EFECTO DEL NÚMERO DE VARIABLES Y RESTRICCIONES SOBRE EL DESEMPEÑO COMPUTACIONAL DE UN SBC EN LA EJECUCIÓN DE UN NMPC

Sandra Milena Rodríguez Mancera

Facultad de Ingeniería
Maestría diseño y gestión de procesos - Énfasis en Procesos Químicos

Chía - Cundinamarca

Julio 2021

EFECTO DEL NÚMERO DE VARIABLES Y RESTRICCIONES SOBRE EL DESEMPEÑO COMPUTACIONAL DE UN SBC EN LA EJECUCIÓN DE UN NMPC

Sandra Milena Rodríguez Mancera

Director: Msc. Manuel Alfredo Figueredo
Codirector: PhD. Edgar Yesid Mayorga

Facultad de Ingeniería
Maestría diseño y gestión de procesos - Énfasis en Procesos Químicos

Chía - Cundinamarca

Julio 2021

Agradecimientos

Agradezco a todas las personas que han contribuido en la realización de este trabajo.

A mi familia, mi Hija y mi Mamá, quienes son mi fuente de inspiración. Gracias por su dedicación y paciencia.

A los profesores Edgar Mayorga y Manuel Figueredo por sus aportes, tiempo y acompañamiento.

Al profesor John Hedengren, sus explicaciones fueron fundamentales durante este proceso.

A mis amigos, su constante apoyo me dio fuerzas para seguir adelante.

A la Universidad de La Sabana, por su calurosa acogida.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El modelo de control predictivo MPC, *model predictive control* por sus siglas en inglés, es una estrategia de control de procesos caracterizada por su baja variabilidad favoreciendo la productividad, el uso eficiente de los recursos y la calidad de los productos. De amplio uso en industrias como refinerías, petroquímicas, manipulación de alimentos y minería, esta estrategia exhibe resultados satisfactorios principalmente en sistemas multivariables no lineales [1].

Fundamentado en modelos empíricos o en leyes de la conservación, representadas por medio de sistemas de ecuaciones diferenciales y algebraicas, lineales o no lineales, el comportamiento dinámico de un proceso es predicho en un horizonte de tiempo finito, comparándose continuamente con las mediciones reales y una trayectoria o punto de ajuste definido. Este proceso se ejecuta continuamente de modo que, en cada iteración, se ejecutan cambios en la o las variables manipuladas, sujetos a restricciones del proceso.

Este proceso involucra transformaciones de las ecuaciones diferenciales en sistemas de ecuaciones algebraicas, evaluación de derivadas, compilación del código, entre otras actividades que deben ser resueltas en instantes cortos de modo que la predicción sea realista y retroalimentada continuamente, requiriendo de algoritmos númericos eficientes, principalmente en el caso de modelos no lineales [2]. Diferentes autores [2], [3], [4], [5] han evaluado nuevos algoritmos y combinaciones de estos buscando respuestas ágiles y numericamente adecuadas, siendo esta área de gran interés a nivel académico.

Esta velocidad, medida a través del tiempo computacional, también se atribuye al hardware empleado en la resolución del problema de optimización. Actualmente existen diferentes tipos de hardware dedicados al desarrollo del MPC, encontrándose desde servidores de alto desempeño hasta dispositivos de bajo costo, siendo estos últimos importantes ya que su desarrollo ha impactado diversas áreas al brindar mayores capacidades de cómputo en tamaños más reducidos y, en consecuencia, mayores beneficios económicos, favoreciendo su uso a nivel acádemico e industrial.

Considerando lo anterior, el presente trabajo integra un NMPC empleando un Software de optimización dinámica para una columna de destilación continua en un computador de placa única (Tarjeta Raspberry PI 3B+) . Esta operación unitaria, caracterizada por el número de ecuaciones relacionadas, analiza 3 modelos, fundamentados en los principios de conservación de masa, energía y equilibrio termodinámico del sistema. Evaluando el desempeño del NMPC y el tiempo computacional de la tarjeta frente a cambios en el número de variables manipuladas.

Capítulo 2

Justificación

El control regulatorio, como primera y más conocida estrategia para realizar control de procesos, ha sido ampliamente utilizado en todas las industrias de manufactura donde su lógica y facilidad de implementación ha permitido mantener las variables controladas en el valor deseado con una variabilidad mínima. Como desventaja de este tipo de control se encuentra una pobre respuesta frente a largos tiempos muertos, un bajo desempeño en procesos integrantes, imposibilidad de ser usado en sistemas de múltiples entradas y múltiples salidas (MIMO) y dentro de su lógica de control no pueden aplicarse restricciones para llevar la variable controlada a un nuevo estado estable [6].

Como respuesta a las limitaciones presentadas por el control regulatorio surge el modelo de control predictivo (MPC), con un uso importante en la industria, reportando para 1999 más de 4500 aplicaciones en el mundo, principalmente en las refinerías y en la industria petroquímica [1]. Entre sus ventajas están la mejora en la calidad de la producción, disminución de la variabilidad y reducción los costos de energía [1], [7].

A través del control MPC se predice una secuencia de movimientos de una o varias variables manipuladas en un horizonte de predicción finito, los cuales son implementados sucesivamente y corregidos a partir de la respuesta del proceso. Esta secuencia se repite a lo largo de la operación de una planta [1]. Para la implementación de un controlador MPC es necesario contar con un modelo matemático del proceso, descrito por ecuaciones diferenciales y algebraicas DAE y una función objetivo a ser optimizada [8]. Este modelo se puede obtener a través del planteamiento de balances de masa y energía o mediante el uso de técnicas de identificación de procesos [9]. La función a optimizar puede generarse a partir de la pureza requerida o los costos energéticos sujetos a restricciones físicas, económicas o de composición.

La figura 2.1 presenta la jerarquía de las actividades de control de procesos, conformada por las funciones necesarias ubicadas en la zona inferior y las opcionales en los niveles superiores. La escala de tiempo para cada actividad es presentada al lado derecho [1], [10]. La escala asignada para la ejecución del MPC es del orden de minutos a horas.

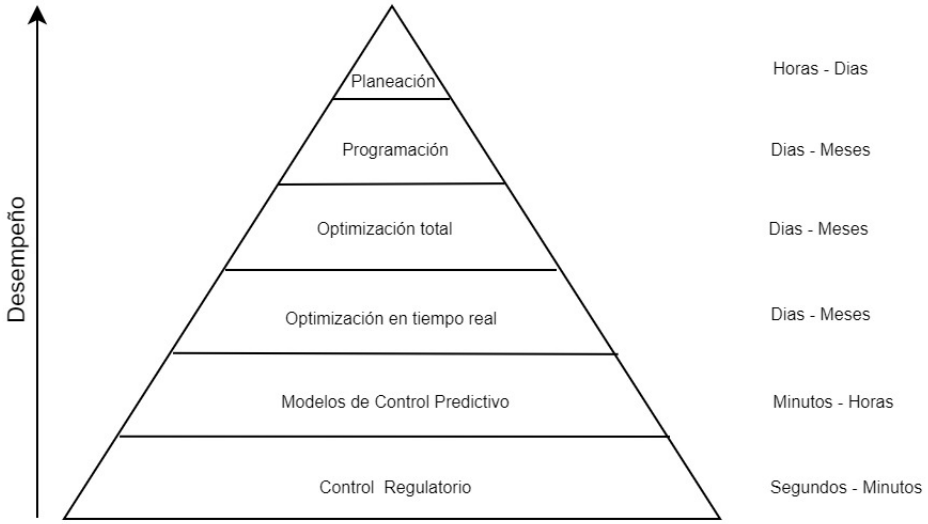

Figura 2.1: Jerarquía de las actividades de Control

Es fundamental garantizar que el algoritmo de optimización junto con la solución del sistema DAE sea resuelto en intervalos cortos de modo que este se encuentre dentro de los límites establecidos en la figura 2.1. Este intervalo depende de factores como el modelo propuesto, número de variables manipuladas y controladas, software y, finalmente, el hardware sobre el que se resuelve el problema de optimización.

Diferentes proveedores ofrecen alternativas de implementación MPC bastante robustas y completas. Dentro de estas alternativas se tienen Pavilion 8 MPC, PlantPAx® MPC de Rockwell Automation, System 800x for APC, ABB Ability™ Advanced Process Control 'I&Ánalytics (APCA) Suite de ABB y Profit Suite de Honeywell. La mayoría de estas plataformas requieren de infraestructuras de control como Sistemas de Control Distribuido (DCS por sus siglas en inglés) y equipos de hardware con altas capacidades de cómputo que elevan su costo de manera considerable. En oposición a este escenario, cabe notar la creciente tendencia a la miniaturización de los dispositivos, lo cual ha impactado diversas áreas de la industria al brindar mayores capacidades de cómputo en tamaños más reducidos y, en consecuencia, mayores beneficios económicos, lo que permite su acceso a un mayor número de usuarios [11]. Dentro de estas tecnologías se encuentran los computadores de placa única SBC por sus siglas en Inglés *Single Board Computer*, que son ordenadores completos en las que una placa de circuito único comprende la memoria, la entrada/salida un microprocesador y todas las demás características necesarias. Debido a su nivel de integración y reducción de componentes y conectores, los SBC suelen ser más pequeños, livianos, económicos y con un mejor manejo de la potencia eléctrica que los computadores de múltiples tarjetas. Programables a través de Python, que es un lenguaje de programación orientado a objetos, de fácil programación, de acceso libre, considerado como el lenguaje de programación número uno en 2017 [12] y con variedad de librerías en las que pueden ejecutarse tareas de optimización de procesos. Su constante evolución los ha llevado a elevar su rendimiento hasta cuatro veces en 5 años, ganándose la posibilidad de ser usados tanto en ambientes académicos como industriales [13]. Dentro de los SBC mas comunes se encuentran las tarjetas Arduino, el ECB AT91, los Gumstiks y las tarjetas Raspberry Pi.

2.1. Tarjetas Raspberry Pi

Introducida al mercado en marzo de 2018, de un tamaño pequeño, alta capacidad y memoria y diversos protocolos de comunicación [14], cuenta con sistemas operativos basados en Unix, cuya principal distribución es Ubuntu, lo que le permite ser programada a través de Python [15]. El cuadro 2.1 presenta las características de una tarjeta Raspberry Pi 3B+.

Característica	Valor
Procesador	Broadcom BCM2837B0 / Cortex-A53 (ARMv8) / 64-bit SoC
Frecuencia de Reloj	1,4 GHz
Memoria	1GB LPDDR2 SDRAM
Conectividad Inhalambrica	2.4GHz / 5GHz / IEEE 802.11.b/g/n/ac Bluetooth 4.2, BLE
Conectividad de Red	Gigabit Ethernet over USB 2.0 / 300 Mbps de máximo teórico
Puertos	PIO 40 pines / HDMI / 4 x USB 2.0 / Micro SD / Micro USB (alimentación) Power-over-Ethernet (PoE)

Cuadro 2.1: Características de la Tarjeta Raspberry Pi 3B+

Estas tarjetas han sido evaluadas para determinar el desempeño del MPC, encontrando resultados satisfactorios en tanques interactuantes [14] y sistemas de control de presión [16].

El desempeño de ese sistema de cómputo puede ser medido a través del Tiempo Computacional, definido como el tiempo que toma al conjunto hardware y software desarrollar los cálculos respectivos para resolver una tarea específica [4], [17].

2.2. Tiempo Computacional

Bibliotecas de Python permiten medir este tiempo y discriminarlo para algunas actividades involucradas en la ejecución del NMPC como son tiempo de ejecución del optimizador, cálculos de derivadas, discretización de ecuaciones diferenciales entre otras [12].

Adicionalmente, la función *Time* de la librería Time de Python, mide todo el tiempo requerido por el sistema para realizar la comunicación, procesamiento de datos de entrada solución del solver y escritura de los archivos de solución. Este tiempo, tambien es conocido como *Wall Time*.

Estas herramientas, en conjunto, permiten calcular el tiempo, denominado como esfuerzo computacional para realizar toda la ejecución del NMPC.

2.3. Antecedentes de Implementación del NMPC

Varios autores han implementado el NMPC para diferentes procesos químicos con distintas interfases de resolución de ecuaciones diferenciales y optimización. El cuadro 2.2 presenta algunos de sus resultados.

Autor	Año	Proceso	Interfase	Hardware
Federico Lozano Santamaria Jorge M.Gomez [18]	2016	Reactor Batch	Pyomo	No especificado
Yankai Cao David Acevedo Zoltan K. Nagy Carl D.Laird [19]	2017	Cristalizador Batch	Modelica	Intel(R), Xeon(R)
David E. Bernal Carolina Carrillo-Diaz Jorge M. Gomez Luis A. Ricardez-Sandoval[20]	2018	Columna de destilación extractiva	Gams	Intel Core i5 2.70 GHz, 8.0 GB
Logan D.R. Beal Damon Petersen David Grimsman Sean Warnick John D. Hedengren [21]	2018	Reactor CSTR	Gekko	Intel i7 CPU-6700 at 3.40 GHz
Amir Patel Stacey Leigh Shield Saif Kazi Aaron M. Johnson Lorenz T. Biegler[2]	2019	Robots	Gams	Intel Xeon 2.2 GHz 32 GB RAM
Flemming Holtorf Alexander Mitsos Lorenz T. Biegler [3]	2019	Reactor de Polimerización Semibatch	Pyomo	No indicado
Álan C. E Soua Valter J. S. Leite Ignacio Rubio Scola [14]	2018	Sistema de cuatro tanques interactuantes	Scipy y otras librerías	Tarjeta Raspberry PI 3B
V. Bagyaveereswaran Tushar D. Mathur Sukrit Gupta P. Arulmozhivarman [16]	2016	Sistema de Control de Presión	Matlab	Tarjeta Arduino

Cuadro 2.2: Antecedentes de Implementación del NMPC.

Esta cuadro presenta las diferentes posibilidades de configuración entre proceso, software y hardware para el NMPC. Si bien ya se han realizado pruebas en tarjetas Raspberry Pi, no se ha reportado el tiempo computacional de esta ni el efecto del numero de variables de estado en su desempeño. Tampoco se ha reportado su uso para sistemas de mayor complejidad como es el proceso de destilación. Dado el anterior contexto, este trabajo tiene los siguientes objetivos.

2.4. Objetivos del presente trabajo

2.4.1. General

Evaluar la ejecución del Modelo de Control Predictivo en una tarjeta Raspberry Pi 3B+ usando el Tiempo Computacional como una función del número de ecuaciones y variables involucradas.

2.4.2. Específicos

Implementar modelos de control predictivo sobre una tarjeta Raspberry Pi usando la ley de control y tres modelos de simulación del proceso.

Analizar el efecto del número de variables y ecuaciones sobre el tiempo computacional empleado para la solución de los MPC implementados.

Antes de iniciar con este estudio, se presenta la notación utilizada en el documento. Posteriormente, el capitulo 3 se expone la teoria relacionada con optimización dinámica, NMPC, cuadratura Gaussiana y softwares de optimización dinámica en Python. El capitulo 4 presenta

los tres modelos evaluados, consideraciones de cada uno, ecuaciones y variables asociadas al proceso de destilación. El capitulo 5 muestra la metodologia utilizada para cumplir los objetivos, se presenta la arquitectura de comunicación del NMPC vía OPC (OLE for Process Control) para comunicarse con Aspen Dynamics. Los resultados y análisis del estudio son presentados en el capitulo 6 y, finalmente, los capitulos 7,8 y 9 exponen las conclusiones, recomendaciones y participación en ponencias de este trabajo respectivamente.

Notación

Acrónimos

AML	Algebraic Modeling Languaje
DAE	Diferential Algebraic Equation
DCS	Distribuited Control System
EDO	Equation Diferential Ordinary
FPGA	Field Programmable Gate Arrays
IPOPT	Método del punto Interior
KKT	Karush -Kuhn-Tucker
LP	Programación Lineal
MHE	Moving Horizont Estimation
MILP	Programación Lineal Entera Mixta
MINLP	Programación no Lineal Entera Mixta
MPC	Model Predictive Control
NLP	Programación no Lineal
NMPC	Non Lineal Model Predictive Control
NRTL	Non Ramdon Two Liquid
OPC	OLE for process control
PM	Problema de minimización
PMD	Problema de Minimización con Desigualdades
PMDI	Problema de Minimización con Desigualdades e Igualdades
QP	Programación Cuadrática
SBC	Single Board Computer

Subíndices

i	Componente
j	Plato
k	Interacción
m	Ecuaciones de Igualdad
l	Ecuaciones de Desigualdad

Símbolos Griegos

$\boldsymbol{\lambda}$	Multiplicadores de Lagrange asociado a la desigualdad
$\boldsymbol{\beta}$	Multiplicadores de Lagrange asociado a la igualdad
$\mathcal{D}$	Región Factible

ω	Función de Ponderación de Peso
α	Volatilidad Relativa
γ	Coeficiente de Actividad

Símbolos Latinos

A^T	Matriz Transpuesta
C	Número de Componentes
g	Función de Igualdad
h	Función de Desigualdad
HV	Entalpia del Vapor
HL	Entalpia del Líquido
K	Constante de Equilibrio
L	Flujo Molar de Líquido dentro de la columna
M	$HoldUp$ del sistema
P	Número de Platos
Ps	Presión
P^{sat}	Presión de Vapor
Q	Energia
T_c	Temperatura Crítica
T_r	Temperatura Reducida
U	Flujo Molar de Líquido saliendo de la columna
U	Energia Interna del sistema
V	Flujo Molar de Vapor dentro de la columna
W	Flujo Molar de Vapor saliendo de la columna
X	Composición Molar de Líquido
Y	Composición Molar de Vapor

Capítulo 3

Optimización Dinámica

3.1. Optimización

Generalmente, en las aplicaciones de la ingeniería química se buscan mejoras en eficiencia, impacto ambiental, calidad y rentabilidad de determinado proceso. La matemática ha explorado este campo mediante la modelación y la optimización, convirtiendose en una fuente prolífica de problemas desafiantes durante más de los últimos 50 años [2].

Un problema típico de modelación matemática susceptible a ser optimizado consiste en un problema de la forma

$$\begin{aligned} \min_{\mathbf{x}} \quad & F(\mathbf{x}) \\ sujeto\ a: \ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}, \end{aligned} \tag{3.1}$$

donde F es una función a valor real definida sobre $\mathbb{R}^n$ y $\mathbf{g}$ y $\mathbf{h}$ son vectores de funciones a valor real que representan las restricciones de desigualdad e igualdad, respectivamente. El objetivo, de este tipo de problemas, es determinar el valor de $\mathbf{x} \in \mathbb{R}^n$ que hace que $F(\mathbf{x})$ sea mínimo y satisfaga las restricciones.

3.1.1. Optimización no lineal

La metodología matemática que se usa para resolver problemas con la estructura (3.1) depende de la forma de las funciones. En caso de que tanto F, como $\mathbf{g}$ y $\mathbf{h}$ sean funciones lineales, el problema hace parte de la Programación lineal (LP), el cual generalmente se resuelve usando el algoritmo simplex [22]. En caso de que al menos una de las coordenadas de $\mathbf{x}$ sea una variable entera o binaria, el problema hace parte de la programación entera mixta (MILP), nombre que hace referencia a que se pueden mezclar variables reales, enteras y/o binarias; este tipo de problemas generalmente se resuelve con algoritmos adaptativos del método simplex o algoritmos de ramificación [23]. Cuando no todas las funciones F, $\mathbf{g}$ y $\mathbf{h}$ en (3.1) son lineales, el problema hace parte de la programación no lineal (NLP) [3]. En caso de que al menos una de las coordenadas de $\mathbf{x}$ sea una variable entera o binaria, el problema hace parte de la programación no lineal mixta (MINLP) [2].

Cuando la no linealidad del problema (3.1) ocurre porque la función objetivo F es una función multivariable de orden dos, el problema hace parte de la programación cuadrática (QP), el cual recurre al método de mínimos cuadrados o algoritmos relacionados. Ahora, cuando la

función objetivo y/o las restricciones son no lineales, generalmente el problema (3.1) se transforma mediante aproximaciones locales de segundo orden, lo cual genera condiciones necesarias y suficientes para determinar si en un valor específico $\mathbf{x}$ ocurre el mínimo local de $F(\mathbf{x})$. Las condiciones antes mencionadas, se conocen como las condiciones de primer y segundo orden de Karush-Kuhn-Tucker (KKT) [10].

Condiciones de Karush-Kuhn-Tucker (KKT)

Considere el siguiente problema de minimización (PM)

$$\begin{aligned} \min_{\mathbf{x}} \ & F(\mathbf{x}) \\ & \mathbf{x} \in \mathcal{D} \subset \mathbb{R}^n \end{aligned} \tag{3.2}$$

donde F es una función a valor real diferenciable sobre $\mathcal{D}$. Cuando el conjunto $\mathcal{D}$ contiene su frontera y puede ser encerrado en una bola de radio finito, es posible demostrar la existencia de la solución del problema (3.2), más no que el valor x donde ocurre el mínimo sea único [24]. En el problema (3.2), el conjunto $\mathcal{D}$ se conoce como la *región factible* [22], esta región generalmente es descrita algebraicamente con desigualdades, problema de minimización con desigualdades (PMD)

$$\begin{aligned} \min_{\mathbf{x}} \ & F(\mathbf{x}) \\ & g_1(\mathbf{x}) \leq 0 \\ & g_2(\mathbf{x}) \leq 0 \\ & \vdots \\ & g_m(\mathbf{x}) \leq 0 \end{aligned} \tag{3.3}$$

o combinación entre desigualdades e igualdades, problema de minimización con desigualdades e igualdades (PMDI)

$$\begin{aligned} \min_{\mathbf{x}} \ & F(\mathbf{x}) \\ & g_1(\mathbf{x}) \leq 0 \\ & g_2(\mathbf{x}) \leq 0 \\ & \vdots \\ & g_m(\mathbf{x}) \leq 0 \\ & h_1(\mathbf{x}) = 0 \\ & h_2(\mathbf{x}) = 0 \\ & \vdots \\ & h_l(\mathbf{x}) = 0, \end{aligned} \tag{3.4}$$

donde las funciones g_i para $1 \leq i \leq m$ y h_j para $1 \leq j \leq l$ son diferenciables sobre $\mathcal{D}$. Aunque el problema (3.1) es la forma simplificada de escribir cualquiera de los dos problemas (3.3) y (3.4), para la formulación de las condiciones de KKT es necesario explicitar la forma de los vectores $\mathbf{g}$ y $\mathbf{h}$.

Se dice que $g_i(\mathbf{x})$ es una *desigualdad activa* en $\bar{\mathbf{x}} \in \mathcal{D}$ si cumple exactamente la igualdad, es decir, $g_i(\bar{\mathbf{x}}) = 0$. Para un punto $\bar{\mathbf{x}}$ en la región factible, se denotará por $\mathcal{I}$ el conjunto de índices de las desigualdades activas . Un punto $\bar{\mathbf{x}}$ en la región factible para el problema (3.3) se dice *regular* si los gradientes $g_i'(\bar{\mathbf{x}})$ son linealmente independientes o $\mathcal{I} = \emptyset$. Un punto $\bar{\mathbf{x}}$ factible se conoce como un *minimizador local* para el problema (3.2), si existe una bola B_r con radio r y centro en $\bar{\mathbf{x}}$ tal que $F(\mathbf{x}) \geq F(\bar{\mathbf{x}})$ para todo $\mathbf{x} \in B_r \cap \mathcal{D}$.

El problema (3.2) puede tener minimizadores locales que no son la solución del problema, sin embargo, el punto donde se alcanza el valor mínimo debe ser un optimizador local. El siguiente teorema, conocido como las condiciones de KKT de primer orden, identifica características que tiene un valor factible en caso de ser un optimizador local, resultado ser una aproximación para determinar la posible solución del problema (3.3) [23].

Condiciones necesarias de KKT para el PMD Sea $\bar{\mathbf{x}} \in \mathcal{D}$ un valor regular del PMD. Si $\bar{\mathbf{x}}$ es un minimizador local de PMD, entonces existen escalares λ_i para $i \in \mathcal{I}$, tales que

$$F'(\bar{\mathbf{x}}) + \sum_{i \in \mathcal{I}} \lambda_i g_i'(\bar{\mathbf{x}}) = 0, \quad \text{con} \quad \lambda_i \geq 0 \quad \text{para} \quad i \in \mathcal{I}, \tag{3.5}$$

los coeficientes λ_i son conocidos como los *multiplicadores de Lagrange*.

Si $\bar{\mathbf{x}}$ es regular y está en la región factible, se debe comprobar si existen los coeficientes λ_i que satisfacen las condiciones de primer orden de KKT. En términos generales, se debe resolver un problema de la forma

$$A\boldsymbol{\lambda} = \mathbf{b}\boldsymbol{\lambda} \geq \mathbf{0} \tag{3.6}$$

donde A es una matriz de tamaño $n \times \bar{m}$, siendo n el número de variables y $\bar{m}$ el número de desigualdades activas; $\mathbf{b} = -F'(\bar{\mathbf{x}})$ vector columna $n \times 1$ compuesto por el gradiente de F y $\boldsymbol{\lambda}$ el vector columna $\bar{m} \times 1$ compuesto por las variables λ_i para $i \in \mathcal{I}$. Ya que $\bar{\mathbf{x}}$ es regular, $\bar{m} \leq n$.

Intuitivamente, la forma de resolver el problema anterior es resolver el sistema $A\boldsymbol{\lambda} = \mathbf{b}$ y en caso de existir la solución comprobar si esta es no negativa. Sin embargo, el problema también se podría resolver, solucionando el siguiente problema estándar de programación lineal

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \; z &= \mathbf{0^T}\boldsymbol{\lambda} \\ A\boldsymbol{\lambda} &= \mathbf{b} \\ \boldsymbol{\lambda} &\geq \mathbf{0}, \end{aligned} \tag{3.7}$$

donde $\mathbf{0^T}\boldsymbol{\lambda}$ representa la función nula, es decir, no existe función a optimizar [22].

Para el problema PMDI sea $p = \bar{m} + l$, es decir, p es el número de desigualdades activas más el número de restricciones de igualdad. Un punto $\bar{\mathbf{x}}$ en la región factible se dice regular si los gradientes $g_i'(\bar{\mathbf{x}})$ con $i \in \mathcal{I}$ y $h_j'(\bar{\mathbf{x}})$ para todo j son linealmente independientes, o si $p = 0$.

Condiciones necesarias de KKT para el PMDI Sea $\bar{\mathbf{x}} \in \mathcal{D}$ un valor regular del PMD. Si $\bar{\mathbf{x}}$ es un minimizador local de PMDI, entonces existen escalares λ_i para $i \in \mathcal{I}$ y β_j para todo j, tales que

$$F'(\bar{\mathbf{x}}) + \sum_{i \in \mathcal{I}} \lambda_i g_i'(\bar{\mathbf{x}}) + \sum_{j=1}^{l} \beta_j h_j'(\bar{\mathbf{x}}) = 0, \quad \text{con} \quad \lambda_i \geq 0 \quad \text{para} \quad i \in \mathcal{I}, \tag{3.8}$$

los escalares β_i también se conocen como multiplicadores de Lagrange, pero esta vez asociados a las igualdades.

Así, como las condiciones necesarias para PMD, para determinar si un valor regular es un candidato a ser minimizador local, se debe resolver el problema $A\mathbf{z} = \mathbf{b}$ donde A es una matriz $n \times p$, $\mathbf{b} = -F'(\bar{\mathbf{x}})$ y $\mathbf{z}$ es un vector columna de tamaño $p \times 1$ formado por los coeficientes λ_i y β_j; las columnas linealmente independientes de A son los gradientes de las desigualdades activas y de las igualdades evaluados en el valor regular.

Intuitivamente, la forma de resolver el problema anterior es resolver el sistema $A\mathbf{z} = \mathbf{b}$ y en caso de existir la solución comprobar los valores λ_i son no negativos. Sin embargo, el problema se puede resolver como un sistema de programación lineal

$$\begin{aligned} \text{mín}\, \mathbf{z} &= \mathbf{0^T}\boldsymbol{\lambda} + \mathbf{0^T}\boldsymbol{\beta} \\ A\begin{bmatrix}\boldsymbol{\lambda}\\ \boldsymbol{\beta}\end{bmatrix} &= \mathbf{b} \\ \boldsymbol{\lambda} &\geq \mathbf{0}, \end{aligned} \tag{3.9}$$

el cual es un problema estándar de programación lineal en donde las variables $\beta_j = \beta_j^+ - \beta_j^-$ se descomponen en dos variables no negativas y se efectúan los cambios correspondientes para usar un algoritmo de programación lineal con variables no negativas.

Las condiciones necesarias de KKT suministran criterios para determinar si un valor regular puede ser o no un minimizador, sin embargo, no garantiza si es o no un mínimo local, las condiciones suficientes de KKT, conocidas también como las condiciones de segundo orden, suministran un criterio para determinar si un posible minimizador es efectivamente un mínimo local [10].

Se enunciarán las condiciones de segundo orden para el problema PMDI, toda vez que éste es una generalización del problema PMD. Para la compresión del enunciado se listan las siguientes definiciones y notaciones:

- Una matriz cuadrada A, con entradas reales y simétrica ($A^T = A$) se dice *definida positiva* si
$$\mathbf{x}^T A\mathbf{x} > \mathbf{0} \;\text{ para todo } \mathbf{x} \neq \mathbf{0}.$$

- Para $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ una función doblemente diferenciable, la matriz simétrica $A_{n\times n} = a_{ij}$ con entradas
$$a_{ij} = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x})\right] \tag{3.10}$$
se conoce como la *matriz hessiana* y se denotará como $f''(\mathbf{x})$.

- Sean $\bar{\mathbf{x}}$ un punto que cumple las condiciones de primer orden para el problema PMDI, $\bar{\boldsymbol{\lambda}}$ y $\bar{\boldsymbol{\beta}}$ los multiplicadores de Lagrange asociados al punto $\bar{\mathbf{x}}$ y $\mathcal{L}''_{\bar{\mathbf{x}}}$ la siguiente matriz hessiana con respecto a la variable $\mathbf{x}$

$$\mathcal{L}''_{\bar{\mathbf{x}}} = \mathcal{L}''_{\bar{\mathbf{x}}}(\bar{x}, \bar{\lambda}, \bar{\beta}) = F''(\bar{\mathbf{x}}) + \sum_{i=1}^{m} \lambda_i g_i''(\bar{\mathbf{x}}) + \sum_{j=1}^{l} \beta_j h_j''(\bar{\mathbf{x}}). \tag{3.11}$$

- Sea $\bar{m}$ el número de desigualdades activas, es decir, el número de multiplicadores $\lambda_i > 0$. Sea M la matriz de tamaño $(\bar{m} + l) \times n$ cuyas filas son los gradientes de las desigualdades

activas y los gradientes de las restricciones de igualdad calculadas en $\bar{\mathbf{x}}$, considere $\bar{\mathcal{T}}$ el espacio nulo de M, es decir, $\bar{\mathcal{T}} = \{\mathbf{x} \in \mathbb{R}^n \mid M\mathbf{x} = \mathbf{0}\}$. Para el problema PMD, la matriz M podría no existir ($\bar{m} + l = 0$), en tal caso se considera que $\bar{\mathcal{T}} = \mathbb{R}^n$.

Condiciones suficientes de KKT para el PMDI [22] Sea $\bar{\mathbf{x}}$ un punto que satisface las condiciones de primer orden de KKT del PMDI, donde F, g_i, h_j son funciones doblemente diferenciales. Si $\mathcal{L}''_{\mathbf{x}}$ es definida positiva para todo $\mathbf{x} \in \bar{\mathcal{T}}$, entonces $\bar{\mathbf{x}}$ es un minimizador local.

Para garantizar que el minimizador local resulte ser un minimizador global, existen características geométricas sobre las funciones F, $\mathbf{g}$ y $\mathbf{h}$ que resultan condiciones suficientes, tales características van más allá de los objetivos específicos de este trabajo.

Implementación de KKT en un ordenador

Dado un problema de optimización no lineal de la forma (3.1), el método analítico para resolver el problema, consiste primero en determinar las soluciones factibles del sistema de ecuaciones generado por las condiciones de KKT de primer orden. Estas soluciones resultan ser los candidatos en donde puede ocurrir el mínimo global. Para determinar si el candidato es por lo menos un mínimo local, se debe determinar si este, junto con sus correspondientes multiplicadores, cumple las condiciones de KKT de segundo orden.

El método analítico, anteriormente descrito, resulta complejo cuando el problema de optimización es no lineal y/o hay un número considerable de restricciones. En tales casos, es común resolver el problema de optimización de forma numérica. El algoritmo generalmente implementado en los ordenadores es el método del gradiente descendiente, el cual consiste en partir de una iteración inicial $\mathbf{x}^{(0)}$ en la región factible y determinar si cumple las condiciones tanto de primer como de segundo orden de KKT. En caso de que $\mathbf{x}^{(0)}$ no es un mínimo local, se busca un nuevo candidato en la dirección negativa del gradiente de la función objetivo evaluada en $\mathbf{x}^{(0)}$. Hay varias metodologías para determinar el mejor candidato basado en las restricciones que posee el problema de optimización.

Como se mostró anteriormente, una forma equivalente de comprobar las condiciones de primer orden de KKT en un valor específico, es resolver los problemas de programación lineal (3.7) o (3.9). Basado en los tiempos computacionales de diversos algoritmos para resolver problemas de programación lineal, se ha probado que el método de barrera y el de punto interior son eficientes para sistemas con un gran número de variables [10]. El método de punto interior recurre a un subalgoritmo para resolver sistemas de ecuaciones, por lo tanto, antes de realizar una introducción al método de barrera y el de punto interior, se realiza una breve introducción del método de Newton.

Método de Newton Considere el siguiente sistema de ecuaciones:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0, \end{aligned} \tag{3.12}$$

donde cada f_i es una función de $\mathbb{R}^n$ en $\mathbb{R}$ diferenciable. El sistema (3.12) se denota de forma más simplificada como $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, donde $\mathbf{F}$ es un vector columna de tamaño n y $\mathbf{x}$ representa

las variables $(x_1, x_2, \ldots, x_n)$. La matriz de tamaño $n \times n$ cuyas entradas son

$$J_{ij} = \frac{\partial f_i}{\partial x_j}, \text{ para } 0 \leq i \leq n, \ 0 \leq j \leq n, \tag{3.13}$$

se conoce como la *matriz jacobiana* o el *jacobiano* asociado a $\mathbf{F}$. Utilizando la aproximación de Taylor de orden uno, el método de Newton consiste en calcular la sucesión

$$x^{(k+1)} = x^{(k)} - \left(J\left(x^{(k)}\right)\right)^{-1} \mathbf{F}\left(x^{(k)}\right), \text{ para } k \in \mathbb{N}, \tag{3.14}$$

a partir de un punto inicial $\mathbf{x}^{(0)} = (x_1^0, x_2^0, \ldots, x_n^0)$, en donde el J^{-1} representa la matriz inversa del jacobiano, en caso de que esta exista. Cuando la sucesión $x^{(k)}$ converge, indica que el valor de convergencia es una solución del sistema (3.12). Fijado un valor de tolerancia ε, un criterio para detener el algoritmo es realizar la iteración hasta que se cumpla que $||x^{(k+1)} - x^{(k)}|| < \varepsilon$ para algún $k \in \mathbb{N}$.

Método de barrera En un problema de optimización lineal o no lineal, la estrategia del método de barrera consiste en incluir las restricciones de desigualdad en la función a optimizar. Partiendo de puntos que son factibles pero que no pertenecen a la frontera de la región factible generada por las desigualdades, el método de barrera prioriza los puntos que no pertenecen a la frontera de la región generada por las restricciones de igualdad. Sin pérdida de generalidad, se puede suponer que el método de barrera se aplica a problemas de optimización de la siguiente forma:

$$\text{mín } F(\mathbf{x})\mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \tag{3.15}$$

donde $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \mathbf{g_2}(\mathbf{x}), \cdots, \mathbf{g_m}(\mathbf{x}))$, F es una función a valor real y se supone que el conjunto factible tiene interior no vacío, es decir,

$$\mathcal{A} = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}) < \mathbf{0}\} \neq \emptyset. \tag{3.16}$$

Se define una función barrera B que aumente su valor cuando los puntos están cerca de la frontera

$$B(\mathbf{x}) = \sum_{i=1}^{m} \chi(g_i(\mathbf{x})) \tag{3.17}$$

donde la función es cualquiera con las siguientes propiedades:

- $\chi(t)$ para $t \in (-\infty, 0)$ continua y estrictamente creciente.
- $\chi(t) \to +\infty$ cuando $t \to 0^-$

Ejemplos de funciones que cumplen las anteriores propiedades son

$$\chi(t) = -\frac{1}{t}, \chi(t) = -\log(-t). \tag{3.18}$$

El método de barrera consiste en minimizar la siguiente función

$$G(\mathbf{x}) = F(\mathbf{x}) + \mu B(\mathbf{x}), \tag{3.19}$$

en donde, algorítmicamente se arranca de un punto interior $\mathbf{x} \in \mathcal{A}$ y μ que es siempre positivo, disminuye en cada iteración [25].

Método de punto interior En la notación usada sobre métodos de punto interior es común denotar con letras minúsculas vectores columna en $\mathbb{R}^n$ y con letras mayúsculas matrices diagonales:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad X = \begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix}. \tag{3.20}$$

Además, $\mathbf{e}$ denota el vector columna de n unos:

$$\mathbf{e} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{3.21}$$

Considere el siguiente problema estándar de programación lineal

$$\begin{aligned} \text{mín } & \mathbf{c}^T\mathbf{x} \\ & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3.22}$$

donde A es una matriz $m \times n$, $m \leq n$, rango(A)=m, $\mathbf{c}$ vector columna $n \times 1$ y $\mathbf{b}$ vector columna $m \times 1$. Las condiciones de KKT de primer orden (3.8) para el problema (3.22) se pueden escribir de la forma

$$\mathbf{c} - I\boldsymbol{\beta} + A^T\boldsymbol{\lambda} = \mathbf{0}\boldsymbol{\beta} \geq \mathbf{0}\boldsymbol{\beta}^{\mathbf{T}}(-\mathbf{x}) = \mathbf{0} \tag{3.23}$$

donde I es la matriz identidad de tamaño n, $\boldsymbol{\lambda}$ es el vector de multiplicadores asociados a las igualdades y $\boldsymbol{\beta}$ el asociado a las desigualdades. Cambiando $\boldsymbol{\beta}$ por $\boldsymbol{s}$ y cambiando $\boldsymbol{\lambda}$ por $-\mathbf{y}$, las condiciones de factibilidad y las condiciones de KKT se pueden escribir de la siguiente manera:

$$\begin{aligned} A^T\mathbf{y} + \mathbf{s} - \mathbf{c} = \mathbf{0}, \ \mathbf{s} > \mathbf{0} \\ A\mathbf{x} - \mathbf{b} = \mathbf{0}, \ \mathbf{x} \geq \mathbf{0} \\ XS\mathbf{e} = \mathbf{0}. \end{aligned} \tag{3.24}$$

Los métodos de punto interior trabajan con puntos factibles que no están en la frontera de la región factible, es decir, trabajan con puntos que satisfacen la primera parte de la segunda ecuación de (3.24) con $\mathbf{x} > \mathbf{0}$. Por lo tanto, se requiere encontrar soluciones al sistema de ecuaciones (3.24) con $\mathbf{x} > \mathbf{0}$.

Considerando las variables $\boldsymbol{\xi} = (x, y, s) \in \mathbb{R}^{2n+m}$ en el sistema (3.24), al calcular la sucesión (3.14) desde el punto inicial $\boldsymbol{\xi}^{(0)} = (x^0, y^0, s^0)$ se obtiene

$$\boldsymbol{\xi}^{(k+1)} = \boldsymbol{\xi}^{(k)} - (\Phi'(\boldsymbol{\xi}^{(k)}))^{-1}\Phi(\boldsymbol{\xi}^k) \tag{3.25}$$

donde $\Phi(x, y, s)$ son todas las ecuaciones del lado izquierdo del sistema (3.24) y

$$\Phi'(x, y, s) = \begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \tag{3.26}$$

es su matriz jacobiana. Como el método de Newton no garantiza que a partir de un $\boldsymbol{\xi}^0 = (x^0, y^0, s^0)$ se obtenga un $\boldsymbol{\xi}^k = (x^k, y^k, s^k)$ que cumpla que $x^k > 0$ y $s^k > 0$, se aplica la siguiente variación en la forma recursiva

$$\boldsymbol{\xi}^{k+1} = \boldsymbol{\xi}^k + t_k \Delta\boldsymbol{\xi} t_k = \eta\ t_{max} \ \text{ con } 0 < \eta < 1 \tag{3.27}$$

donde $t_{max} = \text{máx}\{t \mid \boldsymbol{\xi}^k + t\Delta\boldsymbol{\xi} \geq 0\}$. El valor t_{max} hace que $\boldsymbol{\xi}^k + t_{max}\Delta\boldsymbol{\xi}$ sea un valor en la frontera, mientras el valor $t_k = \eta\ t_{max}$ hace que se obtenga un punto interior, usualmente η es un valor muy próximo a 1.

El paquete de software **IPOPT** (*Interior Point OPTimizer*), que principalmente usa algoritmos de punto interior, se ha consolidado como un paquete estable para resolver problemas de programación no lineal complejos o de gran escala. Al ser IPOPT de licencia pública, la mayoría de software disponibles en el mercado para resolver problemas de optimización, bien sea licenciados o libres, internamente recurren a esta librería [10].

3.1.2. Optimización dinámica

Un modelo de optimización como (3.1) se puede extender de forma más general a un modelo susceptible de ser controlado

$$\begin{aligned} &\min_{x(t),u} \ F(x(t), u, p) \\ &sujeto\ a: x' = f(x(t), u, p) \\ &\qquad g(x(t), u, p) \leq 0 \\ &\qquad h(x(t), u, p) = 0 \\ &\qquad x(t_0) = x_0, \end{aligned} \tag{3.28}$$

para $t \in [t_0, t_f]$, donde $x(t)$ es una función a valor real que depende implícitamente del tiempo t, t_0 es el tiempo en que se inicia el modelado, t_f el tiempo final, u y p son parámetros, x' representa la variación instantánea con respecto al tiempo, f es un vector de funciones a valor real y $x(t_0)$ son las condiciones iniciales de las variables de estado. Cuando en un problema de la forma (3.28) se permite que los parámetros u pasen a ser variables, incluso funciones dependientes del tiempo, éste modelo se conoce como un modelo de control óptimo, el que incluye ecuaciones algebraicas y diferenciales, también es conocido como un modelo DAE, por sus siglas en inglés. El objetivo, de este tipo de problemas, es determinar las trayectorias $x(t)$ y $u(t)$ que hacen que $F(x(t), u(t))$ sea mínimo y satisfagan las restricciones. La trayectoria $x(t)$ se conoce como la variable de estado y $u(t)$ la variable de control.

El modelo de optimización dinámica (3.28) muy pocas veces puede resolverse de forma explícita, siendo necesario recurrir a variadas técnicas matemáticas para la aproximación de su solución; generalmente, se recurre al método secuencial o al método simultáneo. El método secuencial resuelve la ecuación diferencial de (3.28) para posteriormente verificar si la solución obtenida satisface las restricciones, este proceso se repite varias veces modificando la variable de

control, para finalmente, entre todas las soluciones factibles, escoger la que mejor se aproxima al comportamiento deseado. En contraste, el método simultáneo, primero resuelve la ecuación diferencial de (3.28) usando algún método numérico, que transforman el sistema en un problema de la forma:

$$\begin{aligned} \min_{x(t)} \quad & F(x(t), u) \\ sujeto\ a: \ & g(x(t), u) \leq 0 \\ & H(x(t), u) = 0, \end{aligned} \tag{3.29}$$

en donde H contiene las restricciones de igualdad h y las soluciones a las ecuaciones diferenciales que ahora se convierten en restricciones de igualdad. Finalmente, al sistema (3.29) se le aplica una rutina de optimización.

En la práctica, para generar trayectorias en que la variable de control depende del tiempo, en ambos métodos, el tiempo se subdivide en intervalos y cada intervalo se soluciona progresivamente. Para sistemas con un gran número de variables, el método simultáneo ha demostrado ser una técnica eficiente desde el punto de vista computacional [12], ya que en éste método se requiere resolver sistemas de ecuaciones diferenciales ordinarias, a continuación se realiza una descripción detallada del método numérico Colocación ortogonal el cual también es eficiente computacionalmente para resolver ecuaciones diferenciales ordinarias.

Colocación ortogonal

Sea $y(t)$ una función por determinar para la siguiente ecuación diferencial ordinaria

$$\frac{dy}{dt} = f(y) \quad \text{para} \quad t_0 \leq t \leq t_f \tag{3.30}$$

con la condición inicial $y(t_0) = y_0$, donde f es una función conocida. Cuando la función f cumple ciertas características es posible garantizar la existencia y unicidad de la solución de la ecuación diferencial, sin embargo, en la mayoría de las ocasiones es imposible encontrar una representación algebraica de $y(t)$ en términos de funciones elementales. Una forma para aproximar la solución de la ecuación (3.30) es encontrar un polinomio que se aproxime a la solución.

Supongamos que se conocen valores aproximados de la solución de la ecuación (3.30) en $n+1$ nodos; usando métodos de interpolación, se podría encontrar un polinomio $P(t)$ de grado n que aproxime la solución $y(t)$. Supongamos que el intervalo $[t_0, t_f]$ se particiona de la siguiente forma:

$$t_0 < t_1 < \ldots < t_n, \tag{3.31}$$

si se aproximan los valores

$$(t_i, y(t_i)) \quad \text{para} \quad 1 \leq i \leq n \tag{3.32}$$

usando interpolación se podría determinar $P(t)$. Para determinar una aproximación de los valores $y(t_i)$, determinemos una matriz M que satisfaga el siguiente sistema de ecuaciones lineales

$$\begin{bmatrix} \frac{dy}{dt}(t_1) \\ \vdots \\ \frac{dy}{dt}(t_n) \end{bmatrix} = M_{n\times n} \begin{bmatrix} y(t_1) - y_0 \\ \vdots \\ y(t_n) - y_0 \end{bmatrix}. \tag{3.33}$$

Como se busca $P(t)$ tal que $\dfrac{dy}{dt}(t_i) \approx P'(t_i)$, se puede probar que

$$M_{n\times n} = \begin{bmatrix} 1 & 2t_1 & 3t_1^2 & \cdots & nt_1^{n-1} \\ 1 & 2t_2 & 3t_2^2 & \cdots & nt_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2t_n & 3t_n^2 & \cdots & nt_n^{n-1} \end{bmatrix} \begin{bmatrix} t_1 & t_1^2 & \cdots & t_1^n \\ t_2 & t_2^2 & \cdots & t_2^n \\ \vdots & \vdots & \ddots & \vdots \\ t_n & t_n^2 & \cdots & t_n^n \end{bmatrix}^{-1}, \tag{3.34}$$

así, la ecuación (3.33) provee una relación entre la derivada y la diferencia de la solución en los nodos con el valor inicial y_0. Por lo tanto, la ecuación (3.30) se puede escribir de la siguiente forma

$$\begin{bmatrix} 1 & 2t_1 & 3t_1^2 & \cdots & nt_1^{n-1} \\ 1 & 2t_2 & 3t_2^2 & \cdots & nt_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 2t_n & 3t_n^2 & \cdots & nt_n^{n-1} \end{bmatrix} \begin{bmatrix} t_1 & t_1^2 & \cdots & t_1^n \\ t_2 & t_2^2 & \cdots & t_2^n \\ \vdots & \vdots & \ddots & \vdots \\ t_n & t_n^2 & \cdots & t_n^n \end{bmatrix}^{-1} \begin{bmatrix} y(t_1) - y_0 \\ y(t_2) - y_0 \\ \vdots \\ y(t_n) - y_0 \end{bmatrix} = \begin{bmatrix} f(y(t_1)) \\ f(y(t_2)) \\ \vdots \\ f(y(t_n)) \end{bmatrix}, \tag{3.35}$$

lo cual generaría un sistema de n ecuaciones, posiblemente no lineales, en donde las incógnitas son los valores $y(t_i)$ para $1 \leq i \leq n$. Después de solucionar el anterior sistema, se tendría los n puntos $(t_i, y(t_i))$ para $1 \leq i \leq n$, necesarios para determinar el polinomio $P(t)$ de grado n que aproxima la solución $y(t)$.

Con los valores aproximados de las parejas ordenadas de la ecuación (3.32), la aproximación de la solución $y(t)$ se puede realizar usando interpolación de Lagrange, el cual genera un polinomio $P(t)$ de la forma

$$P(t) = \sum_{i=0}^{n} y(t(i)) L_{n,k}(t) donde L_{n,k}(t) = \prod_{\substack{n=0 \\ i\neq k}}^{n} \frac{(t - t_i)}{(t_k - t_i)}. \tag{3.36}$$

En total, el anterior método muestra cómo reduciendo la ecuación diferencial (3.30) a un sistema de ecuaciones y usando interpolación es posible encontrar una aproximación explícita de la solución de la ecuación diferencial.

La forma tradicional de elegir los nodos para realizar el proceso de aproximación de la ecuación diferencial ordinaria es seleccionar nodos equidistantes, sin embargo, los nodos pueden elegirse de una forma distinta de tal manera que la aproximación tenga mejores resultados tanto desde el punto de vista de la aproximación de la solución como del tiempo computacional empleado para determinar la aproximación. Cuando se aproxima la solución del problema (3.30) con un polinomio, la forma óptima de determinar este polinomio es seleccionar los nodos asociados a la cuadratura gaussiana. Realizar la selección de nodos sugerida por la cuadratura gaussiana y posteriormente realizar la interpolación se conoce como el método Colocación ortogonal de elementos finitos o sucintamente Colocación ortogonal. A continuación se realiza la explicación de qué es la cuadratura gaussiana, tomando como guía un conjunto particular de polinomios conocidos como familia de *polinomios ortogonales* [21].

Polinomios ortogonales Inicialmente se trabajará sobre el espacio de funciones polinómicas $P_N[-1,1]$ que está formado por todos los polinomios de grado menor o igual que N definidos en el intervalo $[-1,1]$, para posteriormente extender la teoría al espacio $P_N[0,\tau]$ con $\tau \in \mathbb{R}$ fijo, el cual es el espacio de interés para la aproximación de soluciones de ecuaciones diferenciales ordinarias con variable temporal.

Una familia de polinomios $R_1(t), R_2(t), \ldots, R_m(t)$ linealmente independiente es una base para $P_N[-1,1]$ si para cualquier polinomio $Q(t)$, existen constantes $u_i \in \mathbb{R}$ con $0 \leq i \leq m$ tales que

$$Q(t) = \sum_{i=0}^{m} u_i R_i(t). \tag{3.37}$$

Dada la función continua $\omega(x) > 0$ definida en $(-1,1)$, la familia de funciones $R_0(x), R_1(x), \ldots, R_m(x)$ se conoce como *familia de funciones ortogonales* si cumple

$$\int_{-1}^{1} \omega(t) R_i(t) R_j(t)\ dt = \delta_{ij}, \tag{3.38}$$

donde δ_{ij} es la constante uno cuando $i = j$ y es nula cuando $i \neq j$; la función $\omega(x)$ se conoce la *función de ponderación* o el peso.

Para una función de ponderación fija, si el conjunto de polinomios $\{R_0(t), R_1(t), \ldots, R_N(t)\}$ donde $R_i(t)$ tiene grado i, es una familia ortogonal, entonces el conjunto de polinomios resulta ser una base para el espacio $P_N[-1,1]$. Este resultado es de gran relevancia en el estudio de los espacios vectoriales, pues para probar algunas propiedades en todos los elementos del espacio, es suficiente con probarlas sobre una base [10].

Polinomios de Legendre La solución de la siguiente ecuación diferencial

$$\frac{d}{dt}\left[(1-t^2)\frac{d}{dt}Q_k(t)\right] + k(k+1)Q_k(t) = 0 \tag{3.39}$$

para cada $k \in \mathbb{N}$, genera un polinomio de grado k. El conjunto de soluciones $\{Q_0(t), Q_1(t), \ldots, Q_N(t)\}$ forman una base ortogonal para $P_N[-1,1]$ cuando la función de ponderación es $\omega(t) \equiv 1$. La anterior familia de funciones se conoce como los polinomios de Legendre, una fórmula explícita de estos se pueden generar mediante la siguiente expresión

$$Q_k(t) = \frac{1}{2^k}\sum_{i=0}^{k}\binom{n}{i}^2 (t+1)^{k-i}(t-1)^i. \tag{3.40}$$

Algunos de los polinomios de Legendre se encuentran en el siguiente cuadro:

Cuadratura Gaussiana Es conocido, desde la definición de integral de Riemann, que una forma para aproximar una integral es usar una partición equidistante en el intervalo $[a,b]$

$$\int_{a}^{b} f(t)\ dt \approx \sum_{i=0}^{n} f(t_i)\Delta t \tag{3.41}$$

Polinomios de Legendre en $[-1, 1]$	
Grado k	$Q_k(t)$
0	1
1	t
2	$\frac{1}{2}(3t^2 - 1)$
3	$\frac{1}{2}(5t^3 - 3t)$
4	$\frac{1}{8}(35t^4 - 30t^2 + 3)$
5	$\frac{1}{8}(63t^5 - 70t^3 + 15t)$

Cuadro 3.1: Cuadro de polinomios de Legendre

donde $\Delta t = \dfrac{b-a}{n}$, $t_i = a + i\Delta t$ para $0 \leq i \leq n$ y n es el número de particiones elegidas para el intervalo $[a, b]$. Sin embargo, la igualdad no siempre se tiene incluso para un polinomio de grado de grado n; la aproximación por curvatura gaussiana consiste en buscar una suma ponderada y una partición conveniente de tal forma que la integral coincida cuando la función $f(t)$ es un polinomio de cierto grado, más específicamente, se busca que

$$\int_a^b f(t)\ dt = \sum_{i=0}^{n} w_i f(t_i) \tag{3.42}$$

para $f(t)$ un polinomio de grado $2n + 1$, donde w_i para $0 \leq i \leq n$ son valores convenientes asociados a los nodos t_i de la partición del intervalo $[a, b]$. Sin pérdida de generalidad, como se mostrará posteriormente, el intervalo $[a, b]$ se puede considerar momentáneamente como el intervalo $[-1, 1]$ [22].

La cuadratura Gaussiana asociada a una familia de polinomios ortogonales $\{P_0(t), P_1(t), \ldots, P_n(t), \ldots\}$, es la partición del intervalo $[-1, 1]$ en los nodos $t_0 < t_1 < \ldots < t_n$ los cuales son las raíces del polinomio $P_{n+1}(t)$ y los valores $\omega_0 < \omega_1 < \ldots < \omega_n$ que son la solución única del siguiente sistema de ecuaciones

$$\sum_{i=0}^{n} \omega_i t_i^k = \int_{-1}^{1} w(t) t^k\ dt \quad \text{para} \quad k = 0, 1, \ldots, n \tag{3.43}$$

donde $\omega(t)$ es la función de ponderación asociada a la familia de polinomios.

Se puede probar que cuando $f(t)$ es un polinomio de grado menor o igual a $2n - 1$, si se usa la cuadratura gaussiana asociada a una familia de polinomios ortogonales, la ecuación (3.42) se cumple para $a = -1$ y $b = 1$. La anterior propiedad muestra que seleccionando adecuadamente una partición con n nodos internos se puede calcular de forma exacta una integral para polinomios de grado $2n - 1$, esta selección casi duplica la efectividad sobre polinomios en comparación con la aproximación de una integral por interpolación. Por lo tanto, si para aproximar una solución por interpolación del problema (3.30) se usa la partición sugerida por

la cuadratura gaussiana, en una partición de tamaño n se alcanzaría una mejor exactitud que si se realiza una partición equidistante de tamaño $2n$.

El método de la cuadratura gaussina se puede generalizar a cualquier intervalo $[a, b]$ teniendo en cuenta el siguiente cambio de variable

$$\int_a^b f(t)\ dt = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right)\ dt. \tag{3.44}$$

Así, la partición generada por la cuadratura gaussiana a partir de una familia de polinomios ortogonales, puede ser usada en los métodos de interpolación para aproximar soluciones a ecuaciones diferenciales ordinarias en $P_N[0, \tau]$ con $\tau \in \mathbb{R}$ fijo, lo cual mejora la aproximación en comparación con una interpolación que usa particiones equidistantes [10],[12],[25].

NMPC

Modelos como (3.28) pueden ser resueltos en cortos horizontes de tiempo, permitiendo la corrección de las variables de control con base en la predicción del estado del sistema en el horizonte de predicción fijado. Esta estrategia de solución, conocida como Modelo de control predictivo (MPC) o NMPC cuando el modelo incluye alguna función no lineal o restricciones no lineales [26], en cortos espacios de tiempo, compara el resultado real del proceso (X) con el estado predicho por modelo del proceso (X_p), buscando manipular parámetros del sistema (u) para que X alcance una trayectoria de referencia o *setpoint* satisfacciendo los requisitos de la ley de control. La diferencia entre los estados X_p y X, junto con la solución del modelo de optimización es alimentada a una restricción de sensibilidad, cuyo resultado son las condiciones de entrada que el proceso debe tener para alcanzar y/o mantener su punto de operación óptimo [27]. Las condiciones de entrada u son alimentadas al proceso y su modelo, quien predice el comportamiento en un horizonte de tiempo cerrando el ciclo de control [8],[28]. La figura 3.1 presenta, de forma gráfica el comportamiento descrito.

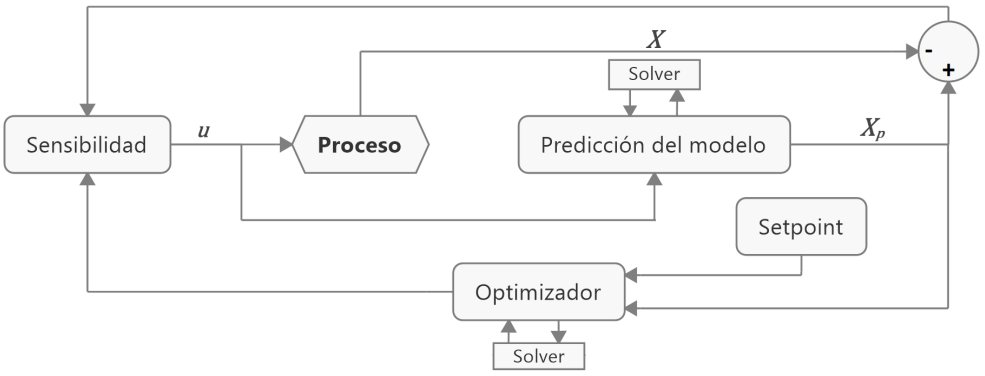

Figura 3.1: Estructura MPC

El sistema (3.28) generalmente no puede ser resuelto de forma explícita, por lo tanto, se fija un *horizonte de predicción* $[t_0, t_{N_p}]$ el cual se discretiza, generando un sistema de ecuaciones en

diferencias como el siguiente

$$\begin{aligned} \min_{x,u} \quad & \sum_{i=0}^{N_p} F(x_i, u_i) \\ sujeto\ a: \ & g(x_i, u_i) \leq 0 \\ & h(x_i, u_i) = 0 \\ & x(t_0) = x_0. \end{aligned} \tag{3.45}$$

para todo $i \in \{0, 1, \ldots, N_p\}$, donde se ha suprimido el parámetro p para facilitar la escritura y $t_{N_p} < t_f$. Así, el problema se resuelve secuencialmente en horizontes de predicción que en total completan el intervalo $[t_0, t_f]$, donde la condición inicial de cada horizonte de predicción depende del estado final del horizonte de predicción inmediatamente anterior. La transformación (3.45) es independiente de la partición usada en el intervalo, por ejemplo, se podría haber usado una partición equidistante o usado la cuadratura gaussiana asociada a alguna familia de polinomios ortogonales.

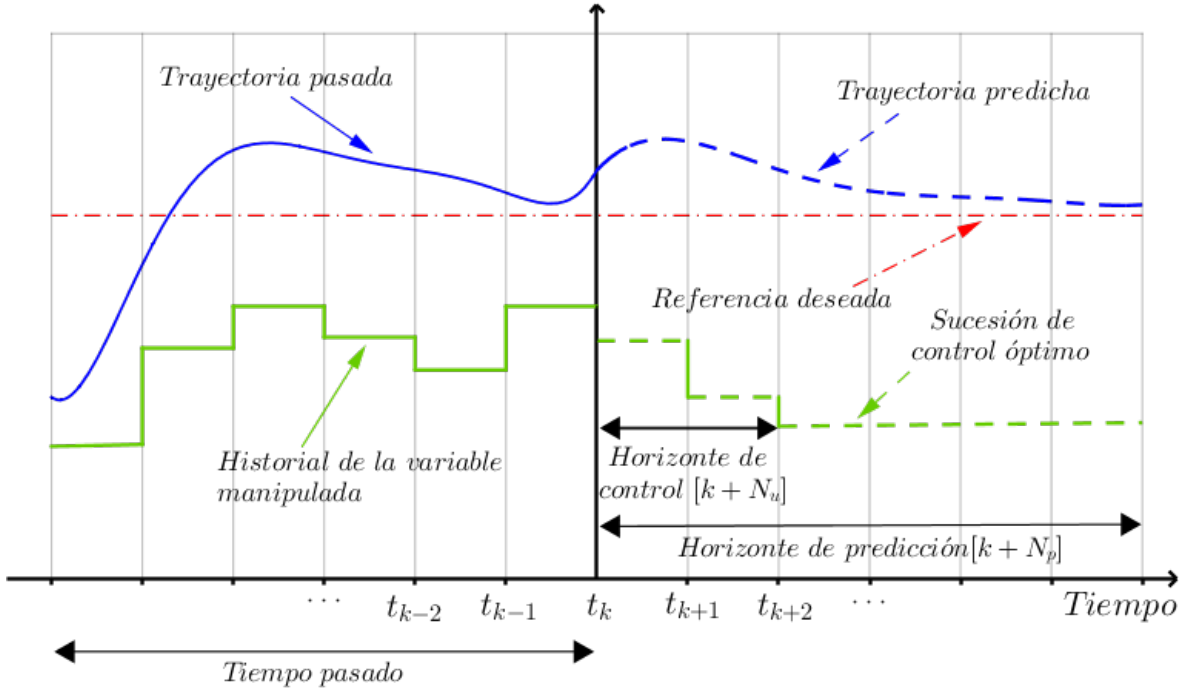

Figura 3.2: Esquema modelo de control

Para un horizonte de predicción fijo $[t_k, t_{k+N_p}]$, donde $t_{k+N_p} < t_f$, la figura 3.2 esquematiza la estrategia de solución realizada en cada instante de tiempo t_k. Los tiempos previos a t_k son el historial de solución de evolución del sistema. En el paso t_k la variable de control está fija, la cual está representada en la figura 3.2 con una línea continua, en este instante se resuelve el problema de optimización, lo cual genera una trayectoria predicha y unos parámetros de control, los cuales se representan con las líneas punteadas. Dentro del intervalo del horizonte de predicción, se fija el subintervalo $[t_k, t_{k+N_u}]$ llamado *horizonte de control*, con $N_u < N_p$, en este subintervalo u puede ser diferente en cada partición y a partir de t_{k+N_u} la variable u es constante.

Hasta aquí, sólo se ha esquematizado una estructura típica de discretización en un problema de control optimal, sin embargo, la técnica MPC o NMPC se diferencia, en que luego de predicho el instante de tiempo t_{k+1}, se repite el proceso anterior en el siguiente horizonte de predicción reajustando el parámetro de control. Determinar qué tanto se pueden modificar parámetros en un problema de optimización sin que se altere el valor en el cual se ha obtenido un óptimo se conoce como *análisis de sensibilidad*. Al introducir variables de control, puede pasar que restricciones inactivas pasan a ser activas y viceversa, estos cambios pueden afectar la precisión de la solución y la estabilidad del algoritmo de solución. Así que, basado en técnicas estándar de análisis de sensibilidad, se ha desarrollado una metodología que restringe la amplitud de las variaciones de las variables de control [18].

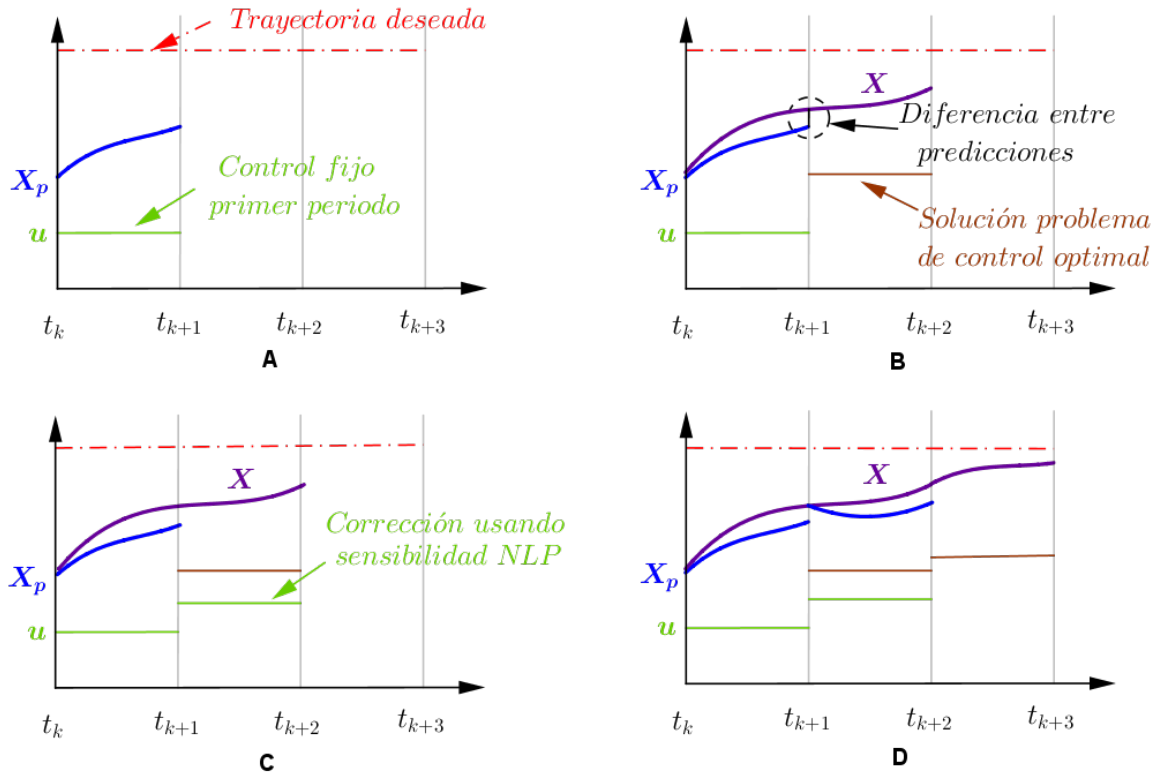

Figura 3.3: Esquema modelo de control predictivo

La figura 3.3 representa secuencialmente la estructura y el reajuste del proceso MPC en el horizonte de control. Para el instante t_k está fijo el valor de control u hasta t_{k+1}, para este valor fijo se resuelve la ecuación diferencial (figura 3.3 numeral a). Posteriormente, se resuelve el problema de optimización con restricciones de igualdad, donde se corrige la trayectoria y se ajusta la variable de control para el siguiente intervalo de tiempo (figura 3.3 numeral b). Para determinar la corrección de la variable de control, de tal manera que no altere el conjunto de restricciones activas, se soluciona el siguiente problema de optimización como se presenta en el numeral C. Finalmente, se reinicia el proceso de cálculo, mostrado en el numeral D.

$$\begin{aligned} \underset{x(t)}{max} \quad & \tau \\ sujeto\ a: \ & u_k + \tau \Delta u_k \leq u_{ub} \\ & u_k + \tau \Delta u_k \geq u_{lb} \\ & 0 \leq \tau \leq 1, \end{aligned} \tag{3.46}$$

donde u_k es el valor de control del tiempo t_k, u_{lb} y u_{ub} son los límites inferior y superior permitidos de la variación de las variables de control y Δu_k se extrae del sistema (3.24) en el tiempo t_k mediante una técnica de sensibilidad [18]. Ya determinado el valor τ se fija el valor de la variable de control en el segundo intervalo $u_{k+1} = u_k + \tau \Delta u_k$ (figura 3.3 inferior izquierda). Finalmente, todo el proceso se repite renombrando $k + 1 = k$.

3.1.3. Software de optimización dinámica

Con el paso de los años, diferentes paquetes de optimización dinámica cada vez resuelven en menos tiempo sistemas no lineales con cientos de variables, entre los más conocidas se encuentran:

- AMPL: Es un sistema de modelado que integra lenguaje de comandos y scripts. Este incorpora gran variedad de solucionadores así como rápida diferenciación automática. Sus interfases están disponibles en C++, C#, Java, Matlab y Python [12].

- Pyomo: Empleado en el modelamiento y optimización. Permite la diferenciación y discretización automática de los sistemas DAE mediante la colocación ortogonal o la diferenciación finita. El método Colocación ortogonal está implementado con la familia de polinomios ortogonales de Legendre y Radau. Los problemas de programación no lineal (PNL) puede resolverse utilizando cualquiera de las varias docenas de solucionadores compatibles con la biblioteca de solucionadores de AMPL [12].

- ASL: Paquete empleado en modelamiento y optimización. Emplea colocación ortogonal de elementos finitos. La programación no lineal es resuelta por solucionadores AMPL [12].

- JuMP: Realiza optimización en lenguaje Julia. Este realiza la solución de problemas lineales, no lineales y mezcla de enteros por medio de diversos solucionadores [12].

- Casadi: Es una plataforma de permite realizar programación en bloques. No es un paquete de optimización como tal, este permite conectar los bloques para resolver la optimización dinámica. Está disponible para actuar con interfases como Matlab, Python y C++ [12].

- GAMS: Es un paquete para el modelamiento y optimización lineal y no lineal a gran escala con una amplia base de usuarios. Se conecta a una variedad de solucionadores comerciales y de código abierto, y las interfaces de programación están disponibles en Excel, MATLAB y R [12].

- gPROMS: Es un ambiente avanzado de modelado de procesos químicos con posibilidades de optimización. Contiene librerías de propiedades físicas. La optimización se implementa por método secuencial [12].

- JModelica:Es un paquete de optimización de fuente abierta basado en el lenguaje de programación Modelica. Este relaciona diferentes paquetes de fuente abierta. Los sistemas dinámicos son discretizados usando métodos de colocación ortogonal. Se accede a esta plataforma a través de Python [12].

- Gekko: Es un paquete diseñado en lenguaje de modelamiento algebraico que se extiende a optimización dinámica. Interactúa con solucionadores de código abierto, comerciales y personalizados para sistemas lineales, cuadráticos, no lineales y mixtos (LP, QP, NLP, MILP y MINLP). Su licencia del MIT, es de uso gratuito para aplicaciones académicas y comerciales. Estas características, entre otras, le permiten estabilidad computacional para sistemas de gran escala. Este paquete, incluye solucionadores como IPOPT, BPOPT, MINLP y APOPT [12].

Desde hace más de una década, el lenguaje Python se ha mostrado como una solución frente al software licenciado de alto costo, no necesita ser compilado y la facilidad en su programación lo ha llevado a crecer rápidamente. Cuenta con estructuras de datos eficientes y de alto nivel. Su sintaxis y la forma de escritura es ideal para *scripting* y desarrollo de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (tanto Unix, Mac o Windows) [29].

El MPC y el NMPC se ha implementado en Python con resultados satisfactorios [14], [16]. Entre las librerías anteriormente mencionadas que han implementado MPC y NMPC en lenguaje Python se encuentran Pyomo y Gekko. Con base en el número de variables del modelo propuesto a evaluar en este trabajo y los reportes bibliográficos relacionados con los resultados en desempeño computacional [21], la implementación en un paquete de programación se realiza en Gekko.

En el paquete Gekko, se puede recurrir a la implementación de algoritmos de optimización que usan métodos secuenciales o simultáneos, entre otros. El método simultáneo se recomienda en problemas con un gran número de variables, la implementación algorítmica de éste en el paquete Gekko, discretiza usando la cuadratura gaussiana asociada a la familia de polinomios de Lobatto, también convierte las restricciones de desigualdad en restricciones de igualdad usando un algoritmo del método de barrera.

Colocación ortogonal con polinomios de Lobatto

La solución de la siguiente ecuación diferencial

$$(1-t^2)\frac{d^2P_n(t)}{dt} - 4t\frac{dP_n(t)}{dt} + n(n+3)P_n(t) = 0 \tag{3.47}$$

para cada $n \in \mathbb{N}$, genera un polinomio de grado n. El conjunto de soluciones $\{P_0(t), P_1(t), \ldots, P_N(t)\}$ forman una base ortogonal para $P_N[-1,1]$ cuando la función de ponderación es $\omega(t) = (1-t^2)$. La anterior familia de funciones se conoce como los polinomios de Lobatto en el intervalo $[-1,1]$. Para generalizar la familia de polinomios al intervalo $[0,\tau]$, es suficiente con realizar un cambio de variable afín, $\widetilde{P}_i(t) = P_i((\tau+1)t-1)$ resulta ser una familia de polinomios ortogonales en el intervalo $[0,\tau]$. Los siguientes cuadros listan, respectivamente, algunos de los polinomios de Lobatto en el intervalo $[0,1]$ (ver cuadro 3.2), con sus raíces (ver cuadro 3.3)

Polinomios de Lobatto en $[0,1]$	
Grado n	$\widetilde{P_n}(t)$
0	1
1	$2t-1$
2	$5t^2-5t+1$
3	$14t^3-21t^2+9t-1$
4	$42t^4-84t^3+56t^2-14t+1$
5	$132t^5-330t^4+300t^3-120t^2+20t-1$

Cuadro 3.2: Cuadro de polinomios de Lobatto

Raíces de los polinomios de Lobatto en $[0,1]$	
Polinomio	Raíces
$\widetilde{P_1}(t)$	$\left\{\frac{1}{2}\right\}$
$\widetilde{P_2}(t)$	$\left\{\frac{1}{2}-\frac{\sqrt{5}}{10},\frac{1}{2}+\frac{\sqrt{5}}{10}\right\}$
$\widetilde{P_3}(t)$	$\left\{\frac{1}{2}-\frac{\sqrt{21}}{14},\frac{1}{2},\frac{1}{2}+\frac{\sqrt{21}}{14}\right\}$
$\widetilde{P_4}(t)$	$\left\{\frac{1}{2}-\frac{\sqrt{21}\sqrt{7+2\sqrt{7}}}{42},\frac{1}{2}-\frac{\sqrt{21}\sqrt{7-2\sqrt{7}}}{42},\frac{1}{2}+\frac{\sqrt{21}\sqrt{7-2\sqrt{7}}}{42},\frac{1}{2}+\frac{\sqrt{21}\sqrt{7+2\sqrt{7}}}{42}\right\}$
$\widetilde{P_5}(t)$	$\left\{\frac{1}{2}-\frac{\sqrt{33}\sqrt{15+2\sqrt{15}}}{66},\frac{1}{2}-\frac{\sqrt{33}\sqrt{15-2\sqrt{15}}}{66},\frac{1}{2},\frac{1}{2}+\frac{\sqrt{33}\sqrt{15-2\sqrt{15}}}{66},\frac{1}{2}+\frac{\sqrt{33}\sqrt{15+2\sqrt{15}}}{66}\right\}$

Cuadro 3.3: Cuadro de raíces de polinomios de Lobatto

Ahora, basados en la cuadratura gaussiana, se podría inferir que una selección adecuada de la partición del intervalo $[t_0,t_f]$ disminuiría el error de la aproximación. La selección de la familia de polinomios ortogonales de Lobatto es un método eficiente numéricamente para determinar la forma de realizar la partición del intervalo [12], más específicamente, los nodos de la partición (3.31) serán las raíces (ver cuadro 3.3) del polinomio $\widetilde{P_n}(t)$ el cual proviene de la familia de polinomios ortogonales de Lobatto.

Optimización dinámica por el método simultáneo en Gekko En este trabajo se evalúa el desempeño computacional de un proceso que se modela mediante un sistema de la forma (3.28), el cual se implementa usando el paquete Gekko en el modo de método simultáneo. La metodología de implementación interna en Gekko, se describe someramente a continuación.

Al definir el número de subintervalos de horizontes de predicción, el número de particiones en cada horizonte de control y la cantidad de particiones que se usan para el horizonte de control, internamente Gekko, en cada horizonte de predicción, realiza la partición gaussiana asociada a los polinomios de Lobatto para convertir las ecuaciones diferenciales en restricciones de igualdad.

Luego modifica la función objetivo, para mediante el método de barrera, convertir el problema de optimización en un problema con únicamente restricciones de igualdad. Posteriormente, usando el algoritmo del gradiente descendiente, verifica las condiciones de primer orden de KKT, que se convierten en problemas de programación lineal los cuales son resueltos mediante algoritmos de punto interior. A los que cumplen las condiciones de KKT de primer orden, luego se les verifica las condiciones de segundo orden usando matrices hessianas.

Cuando se ha realizado la optimización, la cual resulta factible en el horizonte de predicción, se compara la solución con el resultado real del proceso y se aplica el método NMPC con su correspondiente corrección de sensibilidad, para posteriormente resolver el problema de optimización asociado al siguiente horizonte de predicción que usa como condición inicial de las ecuaciones diferenciales ordinarias el estado en el tiempo final del inmediatamente anterior horizonte de predicción. Este proceso se repite hasta completar el tiempo t_f. Aunque el método de solución, algorítmicamente, resulta intrincado, Gekko facilita la programación e internamente automatiza estos procesos [12].

Capítulo 4

Caso de Estudio - Columna de destilación

El proceso de destilación ha sido estudiado en los últimos 60 años [30]. Esta es una operación unitaria de amplio uso a nivel industrial por la cual se logran separaciones binarias o multicomponentes a través de la diferencia en los puntos de ebullición relativos de sistemas líquido - vapor [6]. Aplicado a soluciones en las que todos los componentes son relativamente volátiles, mediante el aprovechamiento del intercambio de masa y energía del sistema, los componentes más livianos, también denominado destilado, son separados en la zona superior de la columna. Por otra parte, los más pesados o fondos, se encuentran en la parte inferior de la misma.

4.1. Modelo Dinámico del Proceso

Diversos autores han estudiado la dinámica de esta operación unitaria [31], [32]. El desarrollo del modelo depende las suposiciones realizadas, entre las que se encuentran: Idealidad del modelo, balance de energía, cálculos hidráulicos, número de componentes entre otros.

A continuación, se presentan tres modelos que describen la dinámica de una columna de destilación. El primero, propuesto por Bequette et al [31] será denominado MODELO 1, posteriormente, se cuenta con el modelo propuesto por Seider et al [32] para sistemas con equilibrio ideal llamado MODELO 2 y, finalmente el mismo modelo de Seider et al considerando un equilibrio real, denominado MODELO 3.

4.1.1. Modelo 1

Este modelo, propuesto por Bequette [31], considera idealidad del sistema, no incluye balance energético ni emplea el modelo hidráulico. Divide la columna en 5 zonas; condensador, rectificación, alimentación, empobrecimiento y rehervidor. El flujo de vapor es constante a lo largo de la columna, el de líquido es el mismo encima del alimento y debajo de este se suma el flujo de alimento, este modelo es aplicable a sistemas con máximo 3 componentes, requiere 2CP +4 ecuaciones, siendo C el número de componentes y P el número de platos. La figura 4.1 presenta las variables involucradas en el modelamiento de la columna.

Este sistema se compone de las siguientes ecuaciones:

1. Balance por componente: Aplicada para los componentes i -1 por plato j.

$$\frac{M_j dx_{i,j}}{dt} = LX_{i,j-1} + VY_{i,j+1} - LX_{i,j} - VY_{i,j}, \tag{4.1}$$

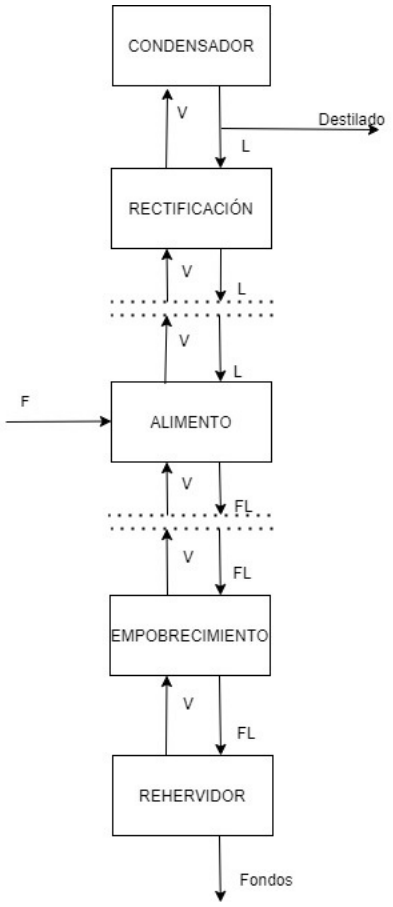

Figura 4.1: Modelo de una columna de Destilación de acuerdo con Bequette et al. 1998

M_j definido como el volumen del plato por la densidad de la mezcla, también es conocido como el *hold up* del sistema.

2. Ecuación de Equilibrio: Aplicada para los componentes i -1 por plato j.

$$\alpha_{i,j} = \frac{\frac{Y_{i,j}}{X_{i,j}}}{\frac{Y_{i+1,j}}{X_{i+1,j}}} \tag{4.2}$$

donde $\alpha_{i,j}$ corresponde a la volatilidad relativa del compuesto i en el plato j.

3. Ecuacion de Suma: Aplicada para cada uno de los platos j para los C componentes del sistema.

$$1 = \sum_{i=1}^{C} Y_{i,j} \tag{4.3}$$

$$1 = \sum_{i=1}^{C} X_{i,j} \tag{4.4}$$

Este modelo ha sido empleado en estudios de MPC como el generado por Safdarnejad et al [33].

4.1.2. Modelo 2

Este es un modelo más complejo que considera las ecuaciones M.E.S.H por sus siglas en inglés que indican Masa, Equilibrio, Suma y Entalpía. Este modelo describe la columna de destilación con más ecuaciones diferenciales, considera sistemas multicomponente, diferentes modelos de equilibrio y, principalmente, el cambio en el flujo de vapor y líquido a lo largo de la columna, representando de forma más real el sistema a separar. Este modelo se compone de P(2C+3) ecuaciones, más las ecuaciones de entalpía y equilibrio, donde C corresponde al número de componentes y P a los platos.

La figura 4.2 presenta las variables involucradas en el modelamiento de la columna de acuerdo a lo presentado por Seider et al [32]

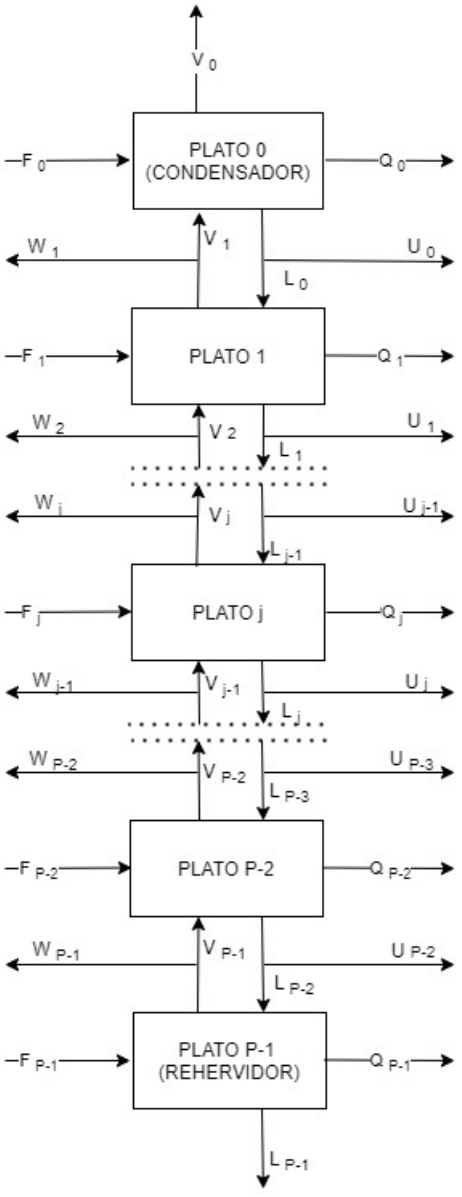

Figura 4.2: Diagrama de una Columna de Destilación de acuerdo con Seader et al.2010.

Este sistema se compone de las siguientes ecuaciones:

1. Balance por componente: Aplicada para cada uno de los componentes i por plato j.

$$\frac{M_j dx_{i,j}}{dt} = L_{j-1}X_{i,j-1} + V_{j+1}Y_{i,j+1} - (L_j + U_j)X_{i,j} - (V_j + W_j)Y_{i,j} \tag{4.5}$$

2. Ecuación de Equilibrio: Descrita como una condición estática donde, con el tiempo no ocurre cambio alguno en las propiedades macroscópicas de un sistema. Las ecuaciones de Raoult y Raoult modificado describen la condición de equilibrio para los sistemas líquido-vapor.

Ecuación de Raoult

Fundamentada en las suposiciones que la fase vapor es un gas ideal y la fase líquida es una solución ideal, es aplicable a sistemas con presiones bajas o moderadas [34], sistemas con sustancias químicamente semejantes, por ejemplo isómeros ($orto, meta, para$) o series homologas como n-hexano / n-heptano o etanol/propanol o benceno/tolueno entre otros. Este modelo de Raoult es presentado por medio de la ecuación (4.6) para cada uno de los componentes i por plato j.

$$Y_{i,j}P_j = X_{i,j}P_{i,j}^{sat} \tag{4.6}$$

donde $X_{i,j}$ es la fracción mol de la fase líquida, $Y_{i,j}$ es la fracción mol de la fase vapor y $P_{i,j}^{sat}$ es la presión de vapor de las especies puras i en el plato j a la temperatura del sistema. Esta es calculada por medio de la ecuación de Antoine. El producto $Y_{i,j} * P$ en el lado izquierdo de la ecuación (4.6) se conoce como la presión parcial de la especie i en el plato j [34]. Proporciona una descripción realista del comportamiento del sistema para algunos fluidos. No obstante, para sistemas más complejos se considera útil como un modelo de comparación únicamente.

La ecuación (4.6) puede ser reescrita de la siguiente forma:

$$Y_{i,j} = K_{i,j}X_{i,j}, \tag{4.7}$$

siendo $K_{i,j}$, que corresponde a la constante de equilibrio igual a:

$$K_{i,j} = P_{i,j}^{sat}/P \tag{4.8}$$

Esta constante es una medida de la "ligereza" de la especie componente, es decir, de su tendencia a favorecer la fase vapor. Cuando K_i es mayor a la unidad, la especie i presenta una mayor concentración en la fase vapor. Por el contrario, si K_i es menor a uno, se considera un fluido "pesado" presentándose en mayor concentración en la fase líquida [34].

3. Ecuación de Suma: Aplicada para cada uno de los platos j.

$$1 = \sum_{i=1}^{C} Y_{i,j} \tag{4.9}$$

$$1 = \sum_{i=1}^{C} X_{i,j} \tag{4.10}$$

4. Balance de Energía: Se realiza por medio de entalpía, propiedad intrínseca que es una función de la temperatura, la composición y el estado de agregación del sistema. Para el sistema líquido - vapor, se requieren las entalpias en cada uno de estos estados. Las ecuaciones

empleadas en el cálculo de estas son:

a. Cálculo de la entalpía del Vapor

$$H_{vapor\ i,j} = H_{formacion\ i} + (A_i T_j + B_i)(T_j - T_{ref}) \tag{4.11}$$

Donde A_i y B_i son constantes de interpolación.
b. Cálculo de la entalpía de Vaporización.

$$H_{vaporizacion\ i,j} = A_i(1 - Tr_i)^{(B_i + C_i Tr_i + DTr_i 2 + E_i Tr_i 3)} \tag{4.12}$$

Donde Tr_i es definido como T_j/Tc_i y A_i, B_i, C_i, D_i y E_i son parámetros de cada componente. Esta ecuación es obtenida de Aspen Properties.
c. Cálculo de la entalpía del líquido

$$H_{liquido\ i,j} = H_{vapor\ i,j} - H_{vaporizacion\ i,j} \tag{4.13}$$

d. Cálculo de la entalpía de la mezcla
La entalpía de mezcla es una propiedad dependiente de las composiciones presentes en cada plato. Para cada fase, esta es calculada por medio de las ecuaciones (4.14) y (4.15).

$$HV_j = \sum_{i=1}^{C} Y_{i,j} H_{vapor\ i,j} \tag{4.14}$$

$$HL_j = \sum_{i=1}^{C} X_{i,j} H_{lquido\ i,j} \tag{4.15}$$

Finalmente, se realiza el balance energético aplicado para cada uno de los platos j.

$$\frac{dU_{Interna\ j}}{dt} = L_{j-1} HL_{j-1} + V_{j+1} HV_{j+1} - (L_j + U_j) HL_j - (V_j + W_j) HV_j + Q_j \tag{4.16}$$

Para este documento, el lado izquierdo de la ecuación (4.16) fue considerado igual a cero.

4.1.3. Modelo 3

Este último modelo emplea las ecuaciones M.E.S.H y dentro del equilibrio considera sistemas reales donde aplica la ecuación de Raoult modificada.

Ecuación de Raoult Modificada

La ecuación de Raoult modificada (4.17), no considera idealidad en la fase líquida, por tanto, se adiciona un coeficiente de actividad γ_i en la ecuación de Raoult, quedando esta expresada de la siguiente manera:

$$Y_{i,j} P = X_{i,j} \gamma_i P^{sat}_{i,j} \tag{4.17}$$

La ecuacion (4.17) puede ser reacomodada y presentada de la siguiente forma:

$$Y_{i,_j} = K_{i,j} X_{i,j} \tag{4.18}$$

Siendo $K_{i,j}$ igual a:

$$K_{i,j} = \gamma_i P_{i,j}^{sat} / P \tag{4.19}$$

Este coeficiente de actividad γ_i, se obtiene a partir de información experimental y se representa en ecuaciones como: Margules, Van Laar, Wilson, NRTL o UNIQUAC. La selección de la ecuación adecuada es función de las características químicas de los componentes [34]. Para este estudio, se empleará el modelo NRTL.

Ecuación NRTL

En particular, la ecuación NRTL por sus siglas en inglés, *Non Random Two Liquids*, basado en las contribuciones moleculares y las interacciones que brindan los diferentes componentes, es utilizado desde 1968 por Renon[35]. Esta, utilizada en equilibrios líquido-líquido y líquido-vapor, representa adecuadamente sistemas no ideales y mezclas parcialmente miscibles. [35].

Las ecuaciones (4.20), (4.21), (4.22), (4.23), (4.24) y (4.25) asociadas a este modelo, para cada uno de los componentes i por plato j, por interacción k [35] son:

$$Ln(\gamma_{i,j}) = \frac{C_{i,j}}{S_{i,j}} * \sum_{i=1}^{C} X_{i,j} * \epsilon_{i,j,k} \tag{4.20}$$

$$\epsilon_{i,j,k} = \frac{G_{i,j,k}}{S_{i,j}} \left(\tau_{i,j,k} - \frac{C_{j,k}}{S_{i,j}}\right) \tag{4.21}$$

$$C_{i,j} = \sum_{i=1}^{C} X_{i,j} G_{i,j,k} \tau_{i,j,k} \tag{4.22}$$

$$S_{i,j} = \sum_{i=1}^{C} X_{i,j} G_{i,j,k} \tag{4.23}$$

$$G_{i,j,k} = exp(-\alpha_{i,j,k} \tau_{i,j,k}) \tag{4.24}$$

$$\tau_{i,j,k} = a_{i,j,k} + \frac{b_{i,j,k}}{T_j} \tag{4.25}$$

Por complejidad numérica, estas ecuaciones pueden ser reemplazadas por un polinomio que relacione el equilibrio del sistema tal como es presentado por Safdarnejad et al [33]. Este polinomio relaciona el valor de γ_i, como una función de las composiciones y la temperatura en fase líquida. La ecuación 4.26 presenta el modelo del polinomio empleado.

$$\gamma_i = AT_i^2 + BX_iT_i + CX_i^2 + D \tag{4.26}$$

Donde A, B, C y D son constantes de interpolación para cada componente i.

4.2. Variables asociadas al control de una columna de destilación

La operación y control de una columna de destilación depende de las variables que puedan ser manipuladas, controladas y cuales consideradas perturbaciones. Estas, en conjunto interfieren el costos de energía, materia prima y en general en los costos de operación [36]. La figura 4.3 presenta estas y su ubicación dentro del proceso, donde, las azules corresponden a las variables controladas, las rojas a manipuladas y moradas a perturbaciones. Una o varias variables pueden ser controladas y manipuladas por medio del MPC, siendo la composición en cima una de las más importantes [36].

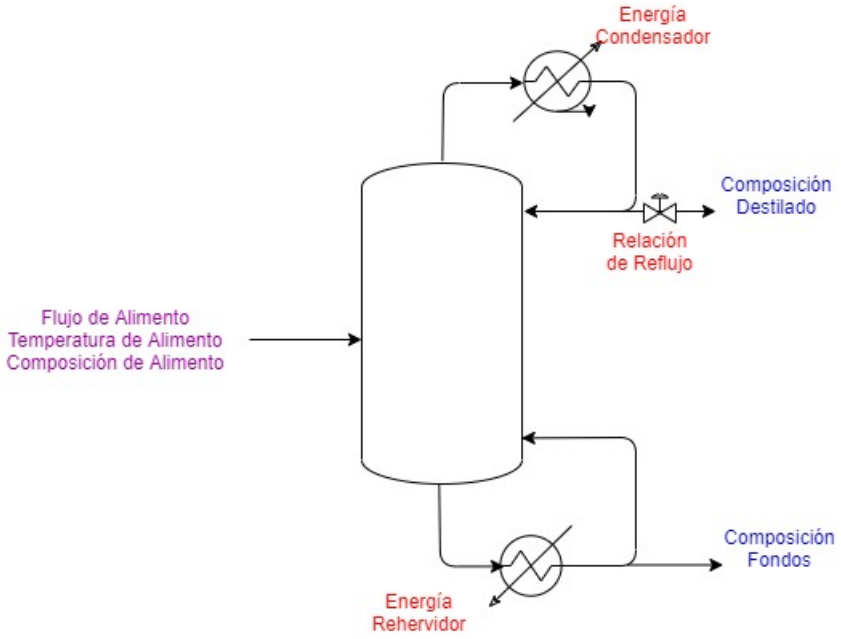

Figura 4.3: Relación de Variables Controladas, Manipuladas y perturbaciones en una Columna de Destilación

Por su parte, la relación de reflujo es la variable manipulada que más influencia tiene sobre la composición [36]. Energía del Rehervidor y del Condensador, consideradas en el MODELO 2 y 3 tienen efecto en la composición, no obstante, el uso óptimo de estas variables puede ser descrito por medio de e- NMPC (Economic NMPC) no estudiado en este documento.

Capítulo 5

Diseño Metodológico

El conjunto de acciones necesarias para llevar a cabo los objetivos de este documento son las siguientes:

5.1. Definición del modelo dinámico

La definición de este debe representar con suficiente exactitud el proceso físico. Para este estudio, se cuentan con tres diferentes modelos de proceso con distinta complejidad numérica que permiten evaluar el esfuerzo de la tarjeta Raspberry Pi en la resolución de cada uno.

Se eligieron tres casos de estudio; El Sistema Etanol - Agua, caracterizado por su alto uso a nivel industrial y la presencia de un azeótropo, lo que lo convierte en un sistema de interés desde el punto de vista numérico. También se analizaron los sistemas Benceno - Tolueno y Benceno - Tolueno - *p*- Xyleno, los que presentan comportamiento ideal [34].

Los tres modelos donde se analizó el sistema Etanol - Agua fueron configurados con las mismas condiciones de operación, presentadas en el cuadro 5.1.

Característica	Valor
Número de Componentes	2
Número de Platos	8 (incluyen condensador y rehervidor)
Plato de Alimentación	5
Flujo de Alimento	100 kmol/min
Composición del Alimento	Equimolar
Temperatura del Alimento	352 K
Relación de Reflujo	1.1656
Tipo de Condensador	Total

Cuadro 5.1: Características de la Columna de Destilación para el sistema Agua - Etanol

Las condiciones de operación del sistema Benceno - Tolueno (que de ahora en adelante se denominará BT) y del sistema Benceno-Tolueno - *p*- Xyleno (que de ahora en adelante se denominará BTX) son presentadas en el cuadro 5.2.

Característica	Sistema BT	Sistema BTX
Número de Componentes	2	3
Número de Platos(incluyen condensador y rehervidor)	13	13
Plato de Alimentación	5	5
Flujo de Alimento	100 kmol/min	100 kmol/min
Composición del Alimento	Equimolar	0.4/ 0.3/0.3
Relación de Reflujo	1.1656	1.1656
Tipo de Condensador	Total	Total

Cuadro 5.2: Características de la Columna de Destilación para los sistemas BT y BTX

El cuadro 5.3 presenta los parámetros físicos utilizados en la simulación del sistema Etanol - Agua.

Propiedad	Unidades	Etanol	Agua
Coeficientes de la Presión de Vapor	Bar	5.37229, 1670.409, -40.191	4.6543, 1435.264, -64.848
Temperatura Critica	K	513.9	647.1
Coeficientes Entalpía de Vapor	kJ/kmol	-235100, 0.1542, 13.148	-241818, 0.0049, 30.712
Coeficientes Entalpia de Vaporizacion	kJ/kmol	65831, 1.1905, -1.7666, 1.0012,0	56600, 0.61204, -0.6257, 0.3988, 0
Constantes A NRTL	Adimensional	0, -0.8009, 0	3.4578, 0, -1.2515
Constantes B NRTL	Adimensional	0, 246.18, 442.713	-586.08, 0, 272.608
Constantes C NRTL	Adimensional	0, 0.3, 0.3	0, 0, 0.3

Cuadro 5.3: Propiedades Físicas del Sistema

Como se mencionó en el capitulo 4, el equilibrio termodinámico puede ser modelado por medio de un polinomio. En este estudio, se modeló el coeficiente γ para cada uno de los componentes, encontrándose coeficientes de correlacción mayores a 0.99. Para el caso del Etanol, el polinomio obtenido fue:

$$\gamma_{Etanol} = 2{,}42828214E{-}04T_i^2 - 9{,}70474987E{-}03X_{Etanol}T_i + 2{,}46892759X_{Etanol}^2 - 2{,}78888683E{+}01 \tag{5.1}$$

Para el agua, el polinomio obtenido fue:

$$\gamma_{Agua} = 1{,}68999117E{-}05T_i^2 - 7{,}96232319E{-}03X_{Agua}T_i + 1{,}12705944X_{Agua}^2 + 4{,}99974760E{-}01 \tag{5.2}$$

Una vez definido y programado el modelo, es necesario garantizar que este muestre convergencia en estado estable y modo dinámico. Para cumplir este objetivo, Safdarnejad et al [33], ha establecido el esquema presentado en la figura 5.1. Cabe destacar que esta actividad representa una proporción importante de la actividad de definición y programación. Requiere de conocimiento del proceso y habilidad para llegar a la convergencia del sistema [12].

5.2. Definición de la función objetivo

Esta corresponde al set point o trayectoria deseada. Para este estudio, la variable de interés del sistema Etanol - Agua es la composición de Etanol en cima. En el caso de los modelos 1 y 2, se estableció un setpoint de 0.8 molar. Para el modelo 3 se estableció un valor de 0.78 molar, considerando la presencia del azeótropo.

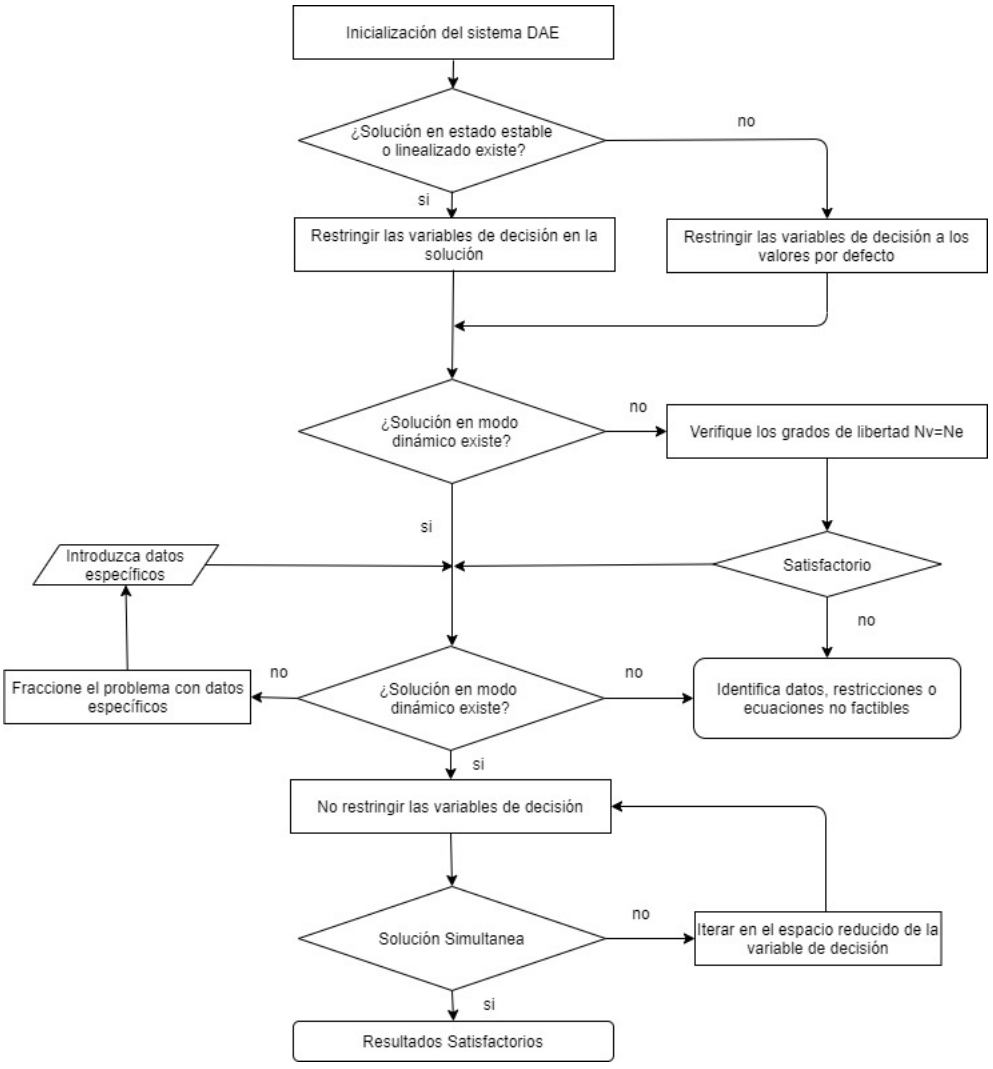

Figura 5.1: Estrategía de inicialización de sistemas DAE

Para los sistemas BT y BTX la variable de interés es la composición de Benceno en cima, el setpoint configurado fue de 0.83 molar.

Dentro de las variables manipuladas, se considera inicialmente la Relación de Reflujo, posteriormente la Energía del Rehervidor y, finalmente la Energía del Condensador. En este orden se realizó el analisis del desempeño computacional de la tarjeta.

5.3. Configuración del MPC en Gekko

La programación del modelo MPC es presentado en el cuadro 5.4

Propiedad	Valor
Número de nodos	3
Horizonte de Control	10 min
Horizonte de Predicción	80 min
Solver de optimización	IPOPT
Tolerancia del método	1 E-6

Cuadro 5.4: Condiciones de Configuración del MPC

Cabe resaltar que la discretización la realiza la libreria Gekko por medio de los polinomios de Lobatto de manera automática.

5.4. Configuración de la Tarjeta Raspberry Pi

Con el objetivo de minimizar consumos de CPU de la tarjeta Raspberry Pi, esta fue configurada en modo que pudiese manejarse por medio de líneas de comando y evitar consumos adicionales como interfases gráficas. El solver IPOPT es un solver de uso libre, para sistemas operativos Windows puede instalarse, no obstante, es demasiado pesado para sistemas Linux. Para este caso, Gekko tiene la posibilidad de utilizar este solver en modo remoto, requiriendo una conexión a Internet. El servidor empleado para correr el solver fue Dell R815, AMD Opteron Processor 6276, CPUs, 64 GB Ram, RAID array 15k RPM de disco duro.

5.5. Medición del esfuerzo computacional

Una vez configurados los tres modelos con las respectivas condiciones del MPC, se midió el tiempo computacional para 20 mediciones de cada simulación. Por medio de la herramienta *Solver Time* de Gekko y *Time*, biblioteca de Python.

Capítulo 6

Resultados y Análisis

Este análisis incluyó el estudio de 3 modelos dinámicos de una columna de destilación de platos con tres diferentes sistemas de componentes.

El cuadro 6.1 presenta, para mayor claridad, los códigos empleados en este estudio de acuerdo con el modelo y sistema utilizado.

Modelo	Sistema Estudiado	Característica	Ecuaciones en modo 3
1a	Etanol Agua	Sistema Ideal Manipula Relación de Reflujo	36
1b	BT	Sistema Ideal Manipula Relación de Reflujo	56
1c	BTX	Sistema Ideal Manipula Relación de Reflujo	82
2a	Etanol Agua	Ecuaciones MESH, ideal Manipula la Relación de reflujo	186
2b	Etanol Agua	Ecuaciones MESH, ideal Manipula la Relación de reflujo y energía del rehervidor	186
2c	Etanol Agua	Ecuaciones MESH, ideal Manipula la Relación de reflujo, energía del rehervidor y energía del condensador	186
3a	Etanol Agua	Ecuaciones MESH, sistema real Manipula la Relación de reflujo	186
3b	Etanol Agua	Ecuaciones MESH, sistema real Manipula la Relación de reflujo y energía del rehervidor	186
3c	Etanol Agua	Ecuaciones MESH, sistema real Manipula la Relación de reflujo, energía del rehervidor y energía del condensador	186

Cuadro 6.1: Resumen de los modelos y sistemas analizados

Vale la pena mencionar que el modelo 3, el cual representa un sistema real, fue evaluado por dos vías diferentes; Se programó el modelo NRTL de acuerdo a lo expuesto en la ecuaciones (4.20), (4.21), (4.22), (4.23), (4.24) y (4.25) y también se empleó una curva polinómica de 2 orden. El modelo obtenido con el polinomio se denomina modelo 3 y el obtenido por las ecuaciones de Prausnitz et al [35] se denomina modelo 4.

6.1. Programación en Gekko

Como se mencionó en el capitulo 2, Gekko es una librería diseñada para realizar optimización estática y dinámica de sistemas de gran escala. Esta tiene diferentes modos de operación entre los que se encuentran; cálculos en estado estable o modo 3, cálculos en estado dinámico o modo 4, MPC de manera simultanea o modo 6. Cuenta con más opciones de operación como MPC de manera secuencial, MHE por sus siglas en inglés *moving horizon estimation*, entre otros.

También tiene la posibilidad de utilizar variables intermedias que permiten disminuir la complejidad del modelo. Se almacenan en este pero no son reportadas al final. La figura 6.1 muestra un extracto de la programacion realizada y la forma en que las variables intermedias son declaradas.

Figura 6.1: Ejemplo de la programación y el uso de variables intermedias en Gekko

6.2. Validación de Resultados en estado estable

Siguiendo el esquema presentado en la figura 5.1, a continuación se presentan los resultados de la validación de los modelos en estado estático o modo 3 en Gekko. Esta validación se realizó contra Aspen Plus, software de simulación de procesos, reconocido a nivel industrial y académico. Se empleó el modelo de la columna de destilación *Radfrac* en todos los casos.

6.2.1. Sistema Etanol - Agua

Las figuras 6.2, 6.3, 6.6 y 6.7 presentan los diferentes perfiles en los casos en los que se considera sistema ideal.

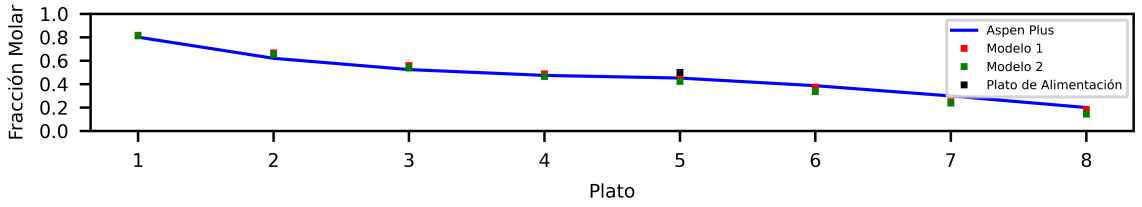

Figura 6.2: Perfil de composición de Etanol en fase líquida para los modelos 1a y 2a,b,c

Para el caso de la composición del agua en fase líquida, la figura 6.3 presenta los resultados en estado estable para cada plato.

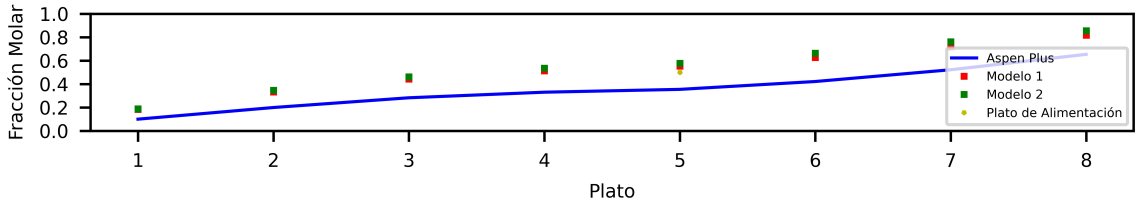

Figura 6.3: Perfil de composición de Agua en fase líquida para los modelos 1a y 2a,b,c

La figura 6.2, muestra alto grado de similitud para los modelos analizados validando su correcta programación. Sin embargo para el caso del agua no se encuentra el mismo resultado, exhibiendo diferencias en composición máximas de 0.1 fracción molar, principalmente en fondos. Si bien todos los modelos consideraron sistema ideal (Aspen Plus y modelos 1 y 2), Aspen no utiliza exactamente el mismo modelo que los propuestos en el documento de manera que puede llevar a estas desviaciones.

Las figuras 6.4 y 6.5 presentan los resultados de las simulaciones en estado estable para el modelo 3 y modelo 4. Estos modelos consideran sistema real al igual que la simulación en Aspen que en este caso empleó el modelo NRTL.

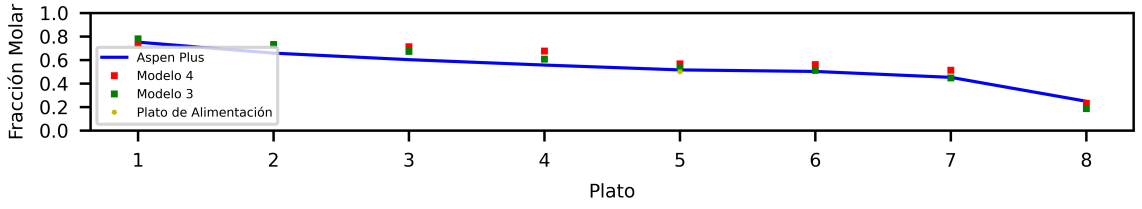

Figura 6.4: Perfil de composición de Etanol en fase líquida para los modelos 3 y 4.

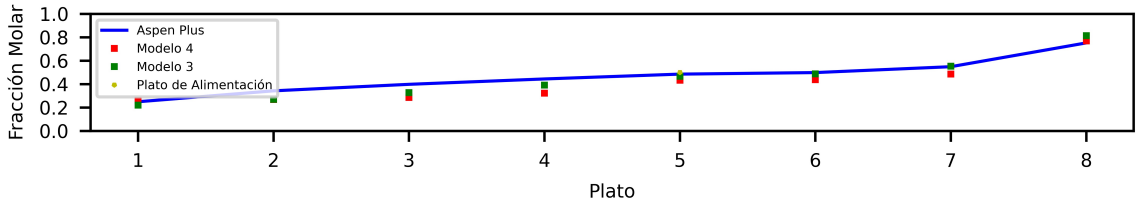

Figura 6.5: Perfil de composición de Agua en fase líquida para los modelos 3 y 4

Estas figuras describen con mayor precisión el perfil de composiciones en la columna. A diferencia de la figura 6.3, la figura 6.5 es más cercana a la simulación de Aspen, lo que es esperado, ya que los paramentros NRTL y la curva de aproximación fueron tomadas del simulador. El uso de un polinomio de aproximación muestra perfiles de composición muy similares al modelo con las ecuaciones de equilibrio validando correcta programación de las ecuaciones de equilibrio y del polinomio.

Las figuras 6.6 y 6.7 presentan los valores en estado estable para el flujo de líquido y vapor por cada plato para los 4 modelos evaluados.

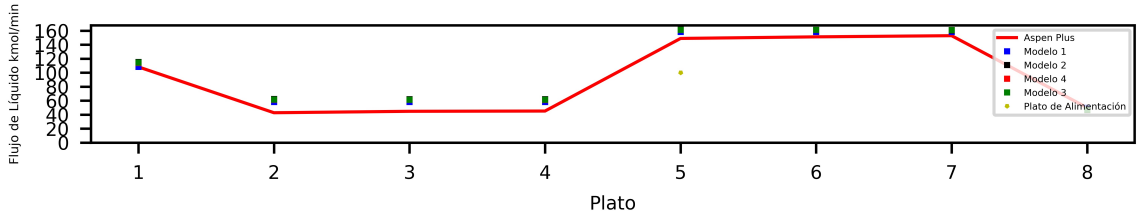

Figura 6.6: Perfil de flujo de Líquido para los modelos 1a, 2,3 y 4 para el sistema Etanol-Agua

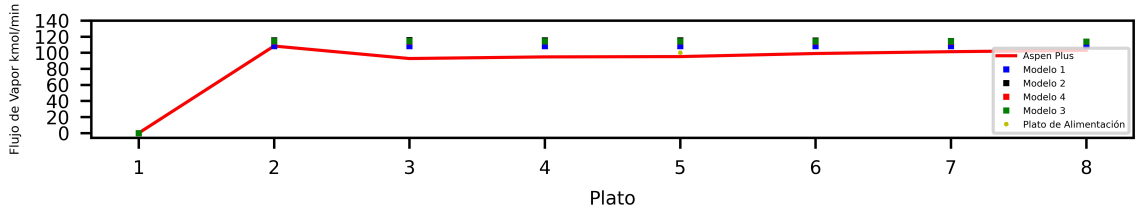

Figura 6.7: Perfil de flujo de Vapor para los modelos 1a, 2, 3 y 4 para el sistema Etanol- Agua

Los flujos de líquido y vapor en la columna presentan alto grado de similitud con relación a la simulación en Aspen, así como entre ellos. Cabe mencionar que el modelo 1a considera flujo de vapor constante con un sistema ideal mientras que los restantes lo discriminan por etapa, aplicado a sistemas ideales (modelo 2) y reales (modelos 3 y 4).

En general, los perfiles de composición y flujo para los 4 modelos analizados evidencian una adecuada programación de estos para el sistema Etanol - Agua.

6.2.2. Sistemas BT y BTX

Las figuras 6.8 y 6.9 presentan el comportamiento en estado estable del modelo 1b. Por su parte, las gráficas 6.10, 6.11 y 6.12 muestran los resultados obtenidos para el modelo 1c también presentadas en [37].

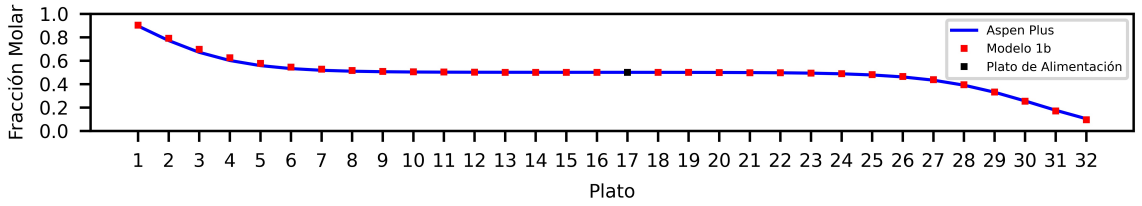

Figura 6.8: Perfil de composición de Benceno en fase líquida para el modelo 1b

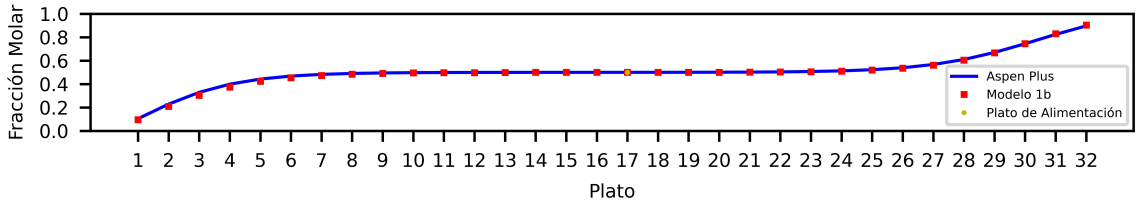

Figura 6.9: Perfil de composición de Tolueno en fase líquida para el modelo 1b

Las figuras 6.8 y 6.9 describen con mayor grado de precisión el perfil de composición a lo largo de la columna que el modelo Etanol - Agua . Si bien el modelo 1a y 1b son el mismo y lo único que cambia son los componentes, el sistema BT es un sistema con mayor grado de idealidad [34] por su similitud química.

Las figuras 6.10, 6.11 y 6.12 presentan los perfiles de composición para el modelo 1c. A diferencia del modelo 1b, este considera un componente adicional al sistema, en este caso *p* Xyleno.

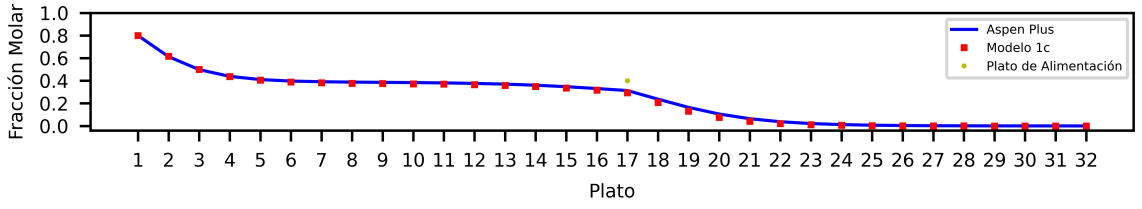

Figura 6.10: Perfil de composición de Benceno en fase líquida para el modelo 1c

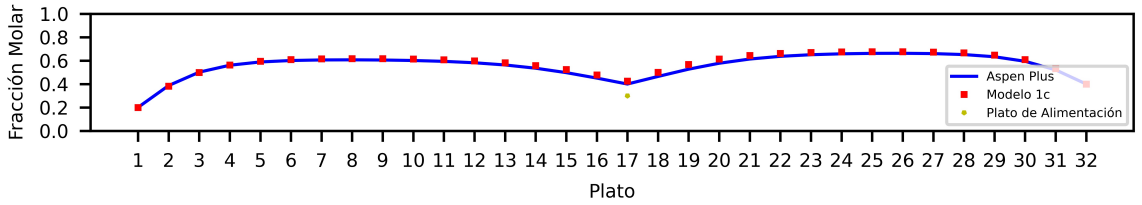

Figura 6.11: Perfil de composición de Tolueno en fase líquida para el modelo 1c

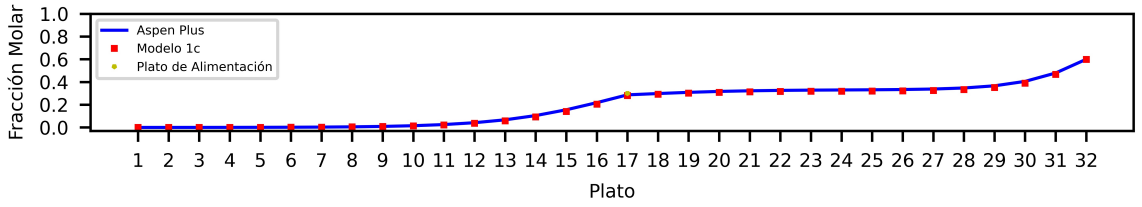

Figura 6.12: Perfil de composición de p-Xyleno en fase líquida para el modelo 1c

Al igual que en el caso 2a, el sistema es descrito con alto grado de similitud, validando lo encontrado con relación al tipo de componentes y, adicionalmente el uso del modelo 1 a sistemas con 3 componentes.

6.3. Resultados del NMPC

Una vez validado el comportamiento en estado estable, a continuación, se presenta la predicción del MPC para los 9 casos. Esta predicción se muestra por medio de las curvas de error, definido como SP - PV y el cambio en la relación de reflujo en la ejecución del NMPC para una sola iteración. Estos resultados se presentan de acuerdo con el número de variables manipuladas para cada modelo empleado.

6.3.1. Sistema Etanol - Agua

Este sistema fue evaluado por medio de los tres modelos anteriormente descritos. La figura 6.13 presenta el comportamiento obtenido en el horizonte de control para los casos 1a, 2a, 3a y 4 donde únicamente se manipuló la relación de reflujo.

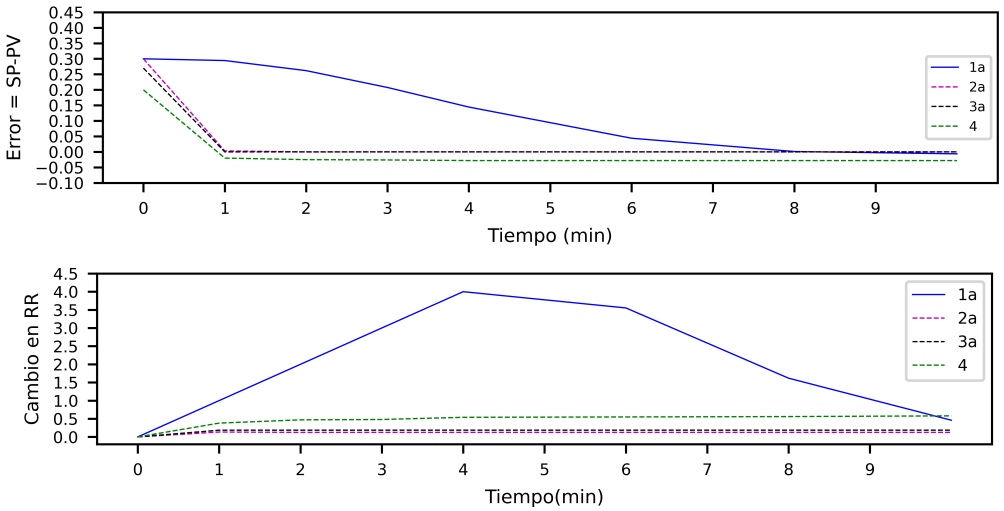

Figura 6.13: Comportamiento del error y la Relación de reflujo para los 4 modelos en un sistema donde sólo se manipula una variable

Esta figura presenta varios resultados en cuanto al movimiento de las variables controladas y manipuladas, tambien el error para cada caso y la magnitud de cambio en la relación de reflujo. Se encuentra que el modelo 1a presenta mayor tiempo para llegar al SP que los otros modelos, así mismo, la variación en el relación de reflujo es hasta 8 veces más grande que en los casos restantes. Mientras los modelos 2a, 3a y 4 presentan una variación ascendente en la relación de reflujo, el modelo 1a tiene cambios considerables en este. Este comportamiento muestra que el NMPC generado para este sistema es un poco más agresivo comparado con los otros sistemas. Vale la pena aclarar que todos los modelos tiene el mismo *Hold up* y no tienen restricciones en el movimiento de la variable manipulada.

También se encuentra que los modelos 2a y 3a presentan comportamientos similares relacionados con el error en el sistema y el movimiento en la variable manipulada. Estos modelos se diferencian del valor de *Gamma*, en el caso 2a es 1, en el caso 3a es un polinomio. Las variaciones en la relación de reflujo son del orden de 0.5 llevando la variable controlada a una estabilidad rápida.

Para el caso del modelo 4 que considera todas las ecuaciones de equilibrio sugeridas en [35], se encuentra que existe un error constante durante la predicción. Para este modelo, también se encontró mayor grado de dificultad en la programación y estabilidad númerica debido a la existencia de logaritmos y exponenciales en la ecuaciones. Por lo anterior, de ahora en adelante el modelo 4 no es considerado en el análisis, solamente se toma el modelo 3 que, como se nota en las figuras 6.4 y 6.5 tiene el mismo comportamiento del modelo 4 sin error.

La figura 6.14 presenta el comportamiento para los modelos 2b y 3b donde se manipuló la relación de reflujo y la energía del rehervidor. Para estos dos casos, se encontró que la energía del rehervidor no cambió, manteniendose en 4576140 kJ/min. Por lo anterior, sólo se presenta el cambio en la relación de reflujo.

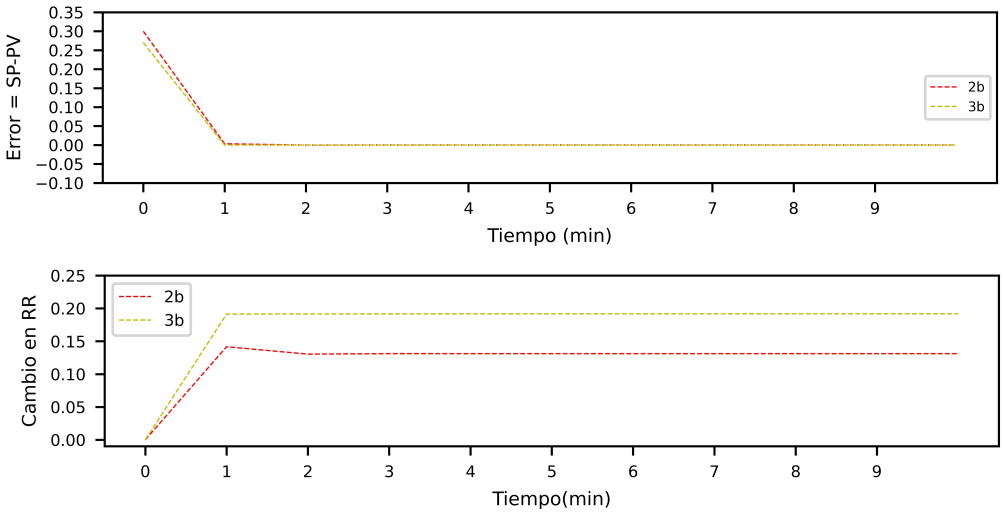

Figura 6.14: Comportamiento del error y la Relación de reflujo para los modelos 2b y 3b en un sistema donde se manipulan dos variables

Esta figura muestra la misma velocidad de respuesta para ambos casos, aproximadamente en 1 minuto la variable controlada llega al SP de manera precisa. Con relación al reflujo, se encuentra mayor variación en el modelo 3b, resultado del modelo empleado. Vale la pena configurar una función de costos en este sistema de modo que se logren cambios en la energía del condensador y rehervidor. Esta función aplicaria para los modelos 2 y 3 que tienen asociada la temperatura y entalpia del sistema.

La figura 6.15, finalmente presenta el comportamiento de los modelos 2c y 3c donde se manipulan 3 variables; relación de reflujo, energía del rehervidor y energía del condensador. Al igual que en el caso anterior, ni la energía del rehervidor ni la del condensador presentaron cambios manteniendose en 4576140 kJ/min para el rehervidor y -4576140 kJ/min en el condensador.

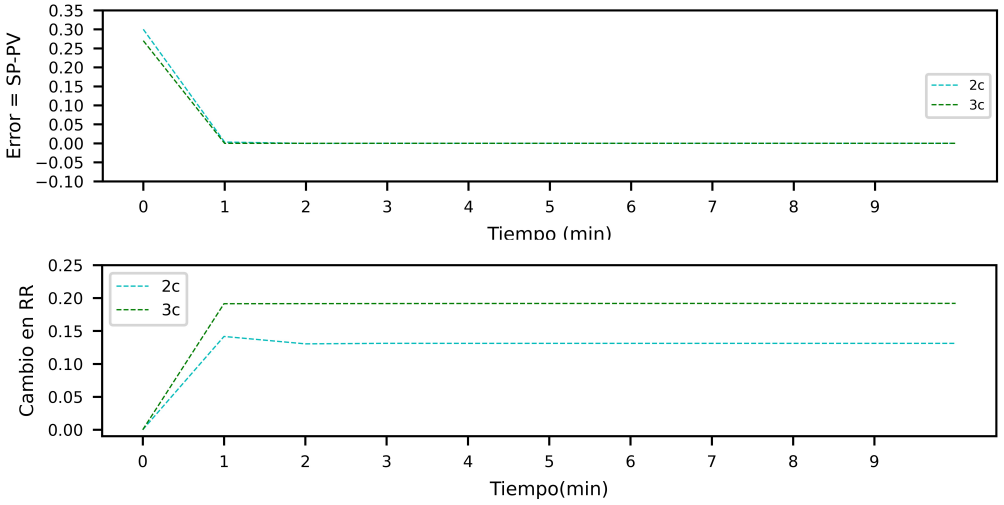

Figura 6.15: Comportamiento del error y la Relación de reflujo para los modelos 2c y 3c en un sistema donde se manipulan tres variables

Las figuras 6.14 y 6.15 prácticamente tienen el mismo comportamiento. Si bien cambia el número de variables manipuladas, únicamente hay variación en el reflujo. Para lograr cambios en la energía del condensador y rehervidor es necesario configurar un eMPC buscando la disminución de energía en el sistema.

En general, las figuras 6.13 6.14 y 6.15 validan que los modelos 1a, 2 y 3 siguen la ley de control establecida, manteniendo la variable controlada alrededor del valor deseado.

6.3.2. Sistemas BT y BTX

Estos sistemas corresponden a los modelos 2a y 3a donde únicamente se manipula la relación de reflujo. La figura 6.16 presenta los resultados de error y movimiento de la variable manipulada.

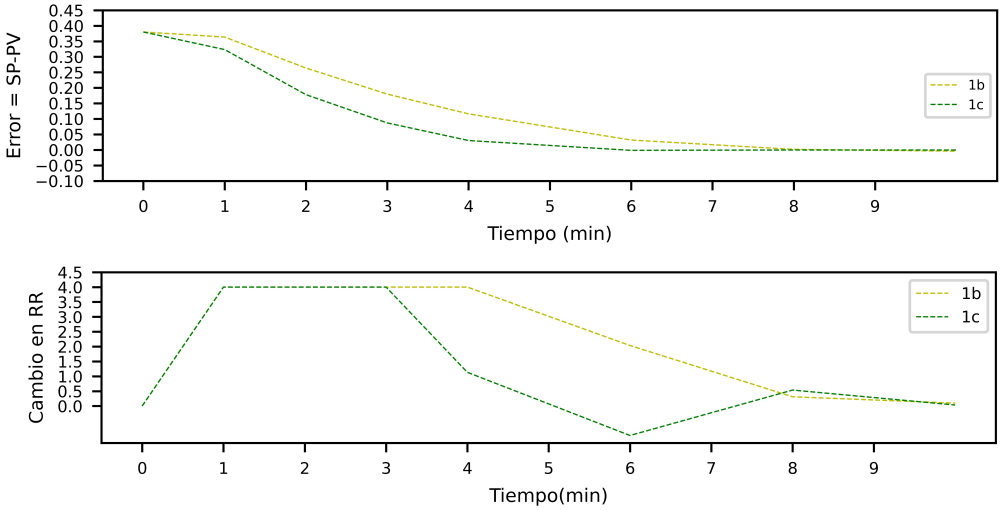

Figura 6.16: Comportamiento del error y la Relación de reflujo para los modelos 1b y 1c en un sistema donde se manipula la relación de reflujo

Al igual que para el sistema Etanol - Agua, se encuentra que el modelo sigue la ley de control. El movimiento en la variable manipulada es comparable con el del modelo 1a, al igual que la trayectoria para alcanzar al SP. Este comportamiento muestra ser característico del modelo 1, que es más sencillo y con la posibilidad de sólo manipular una variable.

6.4. Esfuerzo Computacional

En esta sección se presentan los resultados de ejecución del NMPC junto con el tiempo requerido para resolver el sistema. Para los 9 casos presentados en la sección 6.2, se indican el número de variables evaluadas como sus grados de libertad y evaluaciones de optimalidad. A su vez, también se discrimina, el porcentaje de tiempo dedicado a la ejecución del IPOPT que es el algoritmo de optimización, el tiempo dedicado a la ejecución del sistema de programación no lineal, que entre otras, realiza actividades asociadas a la evalación de derivadas y colocación

ortogonal y, finalmente el tiempo dedidado a otras actividades.

1. Resultados para el modelo 1

Característica	Valor modelo a	Valor modelo b	Valor modelo c
Número de Variables	35	55	81
Número total de Variables de Estado	2229	3429	5301
Grados de Libertad	57	57	57
Jacobianos evaluados	8129	12849	23729
Lagrangianos y Hessianos evaluados	2364	3804	6204
Número Total de Iteraciones	11	12	57
Tiempo total promedio (s)	5.7040	5.3677	13.2171
Consumo de CPU (porcentaje)	6.74	8.21	22.4

Cuadro 6.2: Tiempo Computacional para el modelo 1

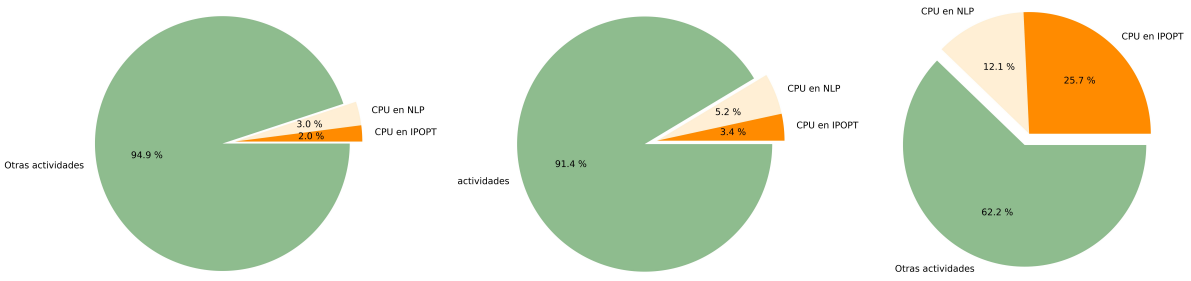

Figura 6.17: Tiempo Computacional para los modelos 1a,b,c

Para el cuadro 6.2 y la figura 6.17 se encuentra una diferencia considerable entre el número de variables para cada caso. Mientras el caso más sencillo emplea 2229 el más complejo 5301. Esta variación también es reflejada en el número que Jacobianos, Lagrangianos y Hessianos evaluados, correspondiendo a una de las actividades de NLP del proceso que va en la misma dirección de este aumento. Con relación al IPOPT, esta actividad consume un tiempo relativamente pequeño para los casos 1a y 1b, sistemas binarios con menos de 3500 ecuaciones. Para el caso 1c, su tiempo aumenta de manera notoria en concordancia con el número de iteraciones del modelo. Mientras los modelos 1a y 2a no exceden 12, el 3a tiene 57. El porcentaje de consumo de CPU para ninguno de los tres casos esta por encima del 23 porciento. Cabe anotar que este consumo esta relacionado con las actividades de NLP y otras actividades ya que el solver corre de forma remota. Es notorio el impacto que las otras actividades tienen sobre el tiempo computacional manteniendose por encima del 62 porciento en los tres casos.

2. Resultados para el modelo 2

El siguiente cuadro presenta los resultados para los casos 2a, 2b y 2c.

Característica	Valor modelo a	Valor modelo b	Valor modelo c
Número de Variables	188	189	190
Número total de Variables de Estado	5013	5061	5109
Grados de Libertad	33	45	57
Número de Jacobianos	15867	15973	16079
Número de Lagrangianos y Hessianos	3180	3192	3204
Número Total de Iteraciones	48	33	208
Tiempo total promedio (s)	13.7524	10.9805	28.3987
Consumo de CPU (porcentaje)	42.5	44.65	45.6

Cuadro 6.3: Tiempo Computacional para el modelo 2

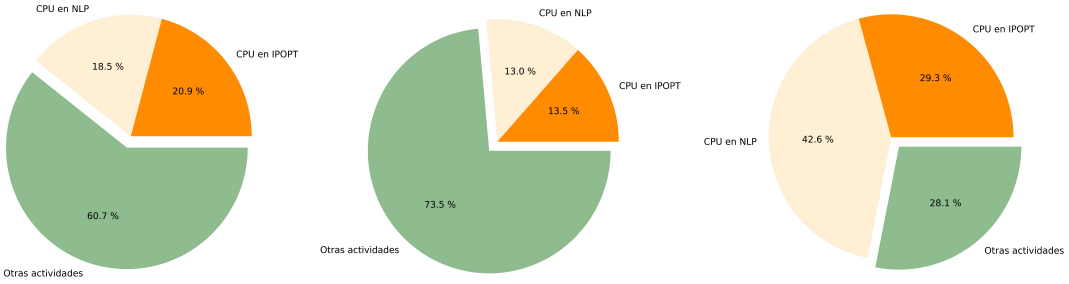

Figura 6.18: Tiempo computacional para los modelos 2a,b,c

Para este nuevo escenario, siendo el número de variables manipuladas la única diferencia, se encuentran entre 5013 y 5109 variables de estado, similares a las del modelo 1c. En este caso, la evaluación de Jacobianos, Lagranjianos y Hessianos es similar entre sí. El número de iteraciones y tiempo total son directamente proporcionales, así como el consumo de CPU. En este caso, el consumo de otras actividades ya tiene menos impacto que en el caso 1. Se encuentra mayor consumo de CPU y tiempo total sin estar estos por encima de 46 y 30s respectivamente.

Si se revisan los resultados del modelo 1c y 2a, los que tienen similares variables de estado. A pesar que el modelo 1c tiene un poco más de variables y requiere más iteraciones en la predicción del proceso, se encuentra menor tiempo total y menor consumo del CPU que el modelo 2a. Esto indica que el tiempo computacional no sólo depende del número de ecuaciones e iteraciones sino, adicional del grado de complejidad del modelo.

A pesar de manipular una variable adicional, se encuentra que el consumo computacional del modelo 2c es significativamente más grande que los modelos 2a y 2b. El cambio en el número de iteraciones es cuatro veces mayor llevando el tiempo de resolución del MPC en aproximadamente 30 s. No es posible indicar que el tiempo de resolución depende del número de variables manipuladas ya que, como se nota en el caso 2b este disminuyó con relación al modelo 2a. La tabla 6.3 muestra que a pesar que los tiempos computacionales son diferentes entre sí, el porcentaje de consumo de CPU no varía significativamente.

3. Resultados para el modelo 3

El siguiente cuadro presenta los resultados para los casos 3a, 3b y 3c.

Característica	Valor modelo a	Valor modelo b	Valor modelo c
Número de Variables	188	189	190
Número total de Variables de Estado	5013	5061	5109
Grados de Libertad	33	45	57
Número de Jacobianos	16203	16309	16415
Número de Lagrangianos y Hessianos	3516	3528	3540
Número Total de Iteraciones	70	97	221
Tiempo total promedio (s)	16.7764	17.5406	29.1858
Consumo de CPU (porcentaje)	42.8	44.15	45.45

Cuadro 6.4: Tiempo Computacional para el modelo 3

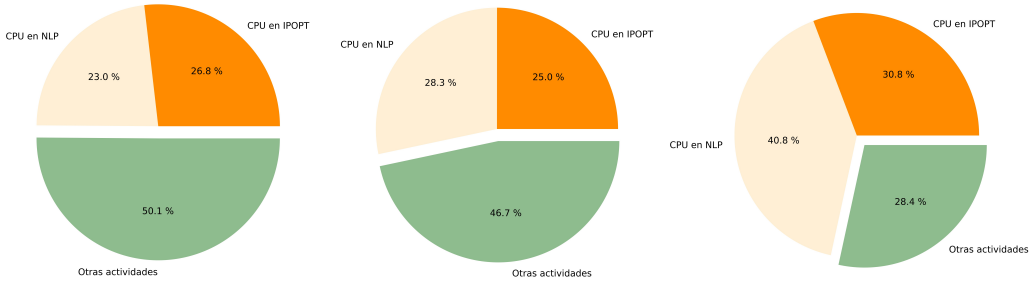

Figura 6.19: Tiempo computacional para los modelos 3a,b,c

Vale la pena notar que en este escenario, entre más variables manipuladas se analizan, mayor número de iteraciones y tiempo computacional requiere el modelo, diferente al caso 2b. Ambos modelos presentan tiempos de ejecución y consumo de CPU similar. Con relación a las figuras 6.18 y 6.19, el porcentaje de tiempo de cada actividad es similar para cada escenario. Este resultado abre la posibilidad a ejecutar NMPC a sistemas reales aplicando curvas que describan el equilibrio.

La figura 6.20, presenta el tiempo computacional para cada modelo.

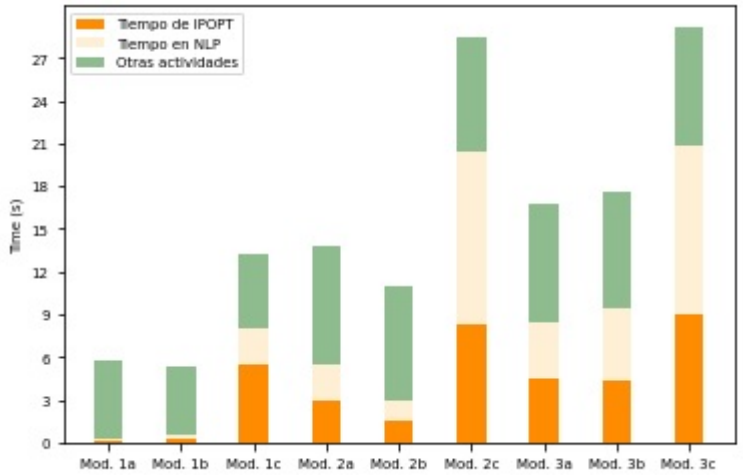

Figura 6.20: Tiempo computacional para los modelos 1, 2 y 3

Fácilmente se identifica el menor tiempo computacional de los modelos 1a y 1b. Sistemas con menor número de variables de estado, modelo más sencillo y menor número de iteraciones. Desde el modelo 1c a 3c las actividades de solución del solver y de las actividades de NLP son más influyentes en el tiempo computacional y aumentan casi en la misma proporción. También se encuentra que el tiempo dedicado en otras actividades es similar para los 9 casos, encontrándose entre 4 y 6 segundos. En ninguno de estos casos el tiempo computacional esta por encima de 29 s.

La figura 6.21 presenta el resumen del número de variables asociadas, variables de estado y Jacobianos evaluados.

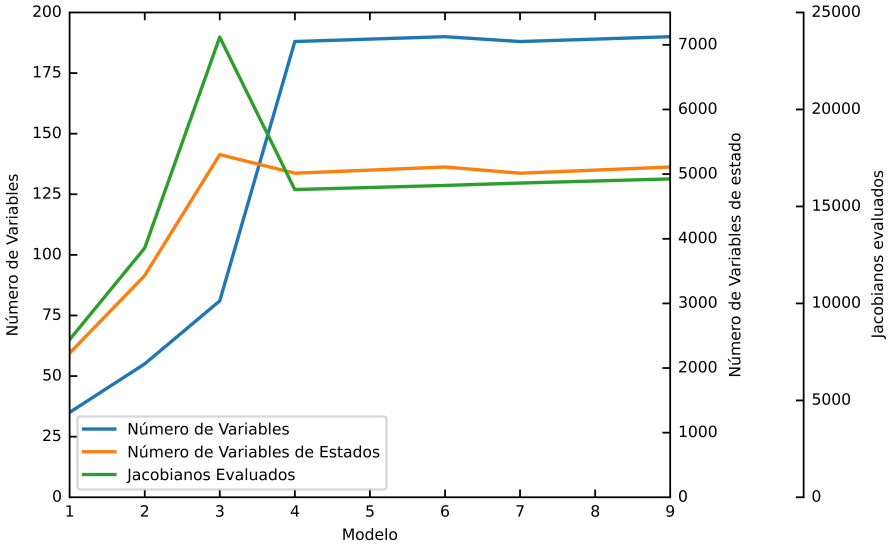

Figura 6.21: Resumen General de algunas propiedades del MPC del proceso

Esta figura muestra que no existe relación entre el número de variables de estado y la cantidad de evaluaciones de derivadas del sistema evidenciado en el caso 3. También se puede notar que los modelos 4 a 9 presentan resultados similares, siendo representados bajo el mismo modelo donde sólo cambia el valor de *Gama*.

La figura 6.22 presenta un resumen del tiempo computacional, consumo de CPU y número de iteraciones de los 9 modelos.

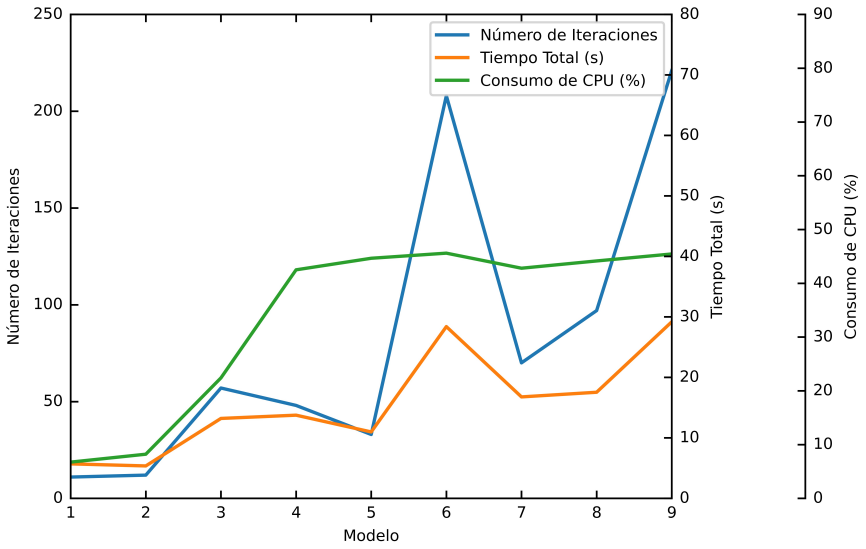

Figura 6.22: Resumen General de algunas propiedades del MPC en el Hardware

Esta figura refleja la influencia que tiene el número de iteraciones sobre el tiempo computacional, no obstante, no se refleja impacto sobre el consumo de CPU, este se encuentra relacionado con el tipo de modelo empleado.

6.5. Ejecución del NMPC

El modelo 3a fue conectado via OPC con la simulación en Aspen Dynamics para evaluar el desempeño de esta lógica en un proceso. Esta conexión se realizó por medio de OPC, por sus siglas en inglés *OLE for process control*, que es una interfaz para la comunicación de información entre diferentes software que elimina los problemas asociados a los driver propietarios. Esta se realiza a través de una arquitectura cliente-servidor.

La figura 6.23 presenta la estrategia de comunicación implementada. Apoyados en los códigos presentados por Yamanee et al [38], se realizó la comunicacíón entre Python y Aspen Dynamics considerando esta simulación el proceso donde se aplica el NMPC.

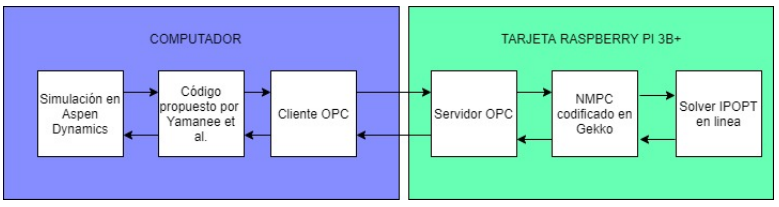

Figura 6.23: Arquitectura de la conexión OPC realizada

Los códigos utilizados para dicha conexión son relacionados en las figuras 6.24, 6.25 y 6.26. Estos son descritos tanto para el cliente como para el servidor OPC.

```python
import win32com.client

class COM_conn:

    adyn=win32com.client.Dispatch("AD Application")

    def __init__(self):
        self.adyn=win32com.client.Dispatch("AD Application")
        self.XethanolCondenser=0;
        self.FeedFLow=0
        self.Tfeed=0
        self.ZethanolFeed=0


    def connect(self,runTime):
        path="C:\\Users\\sammi\\Desktop\\UNIVERSIDAD DE LA SABANA\\v8-2\\"
        model_name="V8-2.dynf"

        #Create connection Python ACM via COM

        #Make apen Dynamics visible and active
        self.adyn.visible=True
        self.adyn.activate()
        self.adyn.OpenDocument(str.format(path+model_name))
        sim=self.adyn.Simulation
        sim.options.TimeSettings.RecordHistory=True
        sim.endTime=runTime
        sim.run(0)


    def update(self,RefluxRatio): #Manipulated Variable
        sim=self.adyn.Simulation
        fsheet= sim.Flowsheet
        bblocks = fsheet.Blocks
        bblocks("B1").RefluxRatio.value=RefluxRatio

    def read(self):
        sim=self.adyn.Simulation
        fsheet= sim.Flowsheet
        sstreams=fsheet.Streams

        self.XethanolCondenser =sstreams("S4").Zn("ETHANOL").value
        self.Tfeed=sstreams("S1").T.value
        self.FeedFLow=sstreams("S1").FR.value
        self.ZethanolFeed=sstreams("S1").Zn("ETHANOL").value

```

Figura 6.24: Código de Comunicación con Aspen

Por medio del código presentado en la figura 6.24 se realiza la comunicación entre el cliente OPC en Python con la simulación realizada en Aspen Dynamics.

```python
import asyncio
import sys
import COM_conn_Aspen as CCA
import logging
from asyncua import Client, Node, ua


sys.path.append("C:\\Users\\sammi\\Desktop\\UNIVERSIDAD DE LA SABANA\\CODIGOS SANDRA\\")
#import opcclient

logging.basicConfig(level=logging.INFO)
_logger = logging.getLogger('asyncua')


async def main():
    objCOM=CCA.COM_conn()
    objCOM.connect(200)
    url = 'opc.tcp://192.168.0.4:4840/freeopcua/server/'

    async with Client(url=url,timeout=40000) as client:
        while True:
            _logger.info('Children of root are: %r', await client.nodes.root.get_children())
            uri = 'http://examples.freeopcua.github.io'
            idx = await client.get_namespace_index(uri)


            varRefluxRatio = await client.nodes.root.get_child(["0:Objects", f"{idx}:MyObject", f"{idx}:RefluxRatio"])
            varXethCond=await client.nodes.root.get_child(["0:Objects", f"{idx}:MyObject", f"{idx}:XethCond"])
            varZethFeed=await client.nodes.root.get_child(["0:Objects", f"{idx}:MyObject", f"{idx}:ZethFeed"])
            varTfeed=await client.nodes.root.get_child(["0:Objects", f"{idx}:MyObject", f"{idx}:Tfeed"])
            varFeedFlow=await client.nodes.root.get_child(["0:Objects", f"{idx}:MyObject", f"{idx}:FeedFlow"])

            RefluxRatio=await varRefluxRatio.get_value()
            print("Reflux Ratio", RefluxRatio)#, await varRefluxRatio.read_value())

            objCOM.update(RefluxRatio)

            await asyncio.sleep(20)

            objCOM.read()

            await varXethCond.write_value(objCOM.XethanolCondenser) # set node value using implicit data type
            await varZethFeed.write_value(objCOM.ZethanolFeed)
            await varTfeed.write_value(objCOM.Tfeed)
            await varFeedFlow.write_value(objCOM.FeedFLow)

if __name__ == '__main__':
    asyncio.run(main())
```

Figura 6.25: Comunicación con el cliente

En la figura 6.25 el cliente OPC envia la información de la composición de Etanol en cima, composición, flujo y temperatura de alimento. También recibe la información de relación de reflujo cálculada por el código NMCP.

```python
import logging
import asyncio
import sys
sys.path.insert(0, "..")
#sys.path.append("C:\\Users\\sammi\\Desktop\\UNIVERSIDAD DE LA SABANA\\MAESTRIA$
import mpcRaspberryPi2 as mpcRP
from asyncua import ua, Server
from asyncua.common.methods import uamethod


logging.basicConfig(level=logging.INFO)
_logger = logging.getLogger('asyncua')


@uamethod
def func(parent, value):
    return value

async def main():
    # setup our server
    server = Server()
    await server.init()
    server.set_endpoint('opc.tcp://0.0.0.0:4840/freeopcua/server/')

    # setup our own namespace, not really necessary but should as spec
    uri = 'http://examples.freeopcua.github.io'
    idx = await server.register_namespace(uri)

    # populating our address space
    # server.nodes, contains links to very common nodes like objects and root
    myobj = await server.nodes.objects.add_object(idx, 'MyObject')
    varRefluxRatio = await myobj.add_variable(idx, 'RefluxRatio', 0.7)
    varXethCond=await myobj.add_variable(idx,'XethCond',0.5)
    varZethFeed=await myobj.add_variable(idx,'ZethFeed',0.5)
    varTfeed=await myobj.add_variable(idx,'Tfeed',352)
    varFeedFlow=await myobj.add_variable(idx,'FeedFlow',100)

    objmpc=mpcRP.MPC()

    # Set MyVariable to be writable by clients
    await varRefluxRatio.set_writable()
    await varXethCond.set_writable()
    await varZethFeed.set_writable()
    await varTfeed.set_writable()
    await varFeedFlow.set_writable()

    #await server.nodes.objects.add_method(ua.NodeId('ServerMethod', 2), ua.Qua$
    _logger.info('Starting server!')
    async with server:

        while True:
            await asyncio.sleep(1)
            XethCond=await varXethCond.get_value()
            ZethFeed=await varZethFeed.get_value()
            Tfeed=await varTfeed.get_value()
            FeedFlow=await varFeedFlow.get_value()

            _logger.info('XethCond is %.5f',XethCond)
            _logger.info('ZethFeed is %.5f',ZethFeed)
            _logger.info('Tfeed is %.5f',Tfeed)
            _logger.info('FeedFlow is %.5f',FeedFlow)


            #objmpc.calculateBequette()
            #objmpc.calculateRaoult()
            objmpc.calculateRaoultNRTL()

            #objmpc.ModifyDisturbancesBequette(FeedFlow,ZethFeed)

            # new_val = await myvar.get_value() +0.03
            new_val = objmpc.RefluxRatio
            #new_val2=await myvar2.get_value()
            _logger.info('RefluxRate is calculated to be %.5f', new_val)
            await varRefluxRatio.write_value(new_val)

            #_logger.info('Get value of %s to %.1f', myvar2, new_val2)
            objmpc.ModifyDisturbancesBequette(FeedFlow,ZethFeed,XethCond)
            #objmpc.ModifyDisturbancesRaoult(FeedFlow,ZethFeed,Tfeed,XethCond)
            #objmpc.ModifyDisturbancesRaoultNRTL(FeedFlow,ZethFeed)


if __name__ == '__main__':
    asyncio.run(main())
```

Figura 6.26: Comunicación con el servidor OPC

Finalmente, la figura 6.26 se encuentra la comunicación entre el cliente OPC y la configuración del NMPC en la tarjeta Raspberry Pi 3B+.

La figura 6.27 presenta los resultados obtenidos frente a un setpoint deseado de 0.767 molar para el Etanol en Cima. También se presenta el comportamiento frente a ruido en el flujo de alimentación.

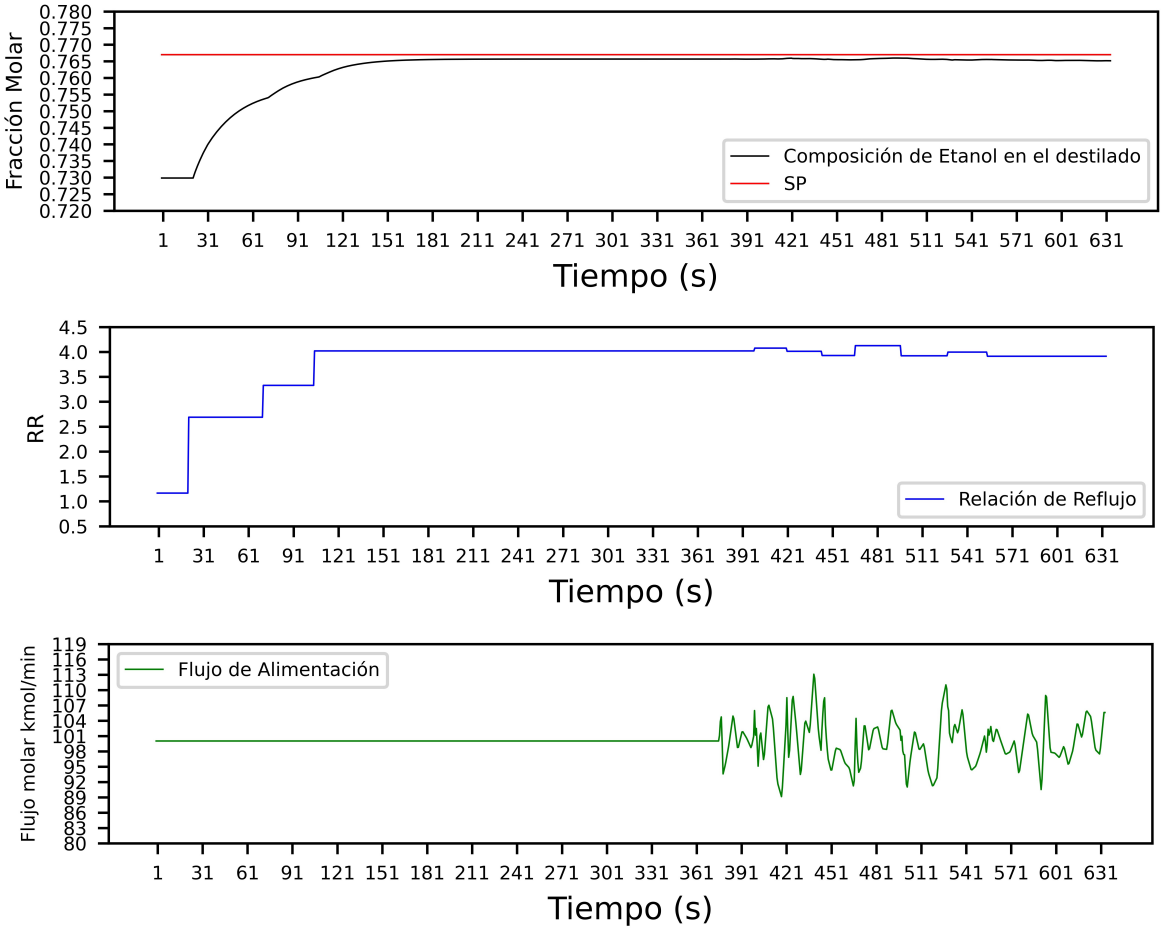

Figura 6.27: Resultado del MPC aplicado

Se encuentra que el modelo predice adecuadamente el comportamiento del proceso, llevándolo al SP deseado y respondiendo de forma satisfactoria a las perturbaciones del proceso.

.

Capítulo 7

Conclusiones

Este trabajo permite concluir items relacionados con el desempeño de los modelos y de tarjeta Raspeberry Pi 3B+.

La biblioteca Gekko permitió resolver los diferentes modelos del proceso tanto en estado estable, como en dinámico y MPC, siendo esta de acceso libre y programable en Python.

Se encuentra que los 3 modelos evaluados en las mezcla Etanol - Agua, BT y BTX describen de forma adecuada el proceso en estado estable simulado en Aspen siendo esta etapa fundamental en el desarrollo del NMPC de acuerdo con lo propuesto en [33]. Con relación al NMPC, el modelo 1, representa de forma satisfactoria sistemas binarios y ternarios con sustancias químicamente similares, donde puede manipularse la relación de reflujo. El modelo 3, por su parte, describe satisfactoriamente el perfil de sistemas reales donde puede manipularse y controlarse más de una variable permitiendo lograr un control más realista y con posibilidad de hacer optimización económica o ENMPC.

Los casos evaluados muestran que actividades como comunicación, procesamiento de datos de entrada y escritura de los archivos de solución, tambien llamado $Wall\ Time$, es similar en todos los casos, mostrando que no es dependiente de la complejidad del sistema evaluado. Los tiempos dedicados a la solución del NLP e IPOPT son similares entre sí para cada caso. Este tiempo esta influenciado en primera medida, por el número de variables, las iteraciones del IPOPT y el número de variables manipuladas. Con relación al consumo de CPU, las iteraciones no tienen tanto efecto en el tiempo como si lo presenta el tipo de modelo programado, presentado en la figura 6.22.

La tarjeta Raspberry Pi 3B+ en conjunto con la libreria Gekko de Python permiten desarrollar el NMPC para 3 diferentes modelos de una columna de destilación cumpliendo la ley de control y resolviendo el problema de optimización dinámica en menos de 1 minuto con un consumo de CPU menor al 50 porciento. Su aplicación, como se presenta en la figura 6.27 muestra convergencia, relación con el proceso y estabilidad frente a perturbaciones. Este resultado valida su uso para sistemas con alta no linealidad y complejidad favoreciendo su uso a nivel académico e industrial.

Capítulo 8

Recomendaciones

El detallado análisis del consumo computacional en la resolución de problemas de optimización dinámica permite validar el uso de las tarjetas Raspberry Pi 3B+ en aplicaciones de alto nivel. Como etapas futuras a este trabajo, se sugieren las siguientes actividades:

- Evaluar el desempeño de las tarjetas en la ejecución del NMPC para sistemas reales.
- Configurar la tarjeta en paralelo de modo que su rendimiento sea más alto.
- Evaluar el desempeño del NMPC para los modelos 2 y 3 en los sistemas BT y BTX. Asi mismo, evaluar estos modelos en sistemas más complejos como son destilaciones extractivas y separaciones multicomponentes.
- Realizar el análisis de los modelos en sistemas donde exista más de una variable controlada.
- Evaluar la influencia el efecto del horizonte de predicción y horizonte de control sobre los resultados de la simulación.
- Basados en el modelo desarrollado, diseñar un eNMPC buscando el mínimo consumo de energía en el sistema.
- Evaluar la posibilidad de ejecutar el IPOPT en la tarjeta Raspberry Pi.

Capítulo 9

Entregables

El presente estudio generó los siguientes documentos:

1. Participación en el IX Congreso Internacional de Ingeniería Mecatrónica y Automatización, CIIMA 2020. La figura 9.1 presenta el Certificado de presentación de comunicación oral.

Certificado de presentación de comunicación oral

Los autores:
Sandra Rodriguez, Edgar Mayorga and Manuel Figueredo

Han participado en el
IX Congreso Internacional de Ingeniería Mecatrónica y Automatización CIIMA 2020,
llevado a cabo en Cartagena, Colombia del 4 al 6 de noviembre de 2020, con la comunicación titulada:
Evaluation of dynamic model and assessing computational time of an embedded system. Case study: A distillation column

Edgardo Arrieta
Presidente
Comité organizador

Andrés G. Marrugo
Director
Comité científico

Universidad Tecnológica de Bolívar

RIMA

Figura 9.1: Certificado de participación en congreso

2. Artículo corto donde se analiza el tiempo computacional para el modelamiento dinámico de la columna de destilación utilizando los modelos 1b y 1c. El articulo se denomina " Evaluation of dynamic model and assesing computational time of an embedded system. Case Study: A distillation column ". Publicado en IEEExplore en Noviembre de 2020.

2. Se esta preparando un artículo para ser sometido en revista indexada.

Bibliografía

[1] D. E. Seborg, *Process Dynamics and Control*. United States: Wiley, John & Sons, third edition ed., 2011.

[2] L. T. Biegler, "New directions for nonlinear process optimization," *Current Opinion in Chemical Engineering*, vol. 21, pp. 32–40, 2018.

[3] F. Holtorf, A. Mitsos, and L. T. Biegler, "Multistage NMPC with on-line generated scenario trees: Application to a semi-batch polymerization process," *Journal of Process Control*, vol. 80, pp. 167–179, 2019.

[4] E. Aydin, D. Bonvin, and K. Sundmacher, "Computationally efficient NMPC for batch and semi-batch processes using parsimonious input parameterization," *Journal of Process Control*, vol. 66, pp. 12–22, 2018.

[5] R. Moriyasu, S. Nojiri, A. Matsunaga, T. Nakamura, and T. Jimbo, "Diesel engine air path control based on neural approximation of nonlinear MPC," *Control Engineering Practice*, vol. 91, no. August, p. 104114, 2019.

[6] A. Sharma, J. Drgoňa, D. Ingole, J. Holaza, R. Valo, S. Koniar, and M. Kvasnica, "Teaching Classical and Advanced Control of Binary Distillation Column," *IFAC-PapersOnLine*, vol. 49, no. 6, pp. 348–353, 2016.

[7] Y. Zhu, Z. Xu, J. Zhao, K. Han, J. Qian, and W. Li, *Development and application of an integrated MPC technology*, vol. 17. IFAC, 2008.

[8] E. Camacho and C. Alba, *Model predictive control*. Sevilla: Springer, second edition ed., 2013.

[9] T. Vermon L, *A guide to the Automation Body of Knowledge*. United States: The instrumentation and automation society, second edition ed., 2006.

[10] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Computers and Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[11] J. Kirby, L. Chapman, and V. Chapman, "Assessing the Raspberry Pi as a low-cost alternative for acquisition of near infrared hemispherical digital imagery," *Agricultural and Forest Meteorology*, vol. 259, no. May, pp. 232–239, 2018.

[12] L. Beal, D. Hill, R. Martin, and J. Hedengren, "GEKKO Optimization Suite," *Processes*, vol. 6, no. 8, p. 106, 2018.

[13] P. J. Basford, S. J. Johnston, C. S. Perkins, T. Garnock-Jones, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, J. Singer, and S. J. Cox, "Performance analysis of single board computer clusters," *Future Generation Computer Systems*, vol. 102, pp. 278–291, 2020.

[14] Á. C. E. Sousa, V. J. S. Leite, and I. R. Scola, "Affordable Control Platform with MPC Application," *Studies in Informatics and Control*, vol. 27, no. September, pp. 265–274, 2018.

[15] M. O'Brien, A. Hall, J. Schrauwen, and J. van der Made, "An open-source approach to automation in organic synthesis: The flow chemical formation of benzamides using an inline liquid-liquid extraction system and a homemade 3-axis autosampling/product-collection device," *Tetrahedron*, vol. 74, no. 25, pp. 3152–3157, 2018.

[16] V. Bagyaveereswaran, T. D. Mathur, S. Gupta, and P. Arulmozhivarman, "Performance comparison of next generation controller and MPC in real time for a SISO process with low cost DAQ unit," *Alexandria Engineering Journal*, vol. 55, no. 3, pp. 2515–2524, 2016.

[17] K. V. Ling, B. F. Wu, and J. Maciejowski, "Embedded Model Predictive Control (MPC) using a FPGA," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 17, no. 1 PART 1, pp. 15250–15255, 2008.

[18] F. Lozano Santamaría and J. M. Gómez, "Framework in PYOMO for the assessment and implementation of (as)NMPC controllers," *Computers and Chemical Engineering*, vol. 92, pp. 93–111, 2016.

[19] Y. Cao, D. Acevedo, Z. K. Nagy, and C. D. Laird, "Real-time feasible multi-objective optimization based nonlinear model predictive control of particle size and shape in a batch crystallization process," *Control Engineering Practice*, vol. 69, no. March, pp. 1–8, 2017.

[20] D. E. Bernal, C. Carrillo-Diaz, J. M. Gómez, and L. A. Ricardez-Sandoval, "Simultaneous design and control of catalytic distillation columns using comprehensive rigorous dynamic models," *Industrial and Engineering Chemistry Research*, vol. 57, no. 7, pp. 2587–2608, 2018.

[21] L. D. Beal, D. Petersen, D. Grimsman, S. Warnick, and J. D. Hedengren, "Integrated scheduling and control in discrete-time with dynamic parameters and constraints," *Computers and Chemical Engineering*, vol. 115, pp. 361–376, 2018.

[22] H. M. Mora Escobar, *Programación Lineal, Métodos y Programas*. Depto de Matemáticas Universidad Nacional de Colombia, primera ed., 1997.

[23] D. de la Fuente García and P. Moreno, *Programación lineal entera y programación no lineal*. Servicio de Publicaciones de la Universidad de Oviedo, 1996.

[24] W. Rudin and L. Garcia, *Análisis funcional*. Reverté, 2002.

[25] H. Mora, *Optimización No Lineal Y Dinámica*. Departamento De Matemáticas Y Estadística Universidad Nacional de Colombia, 0 ed., 2001.

[26] Y. Shin, R. Smith, and S. Hwang, "Development of model predictive control system using an artificial neural network: A case study with a distillation column," *Journal of Cleaner Production*, vol. 277, 2020.

[27] A. Juneja and G. S. Murthy, "Model predictive control coupled with economic and environmental constraints for optimum algal production," *Bioresource Technology*, vol. 250, no. September 2017, pp. 556–563, 2018.

[28] M. Manimaran, A. Arumugam, G. Balasubramanian, and K. Ramkumar, "Optimization and composition control of distillation column using MPC," *International Journal of Engineering and Technology*, vol. 5, no. 2, pp. 1224–1230, 2013.

[29] G. Van, *El tutorial de Python.* Argentina: Fred L. Drake, primera ed ed., 2013.

[30] Skogestad; Sigurd in *Dynamics and Control of distillation Columns].pdf*, ch. 18, pp. Vol 18, No 3 177–217, sciencedirect, 1997.

[31] B. W. Bequette, *Process Dynamics Modeling, Analysis and Simulation.* New Jersey: Prentice Hall PTR, 1998.

[32] D. Seader, J.D. Henley, Ernest J. KeithRoper, *Separation Process Principles, Chemical and Biochemical Operations.* J.Wiley & Sons, Inc., third edit ed., 2010.

[33] S. M. Safdarnejad, "Developing Modeling, Optimization, and Advanced Process Control Frameworks for Improving the Performance of Transient Energy-Intensive Applications," *PhD Thesis, BYU university*, 2016.

[34] J. Smith, H. Van Ness, and M. Abbott, *Introducción a la Termodinámica en Ingeniería Química.* California: Mc Graw Hill, septima ed., 2009.

[35] J. Prausnitz, *Fluid Phase Equilibria.* United States: Prentice Hall, second edition ed., 1987.

[36] T. L. Tolliver, "Fundamentals of Distillation Column Control," *Advances in Instrumentation, Proceedings*, vol. 35, no. pt 2, pp. 611–626, 1980.

[37] R. S. M. E. Figueredo, M., "Evaluation of dynamic model and assessing computational time of an embedded system. Case study: A distillation column," *IEEExplore*, vol. 69, no. November, pp. 1–8, 2020.

[38] M. Yamanee-Nolin, N. Andersson, B. Nilsson, M. Max-Hansen, and O. Pajalic, "Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain," *Chemical Engineering Research and Design*, vol. 155, pp. 12–17, 2020.