (Q_R).

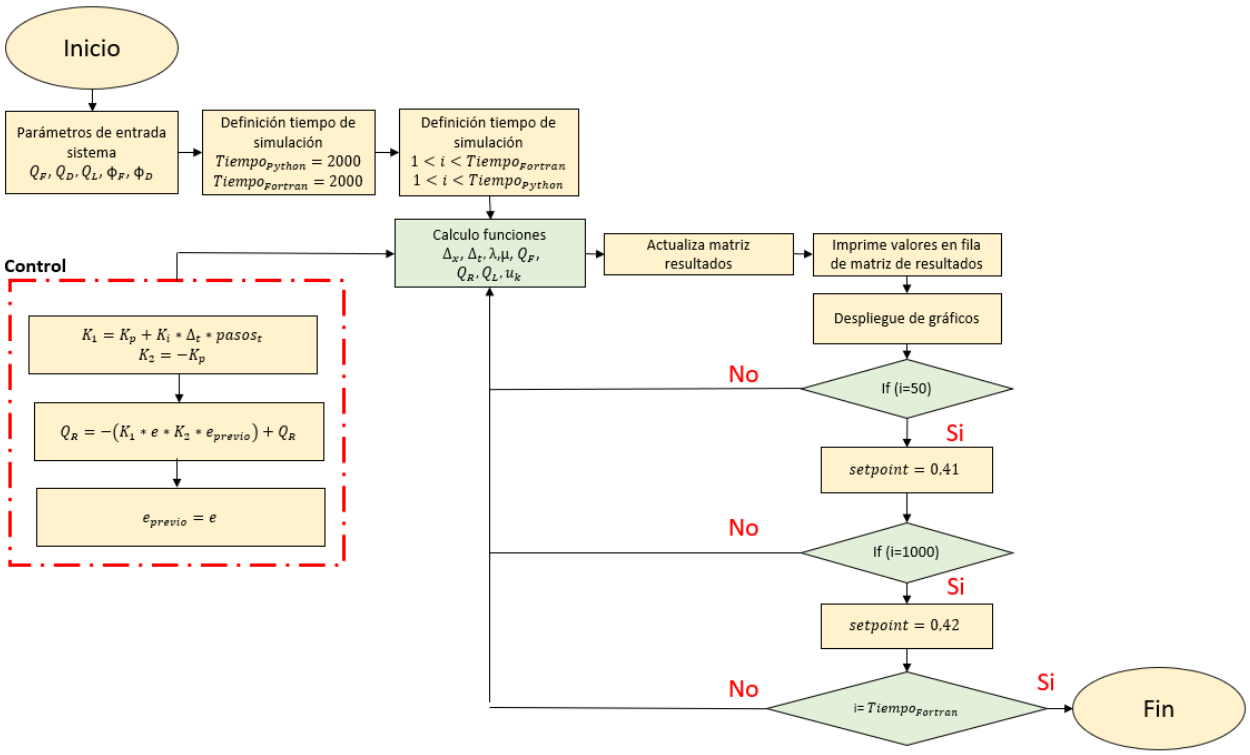

Fig. 7.9 – Simulador para espesadores de relave - con controlador.
(a) Sistema modelado en 2000 pasos (b) Setpoint 0,41 al 0,42 del valor nominal.

Tabla 7.5. Parámetros con variaciones en el SetPoint (S.P) en 0,41.

Parámetros	Simbología		Unidad	Valor 0,41	Valor 0,42
Flujo alimentación	Q_F		m^3/hr	1600,13	1600,13
Flujo descarga	Q_R		m^3/hr	1177	1177
Concentración alimentación	ϕ_F	-		0,304	0,304
Concentración descarga	ϕ_R	-		0,4100	0,4200
Constante Proporcional	k_p	-		10	10
Constante integrative	k_i	-		0,1	0,1

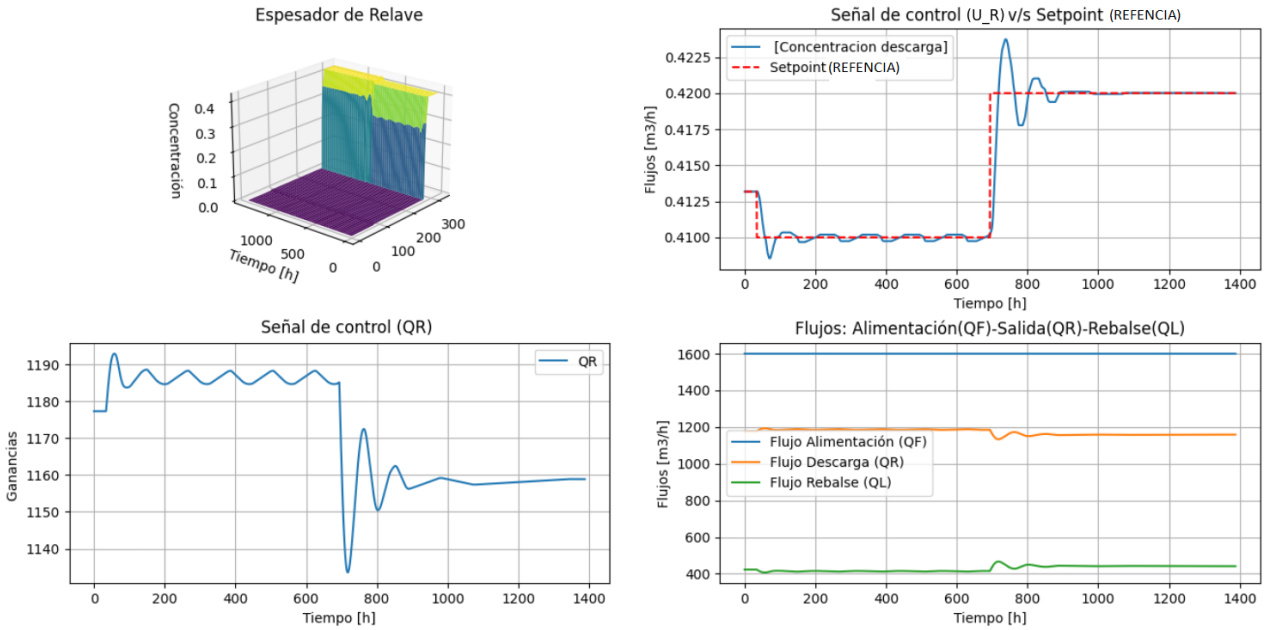

Fig. 7.10 – Simulador para espesadores de relave - con controlador.
(a) Sistema modelado en 2000 pasos (b) Sistema escalón Set Point (S.P) 0,41 al 0,42 concentraciones De descarga (ϕ_R) (c) Señal controlada flujo de descarga (ϕ_R) (d) Flujo de alimentación (Q_F), Flujo de descarga (Q_R), Flujo de rebalse (Q_L).

Para el caso 5, se realiza una evaluación detallada de un controlador del tipo PI. Según se observa en la Fig. 7.9, el controlador demuestra la capacidad para mantener su estabilidad frente variaciones en el (S.P). Este análisis revela que el controlador opera de manera efectiva, ajustando su respuesta para adaptarse a los cambios en la señal de control. Además de mantener su estabilidad, el controlador PI muestra una habilidad para seguir y mantener la señal de control con variaciones en el S.P, lo que sugiere una respuesta dinámica y adaptativa a las condiciones cambiantes del sistema.

7.3. Discusión

Según los resultados obtenidos de las pruebas realizadas, se evidencia que el controlador es capaz de adaptarse frente a variaciones en las condiciones de alimentación del sistema, mostrando una capacidad para ajustar sus variables en tiempo real. Sin embargo, al implementar el sistema de control, se observaron ciertas inestabilidades en los datos, eso se produce debido al nivel de precisión con las que se trabajan, como se visualiza en la Fig. 7.9, donde la curva presenta una falta de suavidad en las etapas finales del proceso. A pesar de esto, el controlador logra cumplir con su propósito al buscar estabilizarse y adaptarse a la señal de control frente a las variaciones en el Setpoint (S.P).

Una de las dificultades encontradas durante la simulación fue la sintonización manual del simulador, lo que resultó en una tarea poco eficiente. Si bien al estar trabajando con un sistema no lineal, y que contiene ecuaciones diferenciales parciales, su linealización resulta compleja. Esto implica que pequeñas variaciones en los parámetros de control puedan ocasionar grandes cambios en la respuesta. Dicha evaluación es compleja, debido a que la estrategia de control involucra la implementación de valores decimales, que vienen definidos como valores flotantes 32 bits. Esto implica que la transferencia de información entre lenguajes de programación Fortran y Python, debe ser exacta, ya que una transferencia poco fina puede ocasionar grandes cambios e inestabilidades inherentes.

Se prone explorar en métodos alternativas, como el lugar geométrico de las raíces, para mejorar la precisión de la sintonización. Dado que los espesadores presentan características prolongadas en su proceso físico de espesamiento, ajustar el valor de la ganancia proporcional (K_P) cada vez que se presentan inestabilidad conlleva a una ineficiencia del simulador.

Capítulo 8. Conclusiones y trabajo futuro

8.1. Sumario

Inicialmente, se llevó a cabo una investigación exhaustiva y una contextualización del área de estudio para el proyecto. Durante este proceso, se identificó una necesidad relacionada con la recuperación y tratamiento de recursos naturales, especialmente en áreas mineras donde la escasez de agua es una problemática significativa. Se identifico el rol fundamental de los espesadores de relaves en el tratamiento de minerales y recursos hídricos, ya que contribuyen de manera importante a la recuperación de agua en las zonas periféricas y su recirculación hacia otro proceso del ciclo industrial.

Posteriormente, se profundizó en el diseño de los espesadores de relave y el fenómeno físico que los rige. Se destacaron los avances en el modelo matemático y su adaptación para procesos continuos. Se enfatizo que el espesador de relaves no se limita a un simple estaque lleno, sino que opera de manera continuada, recibiendo una pulpa mediante una bandeja de alimentación, descargando los sólidos sedimentados y recuperando agua de manera simultánea. Para el modelo de los espesadores, se utilizó la teoría de Kynch como punto de partida para analizar el movimiento de las partículas y su asentamiento en el fondo a lo largo de diversas zonas de operación.

Luego, se llevó a cabo una investigación de los principales softwares que podrían ser útiles en la creación del simulador dinámico. A través de aproximaciones cualitativas en la plataforma GitHub, se determinaron las mejores opciones. Se descarto Matlab debido a su naturaleza de pago. Optando en su lugar por una estrategia de vinculación de dos softwares complementarios Fortran *(f90)* y Python (*.py*). Este vínculo se logró mediante el módulo F2py.

Posteriormente, se procedió al modelamiento del simulador, donde Fortran se utilizó para el tratamiento de datos y cálculos, mientras que Python, a través de sus extensiones, se orientó hacia la visualización de datos y la creación de la Biblioteca dinámica.

Para el final del estudio, se llevaron a cabo pruebas en el simulador para observar su comportamiento bajo diversas condiciones de carga, con tal de evaluar su capacidad de comunicación y someter al simulador a cambios en sus condiciones. Además, se expuso la implementación del controlador como parte integral del proceso de simulación, para ello se realizaron cambios en la señal de referencia setpoint (S.P), con la finalidad de ver si el espesador era capaz de mantener el flujo de descarga y concentraciones en niveles deseados.

8.2. Conclusiones

La etapa de modelamiento del simulador fue fundamental en este proyecto, donde se empleó Fortran para el tratamiento de datos y cálculos numéricos, mientras que Python, mediante sus extensiones, se encargó de visualizar y almacenar los datos en forma de matrices también contribuye a la creación de una biblioteca dinámica. Aunque es posible mejorar el método utilizado, este cumple satisfactoriamente con el desarrollo del proyecto. Principalmente se trabajó con parámetros estimados, lo que permitió que el modelo implementado se comportara de manera estable frente a modificaciones en las condiciones del proceso.

La implementación del simulador del modelo brindó una comprensión y demostración de las etapas involucradas en el proceso. Se logro visualizar las dinámicas del proceso, como la evaluación de las concentraciones en función de la altura del espesador.

Sin embargo, la implantación del controlador presentó mayores desafíos, ya que no fue complemente efectivo debido a la sensibilidad de las concentraciones y la sintonización del mismo. Además, se identificó que definir un nivel adecuado de precisión implica trabajar en 32 bits o 64 bits ya que era crucial para evitar divergencias en la simulación, considerando que el sistema es no lineal y conlleva el modelamiento de ecuaciones diferenciales parciales, por lo que su linealización es compleja. Por lo que establecer un correcto nivel de precisión de los datos se convirtió en un punto conflicto inicial para el estudio de las concentraciones.

A pesar de que el simulador actual ha demostrado la capacidad de manejar y estabilizar el proceso de espesamiento según los requisitos establecidos, existen áreas de mejoras. Se requiere una mayor atención al control del espesador y la precisión con la que se tratan los datos. Además, podría ser beneficioso investigar métodos más avanzados de control y optimización, así como explorar nuevas técnicas de modelamiento que permitan una representación más precisa del comportamiento del espesador bajo diversas condiciones operativas. Estos esfuerzos adicionales podrían contribuir significativamente a mejorar la eficiencia y efectividad del tratamiento de relaves y sistemas similares en el futuro.

8.3. Trabajos futuros

El presente simulador marca el inicio de una serie de proyectos potenciales y áreas de investigación. Si bien se ha logrado establecer un vínculo efectivo entre ambos lenguajes de programación Fortran y Python, es importante destacar que este trabajo representa solo el comienzo. La combinación entre Fortran y Python mediante la extensión del módulo F2py ha demostrado ser exitosa: Fortran es capaz de procesar grandes volúmenes de datos en tiempos reducidos, mientras que Python proporciona una herramienta robusta para la visualización y análisis de ellos. Esta sinergia podría conducir una herramienta aún más poderosa para simulación de espesadores y desarrollo de interfaz gráfica para el simulador.

Bibliografía

[1] R. Bürger, F. Concha, “Mathematical model and numerical simulation of the settling of flocculated suspensions”, International Journal of Multiphase Flow., vol.24, pp. 1005- 1023, (1998).

[2] R.Bürger, M.C. Bustos, F. Concha, “Settling velocities of particulate systems: 9. Phenomenological theory of sedimentation processes: numerical simulation of the transient behavior of flocculated suspensions in an ideal batch or continuous thickener”, International Journal of Miner Process., vol. 55, pp. 267-282, (1999).

[3] P. Garrido, R. Bürger, F. Concha, “Settling velocities of particulate systems: 11. Comparison of the phenomenological sedimentation-consolidation model with published experimental results”, International Journal of Miner., vol. 60, pp.213-227, (1999).

[4] R. Bürger, Stefan Diehl, Sebastian Faraz, Ingmar Noppen’s, “On reliable and unreliable numerical methods for the simulation of secondary settling thanks in wastewater treatment”, Computers and Chemical Engineering., vol. 41, 93-105, (2012).

[5] Fernando Betancourt, Fernando Concha, Daniel Sbárbaro, “Simple mass balance controllers for continuous sedimentation”, Computers and Chemical Engineering., vol. 54, pp. 34-43, (2013).

[6] Fernando Betancourt, Raimund Bürger, Stefan Diehl, Sebastian Faraz, “Modeling and controlling clarifier-thickeners fed by suspensions with time-dependent properties”, Minerals Engineering., vol. 62, pp. 91-101, 2014.

[7] Fernando Concha, “Manual de filtración & separación”, Talcahuano, Chile: Universidad de Concepción, 2001.

[8] Subsecretaría de economía consejo nacional de producción limpia, “Uso Eficiente de Aguas en la Industria Minera y Buenas Prácticas”, Gobierno de Chile, ministerio de minería subsecretaría de minería, Noviembre 2002.

[9] NumPy. (16 de Septiembre de 203). F2py Users Guide and Reference Manual. https://numpy.org.

[10] Comisión Chilena del cobre, “Informe de tendencias del mercado del cobre, proyecciones para los años 2024 y 2025”, Gobierno de Chile, Ministerio de Minería, Diciembre 2023.

Anexo A. Código Fortran.90 - Funciones y Ecuaciones de estado

El simulador para espesadores contempla dos códigos, asociado a distintos lenguajes, pero complementarias entre sí. Por una parte, Fortran 90 almacena el contenido y el modelamiento del espesador, incluyendo ecuaciones de estado y funciones características. Esta información estará vinculada mediante el módulo F2py con el código Python.py.

- **Archivo N°1. Fortran.90: Ecuaciones de estado.**

```
MODULE PARS

    REAL*4, PARAMETER :: x_r          = 10.00
    REAL*4, PARAMETER :: x_l          = -3.00
    REAL*4, PARAMETER :: u_crit       = 0.1
    REAL*4, PARAMETER :: Pi           = 3.1415927
    REAL*4, PARAMETER :: d            = 60.0
    REAL*4, PARAMETER :: vst          = 0.0193
    REAL*4, PARAMETER :: umax         = 1.0
    REAL*4, PARAMETER :: g            = 9.8
    REAL*4, PARAMETER :: alpha        = 4.4
    REAL*4, PARAMETER :: beta         = 10.94
    REAL*4, PARAMETER :: rho_l        = 1000.0
    REAL*4, PARAMETER :: rho_s        = 2531.0
    REAL*4, PARAMETER :: delta_rho    = rho_s-rho_l
    REAL*4, PARAMETER :: C            = 14.82

END MODULE PARS

-----------------------------------------------------------------------------------------


 FUNCTION INTERVALO_DISCRETIZACION_EJE_X(puntos) RESULT(delta_x)
    USE PARS
    IMPLICIT NONE
    INTEGER*4, INTENT(IN)             :: puntos
    REAL   *4                         :: delta_x
    delta_x = (x_r-x_l+3.0)/real(puntos)

END FUNCTION INTERVALO_DISCRETIZACION_EJE_X

-----------------------------------------------------------------------------------------
```

```
FUNCTION INTERVALO_DISCRETIZACION_EJE_T(delta_x) RESULT(delta_t)
    USE PARS
    IMPLICIT NONE
    REAL*4                              :: delta_x
    REAL*4                              :: delta_t
    delta_t = 500.0*delta_x**2.0

END FUNCTION INTERVALO_DISCRETIZACION_EJE_T

-----------------------------------------------------------------------------------------

FUNCTION LAMBDAA(delta_t,delta_x) RESULT(lambda_1)
    USE PARS
    IMPLICIT NONE
    REAL*4                              :: delta_x, delta_t
    REAL*4                              :: lambda_1

    lambda_1 =  delta_t/delta_x

END FUNCTION LAMBDAA

-----------------------------------------------------------------------------------------

FUNCTION VISCOSIDAD_FLUIDO(lambda_1,delta_x) RESULT(mu)
    USE PARS
    IMPLICIT NONE
    REAL*4, INTENT(INOUT)               :: delta_x, lambda_1
    REAL*4                              :: mu

     mu  = lambda_1/delta_x

END FUNCTION VISCOSIDAD_FLUIDO

-----------------------------------------------------------------------------------------

FUNCTION VECTOR_ESPACIO_X(puntos) RESULT(pasos_x)
    USE PARS
    IMPLICIT NONE
    INTEGER*4, INTENT(IN)               :: puntos
    INTEGER*4                           :: pasos_x

    pasos_x = puntos

END FUNCTION VECTOR_ESPACIO_X
```

```
---------------------------------------------------------------------------------------

FUNCTION CONDICION_INICIAL(x, pasos_x, delta_x) RESULT(x_1)
    USE PARS
    REAL   *4, INTENT(INOUT)                    :: x(:)
    REAL   *4, INTENT(INOUT)                    :: delta_x
    INTEGER*4, INTENT(INOUT)                    :: pasos_x
    REAL   *4, dimension(size(x))               :: x_1
    REAL   *4                                   :: aux1, aux2
    INTEGER*4                                   :: j

    x(1) = x_l - 1.0
    DO j = 2,pasos_x
        x(j) = x(j-1) + delta_x
        x_1(j-1)  = x(j)
        if (x_1(j-1) > 4.0) then
            aux1 = pi * (d/2.0 - (x(j-2)-4) * 1.732)**2
            aux2 = pi * (d/2.0 - (x(j-1)-4) * 1.732)**2
        end if
    END DO
    x_1(pasos_x+1) = x(pasos_x+1)
END FUNCTION CONDICION_INICIAL

---------------------------------------------------------------------------------------

FUNCTION
BUCLE_PRINCIPAL(x,mu,lambda_1,FC,x_1,u0_val,QF,QR,delta_x,pasos_x,pasos_t_Fortran….
u_f) RESULT(u_k)
    USE PARS
    REAL   *4, INTENT(INOUT)                    :: x_1(:)
    REAL   *4, INTENT(INOUT)                    :: x(:), FC(:)
    REAL   *4, INTENT(INOUT)                    :: mu, lambda_1
    REAL   *4, INTENT(IN)                       :: u_f, QR, QF
    INTEGER*4, INTENT(IN)                       :: pasos_t_Fortran
    REAL   *4, INTENT(INOUT), DIMENSION(0:size(x))   :: u0_val !
    REAL   *4, INTENT(INOUT)                    :: delta_x
    REAL   *4                                   :: q_f, q_r, q_l
    INTEGER*4, INTENT(INOUT)                    :: pasos_x
    REAL   *4, DIMENSION(0:size(x))             :: u_k
    REAL   *4                                   :: aux1, aux2, aux3
    REAL   *4                                   :: flux, gamma_1, A
    INTEGER*4                                   :: j, n

    q_f=QF/(pi*(d**2.0)*0.25*3600.0)
    q_r=QR/(pi*(d**2.0)*0.25*3600.0)
    q_l = q_r - q_f
```

```
    DO  n = 1,pasos_t_Fortran

        DO j=1,pasos_x-1

            aux1=flux(u0_val(j),u0_val(j+1),x_1(j)+delta_x/2.0,q_r,q_l)-
flux(u0_val(j-1),u0_val(j),x_1(j)-delta_x/2.0,q_r,q_l)

            aux2=gamma_1(x_1(j)+delta_x*0.5)*(A(u0_val(j+1))-A(u0_val(j)))-
gamma_1(x_1(j)-delta_x*0.5)*(A(u0_val(j))-A(u0_val(j-1)))

            tol=0.0001

            aux3=0.0
            if (abs(x_1(j))<tol) then
                aux3=1.0
            end if

           u_k(j)=u0_val(j)+(-lambda_1*aux1 + mu*aux2 +
lambda_1*q_f*u_f*aux3)*FC(j)
           if (u_k(j).le.0.0) then
            u_k(j) = 0.0
           end if

        END DO
        u_k(0)=u_k(1)
        u_k(pasos_x)=u_k(pasos_x-1)
        u0_val = u_k

    END DO

END FUNCTION BUCLE_PRINCIPAL

-----------------------------------------------------------------------------------------

Function A(u)
USE PARS
IMPLICIT NONE
REAL*4                      :: A
REAL*4                      :: b
REAL*4                      :: delta_u
REAL*4                      :: u
INTEGER                     :: i
if (u.le.u_crit) then
A=0.0
else
```

```
    if (u_crit.lt.u) then
    delta_u=(u-u_crit)/50.0
    A=0.0
        do i=1,50
            A=A+(b(u_crit+i*delta_u)+b(u_crit+(i-1)*delta_u))*delta_u/2.0
        end do
     end if
end if
return
end function A

-----------------------------------------------------------------------------------------

Function b(u)
USE PARS
IMPLICIT NONE
REAL*4                              :: b
REAL*4                              :: fbk
REAL*4                              :: u

if ((u_crit.lt.u).and.(u.le.umax)) then
    b=(fbk(u)*alpha*beta*exp(beta*u))/(delta_rho*g*u)
else
    b=0.0
end if
return
end function b

Function fbk(w)
USE PARS
IMPLICIT NONE
REAL*4                            :: fbk
REAL*4                            :: w

if ((0.0.le.w).and.(w.le.umax)) then
fbk=vst*w*((1.0-w/umax)**C)
else
fbk=0.0
end if
return
end function fbk

-----------------------------------------------------------------------------------------
```

```
Function gamma_1(x)
USE PARS
IMPLICIT NONE
REAL*4                          :: gamma_1
REAL*4                          :: x

if ((x_l<x).and.(x<x_r)) then
    gamma_1=1.0
else
    gamma_1=0.0
end if
end function gamma_1

-----------------------------------------------------------------------------------------

Function gamma_2(x,q_r,q_l)

IMPLICIT NONE
REAL*4              ::x, gamma_2, q_l,q_r

if (x<0.0) then
    gamma_2=q_l

else
    if (x>0.0) then
    gamma_2=q_r
    else
    write(*,*)'x=0'
    end if

end if
return
end function gamma_2

-----------------------------------------------------------------------------------------

Function flux(u,v,x,q_r,q_l)
USE PARS
IMPLICIT NONE
REAL*4                          ::  flux, God
REAL*4                          ::  u,v
REAL*4                          ::  x
REAL*4                          ::  q_r
REAL*4                          ::  q_l
```

```
if (x<x_l) then

  flux=q_l*v

else
    if (x<0.0) then

    flux=q_l*v + God(u,v)
    else

         if (x<x_r) then

        flux=q_r*u + God(u,v)

     else

        flux=q_r*u

    end if
    end if
end if

return

-----------------------------------------------------------------------------------------

Function God(u,v)
USE PARS
IMPLICIT NONE
REAL*4                      ::  God, aux2
REAL*4                      ::  fbk
REAL*4                      ::  aux
REAL*4                      ::  u,v

      aux=1.0/(1.0-C)
  if (u.le.v) then

      God=min(fbk(u),fbk(v))

  else
      aux2=(aux-u)*(aux-v)

      if (aux2<0) then

      God=fbk(aux)
```

```
      else

      God=max(fbk(u),fbk(v))

      end if
  end if

return

end function God
```

- **Archivo N°2. Python.py: Definición variables y graficas representativas.**

Una vez establecido el vínculo entre ambos lenguajes de programación, desde el archivo Python.py se pueden modificar las variables en Fortran 90, como también presentar las gráficas asociadas al comportamiento del espesador.

```
import numpy as np
import SIM_SS01
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation


def intervalo_discretizacion_eje_x(puntos):
    delta_x = SIM_SS01.intervalo_discretizacion_eje_x(np.float32(puntos))
    return delta_x

puntos      =   320

delta_x     =   intervalo_discretizacion_eje_x(puntos)

print("Resultados intervalo_discretizacion_eje_x (delta_x):",delta_x)
```

```
def intervalo_discretizacion_eje_t(delta_x):
    delta_t = SIM_SS01.intervalo_discretizacion_eje_t(np.float32(delta_x))
    return delta_t
```

```
delta_t     =   intervalo_discretizacion_eje_t(delta_x)

print("Resultados intervalo_discretizacion_eje_t (delta_t):",delta_t)

-----------------------------------------------------------------------------------------

def lambdaa(delta_t,delta_x):
    lambda_1 = SIM_SS01.lambdaa(np.float32(delta_t),np.float32(delta_x))
    return lambda_1

lambda_1     =   lambdaa(delta_t,delta_x)

print("Resultados Lambdaa (lambda_1):",lambda_1)

-----------------------------------------------------------------------------------------

def viscosidad_fluido(lambda_1,delta_x):
    mu = SIM_SS01.viscosidad_fluido(np.float32(lambda_1),np.float32(delta_x))
    return mu

mu     =   viscosidad_fluido(lambda_1,delta_x)

print("Resultados Viscosidad fluido (mu):", mu)

-----------------------------------------------------------------------------------------

def vector_espacio_x(puntos):
    pasos_x = SIM_SS01.vector_espacio_x(np.float32(puntos))
    return pasos_x

pasos_x     =   vector_espacio_x(puntos)

print("Rango discretizacion (pasos_x):", pasos_x)

-----------------------------------------------------------------------------------------

def condicion_incial(x, pasos_x, delta_x):
    x_1 =
SIM_SS01.condicion_inicial(np.float32(x),np.float32(pasos_x),np.float32(delta_x))
    return x_1[:-1]

x         =   np.zeros(pasos_x+1)
x_1       =   condicion_incial(x,pasos_x, delta_x)

-----------------------------------------------------------------------------------------
```

```
QF                  = 1600.13
QR                  = 1177
u_f                 = 0.304
u_d                 = u0_val[-1]
QR                  = QF * u_f / u_d


pasos_t_Python      = 1000
pasos_t_Fortran     = 1000

-----------------------------------------------------------------------------------------

def bucle_principal(x, mu, lambda_1, FC, x_1, u0_val, QF, QR, delta_x, pasos_x,
pasos_t_Fortran, u_f):
    u_k =
SIM_SS01.bucle_principal(np.float32(x),np.float32(mu),np.float32(lambda_1),np.float
32(FC),np.float32(x_1),np.float32(u0_val),np.float32(QF),
np.float32(QR),np.float32(delta_x),np.float32(pasos_x),np.float32(pasos_t_Fortran),
np.float32(u_f))
    return u_k
x                 = np.zeros(pasos_x)
FC                = np.ones(pasos_x)

u_k = bucle_principal(x, mu, lambda_1, FC, x_1, u0_val, QF, QR, delta_x, pasos_x,
pasos_t_Fortran, u_f)

print("Resultados UK PYTHON:")
print(u_k)
print(len(u_k))

-----------------------------------------------------------------------------------------

fig = plt.figure(figsize=(15, 15))

ax1 = fig.add_subplot(221, projection='3d')

ax2 = fig.add_subplot(222)

ax3 = fig.add_subplot(223)


matriz_resultados = np.zeros((pasos_t_Python, pasos_x), dtype=np.float32)

resultados_lista        = []

tiempo                  = []
qf_data                 = []
```

```
qr_data                = []
ql_data                = []

u_f_data               = []
u_d_data               = []

-------------------------------------------------------------------------------------------

for i in range(pasos_t_Python):

    u_k =
SIM_SS01.bucle_principal(np.float32(x),np.float32(mu),np.float32(lambda_1),np.float
32(FC),np.float32(x_1),np.float32(u0_val),np.float32(QF),
np.float32(QR),np.float32(delta_x),np.float32(pasos_x),np.float32(pasos_t_Fortran),
np.float32(u_f))

    matriz_resultados[i, :(len(u_k)-1)] = u_k[1:]

    if i == 20:
        QF = 1200
    elif i == 60:
        QF = 961

    u0_val = u_k

    resultados_lista.append(matriz_resultados[i])

    print("Python: u_k[-2]%1.10F", u_k[-2])
    print("Numero de iteracion", i)

-------------------------------------------------------------------------------------------

    x_vals = np.arange(0, pasos_x)
    t_vals = np.arange(0, pasos_t_Python)*(delta_t/3600)*pasos_t_Fortran
    x_grid, t_grid = np.meshgrid(x_vals, t_vals)

    ax1.cla()
    ax1.plot_surface(x_grid, t_grid, matriz_resultados, cmap='viridis')
    ax1.set_title('Espesador de Relave')
    ax1.set_xlabel('')
    ax1.set_ylabel('Tiempo [h]')
    ax1.set_zlabel('Concentración')

    tiempo.append(i * (delta_t/3600) * pasos_t_Fortran)
    qf_data.append(QF)
```

```
    qr_data.append(QR)

    ql_data.append(QR-QF)

    ax2.cla()
    ax2.plot(tiempo, qf_data, label='Flujo Alimentación (QF)')
    ax2.plot(tiempo, qr_data, label='Flujo Descarga (QR)')
    ax2.plot(tiempo, ql_data, label='Flujo Rebalse (QL)')
    ax2.set_xlabel('Tiempo [h]')
    ax2.set_ylabel('Flujos [m3/h]')
    ax2.set_title('Flujos: Alimentación(QF)-Salida(QR)-Rebalse(QL)')
    ax2.grid(True)
    ax2.legend()

    u_f_data.append(u_f)
    u_d_data.append(u_k[-2])
    print("Python: u_k[-2]", u_k[-2])

    ax3.cla()  # Limpia el subplot 2D
    ax3.plot(tiempo, u_f_data, label='Concentración Alimentación (u_f)')
    ax3.plot(tiempo, u_d_data, label='Concentración Descarga (u_d)')
    ax3.set_xlabel('Tiempo [h]')
    ax3.set_ylabel('Concentraciones volumentrica')
    ax3.set_title('Concentración Alimentacion (u_f) v/s Concentración Salida
(u_d)')
    ax3.grid(True)
    ax3.legend()

    plt.pause(0.01)

print("Matriz de resultados:")
print(matriz_resultados)

plt.tight_layout()
plt.show()

-----------------------------------------------------------------------------------------------
```

Anexo B. Implementación de la serie de Fibonacci

Se presenta la implementación mediante la serie de Fibonacci utilizada como comprobación y validación de la correcta vinculación entre ambos lenguajes de programación. Esta implementación sirve como un ejemplo que permite demostrar los tiempos de ejecución de Fortran, Python y como se relaciona con el módulo F2py.

- **Archivo N°1. Fortran.90: Ecuaciones de Fibonacci desde Fortran.**

```
Recursive function fibonacci_recursive(n) result(fib)

  integer, intent(in)           :: n
  integer                       :: fib

  if (n <= 0) then
    fib = 0
  elseif (n == 1) then
    fib = 1
  else
    fib = fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)
  endif

end function fibonacci_recursive

function fibonacci_iterative(n) result(fib)
  integer, intent(in)           :: n
  integer                       :: fib
  integer                       :: a, b, i

  if (n <= 0) then
    fib = 0
  elseif (n == 1) then
    fib = 1
  else
    a = 0
    b = 1
    do i = 2, n
      fib = a + b
      a = b
      b = fib
    end do
  endif
end function fibonacci_iterative
```

- **Archivo N°2.Python: Serie de Fibonacci desde Python.**

```
import time
import fibonacci                    # Importa el módulo de extensión generado por F2py
import matplotlib.pyplot as plt

# Lista de valores de n para los cuales queremos calcular la serie de Fibonacci
valores_de_n = [10, 20, 30, 40]

resultados = []                     # Lista para almacenar los resultados
tiempos = []                        # Lista para almacenar los tiempos de ejecución

for n in valores_de_n:
    start_time = time.time()                    # Tiempo de inicio de la ejecución
    result = fibonacci.fibonacci_recursive(n)   # Llama a la función de Fortran desde Python
    end_time = time.time()                      # Tiempo de finalización de la ejecución
    elapsed_time = end_time - start_time        # Tiempo total de ejecución

    resultados.append(result)
    tiempos.append(elapsed_time)

    print(f"Para n = {n}, el resultado es: {result}. Tiempo de ejecución: {elapsed_time:.6f} segundos.")

# Graficar los resultados
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(valores_de_n, resultados, marker='o')
plt.xlabel('Valor de n')
plt.ylabel('Resultado de Fibonacci')
plt.title('Serie de Fibonacci - Resultados')

plt.subplot(2, 1, 2)
plt.plot(valores_de_n, tiempos, marker='o', color='r')
plt.xlabel('Valor de n')
plt.ylabel('Tiempo de ejecución (segundos)')
plt.title('Serie de Fibonacci - Tiempo de ejecución')

plt.tight_layout()
plt.show()
```