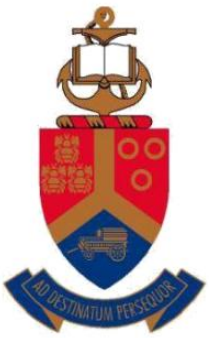


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

9/1/2025

CSC-421 Research Project

Modelling a Reactive H_2S Stripping Column for Na_2CO_3 Production

Student: Mr. DA (Damian) van Tonder, u21449300
Supervisor: Prof. DS van Vuuren

Abstract

This project focuses on developing a computational model for a reactive H_2S stripping column that transforms Na_2SO_4 into Na_2CO_3, with CO_2 as the stripping agent. The model accounts for the slow kinetics of CO_2 hydration to carbonic acid (H_2CO_3), a rate-limiting step often neglected in equilibrium-based simulations. Implemented in *Python 3.13.3*, the model integrates stage-wise mass balances, chemical equilibria, and kinetic rate expressions. It uses temperature-dependent physical properties, mass transfer coefficients, and kinetic constants to solve a system of nonlinear equations per stage, employing *scipy.optimize.fsolve* with enhanced numerical stability.

The simulation tracks sulphur and carbon species, vapor compositions, and pH profiles, and performs sensitivity analyses across variables like temperature, recovery rate, stage volume, feed concentration, and outlet gas flowrate. Results confirm that ignoring CO_2 hydration kinetics leads to significant deviations, especially at high recovery targets. The model is validated through physical constraints and residual analysis, proving its robustness and utility for process design and optimization.

Table of Contents

List of Figures

List of Tables

Nomenclature

English Symbol	Description
a	specific interfacial area
d_e	bubble diameter
D_L	liquid diffusivity
G	gas flowrate in the stripping column
G_N	gas flowrate leaving the stripping column
g	gravitational acceleration constant
K	equilibrium/dissociation/ionization constant
k_f	forward reaction rate constant
k_H	Henry constant
k_L	liquid-side mass transfer coefficient
$k_L a$	volumetric mass transfer coefficient
k_r	reverse reaction rate constant
L	liquid flowrate in the stripping column
N	total number of stages in the stripping column
P	pressure
pH_N	pH of the liquid leaving the stripping column
Re	Reynold's number
r_f	forward reaction rate
r_r	reverse reaction rate
S	salinity
Sc	Scmidt number
Sh	Sherwood number
T	temperature
u_b	rising velocity for a bubble
V	liquid volume for a stage in the stripping column
V_{total}	total liquid volume in the stripping column
x	fractional recovery
y	vapor fraction
y_N	vapor fraction leaving the stripping column

Greek Symbol	Description
ε	gas hold-up
λ_x	a constant in the liquid diffusivity calculation
μ_L	dynamic viscosity of the liquid
π	numerical value of pi (which is 3.141593…)
ρ_L	density of the liquid
σ_L	surface tension of the liquid

1. Introduction

The global chemical industry continuously seeks opportunities to convert low-value industrial byproducts into high-value commodities. Sodium sulfate (Na_2SO_4), a common byproduct from various chemical processes, presents such an opportunity when converted into sodium carbonate (Na_2CO_3), a widely used chemical in glass manufacturing (ChemicalBook, 2022) and in the paper industry (Co., n.d.). A promising approach involves a multi-step chemical pathway in which a key intermediate step is the reactive stripping of hydrogen sulfide (H_2S) from an aqueous sodium hydrosulfide ($NaHS$) solution using carbon dioxide (CO_2). This reaction results in the formation of sodium bicarbonate ($NaHCO_3$), which can subsequently be thermally decomposed to sodium carbonate.

A reliable and predictive model of this reactive stripping process is crucial for scaling the process from laboratory or pilot scale to industrial implementation. Initial models have relied heavily on equilibrium assumptions, treating all reactions and mass transfer steps as instantaneous. However, empirical studies have shown that the hydration of CO_2 in water to form carbonic acid (H_2CO_3) is significantly slower than previously assumed and can become the rate-limiting step in such systems. For instance, (Dreybrodt, 1996) investigated the dissolution of calcite and identified the hydration reaction $CO_2 + H_2O \rightleftharpoons H^+ + HCO_3^-$ as a rate-limiting step in the H_2O-CO_2-$CaCO_3$ system. If this reaction is not properly accounted for in the modeling, the simulation results will diverge significantly from real-world observations.

Thus, this project proposes the development of a robust stage-wise computational model for a reactive H_2S stripping column, implemented in Python, that explicitly accounts for the slow hydration kinetics of CO_2. The model will integrate thermodynamic equilibria, gas-liquid mass transfer, and chemical reaction kinetics to simulate the behavior of the system across a range of operational parameters. This model will serve as a tool for both understanding and optimizing the process, laying the groundwork for future experimental validation and potential industrial deployment.

1.1. Problem Statement

A process is being developed to convert Na_2SO_4 into Na_2CO_3. A key step involves the stripping of H_2S from an aqueous $NaHS$ solution using CO_2, forming $NaHCO_3$. Initial modeling efforts based on equilibrium assumptions have proven insufficient, due to the slow kinetics of the CO_2 hydration reaction. To accurately simulate and optimize this process, a dynamic model must be developed that captures this kinetic limitation while accounting for all relevant equilibria and mass transfer phenomena.

1.2. Background

The conversion of sodium sulfate (Na_2SO_4) to sodium carbonate (Na_2CO_3) has attracted considerable attention due to its environmental and economic benefits. Sodium sulfate is often generated as a low-value byproduct in various industries, particularly in pulp and paper, textiles, and chemical manufacturing. Converting it into sodium carbonate not only reduces waste but also provides a high-demand commodity chemical used in glass production, detergents, and chemical feedstocks. Traditional disposal methods for sodium sulfate, such as landfilling or discharge into water bodies, pose environmental challenges, making valorization pathways like this conversion process increasingly relevant.

A central step in this conversion involves the stripping of hydrogen sulfide (H_2S) using carbon dioxide (CO_2), a process that is inherently complex due to the coexistence of multiple ionic equilibria and simultaneous gas-liquid mass transfer phenomena. In the aqueous phase, CO_2 dissolves and undergoes a hydration reaction to form carbonic acid (H_2CO_3), which can dissociate further into bicarbonate (HCO_3^-) and carbonate (CO_3^{2-}). At the same time, dissolved H_2S dissociates into bisulfide (HS^-) and sulfide (S^{2-}) ions. The delicate interplay between these acid–base reactions and the gas–liquid transfer rates govern the composition of the liquid phase and the driving force for further mass transfer.

A common simplification in literature is to assume that all these reactions reach equilibrium instantaneously. However, Soli & Byrne (2002) and Schulz et al. (2006) have shown that the CO_2 hydration step is considerably slower than the other reactions, making it kinetically limiting under many process conditions. Equilibrium-based models neglect this kinetic barrier and therefore fail to predict accurate concentrations of bicarbonate and carbonate in short-contact time operations such as stripping columns.

In addition, mass transfer coefficients and diffusivities of CO_2 and H_2S depend strongly on temperature, pressure, and solution composition. Henry's law constants and thermodynamic dissociation constants are essential for computing species distributions with accuracy. Correlations and experimental methods for determining these parameters have been reported by Nock et al. (2015) and Guo et al. (2018), but most are based on binary or simplified systems rather than the multicomponent, reactive environments encountered in industrial stripping operations

Despite the abundance of individual reaction and transport data, there remains a significant lack of integrated, stage-wise models that couple kinetics with equilibrium effects in reactive stripping columns. Addressing this gap is critical for reliable process design and optimization. This project takes on that challenge by developing such a model, aiming to provide deeper insight into the coupled reactive behavior of H_2S and CO_2 in complex, multicomponent systems and ultimately enabling more efficient and sustainable conversion of sodium sulfate to sodium carbonate.

1.3. Aim and Objectives

The aim of this project is to develop a computer program that accurately models a reactive H_2S stripping column, with a particular focus on incorporating the kinetic limitation of CO_2 hydration to carbonic acid. To achieve this aim, the project will first involve a comprehensive literature review on CO_2 hydration kinetics, gas-liquid mass transfer, and the chemical equilibria relevant to CO_2-H_2S systems. This will be followed by the collection and analysis of kinetic, thermodynamic, and transport property data essential for reliable model input. Using this foundation, the project will derive detailed stage-wise mass and charge balance equations for a multicomponent reactive stripping column. These equations will integrate Henry's laws, equilibrium relationships, rate expressions for all relevant species, and mass transfer relationships for all relevant species.

The complete set of equations will then be implemented in Python using suitable numerical solvers, allowing for iterative and flexible model development. Sensitivity analyses will be performed to explore the influence of operational parameters such as temperature, pressure, and gas-liquid flow rates, thereby identifying potential process bottlenecks and optimization opportunities. Finally, the model outputs will be validated against available literature data and theoretical expectations to ensure credibility and accuracy.

2. Literature Review

2.1. Chemical Reactions Involved

The model for the reactive stripping column will incorporate reaction equilibria and kinetics for the following reactions:

$$CO_2(aq) + H_2O(l) \rightleftharpoons H_2CO_3(aq)$$

$$H_2CO_3(aq) \rightleftharpoons H^+(aq) + HCO_3^-(aq)$$

$$HCO_3^-(aq) \rightleftharpoons H^+(aq) + CO_3^{2-}(aq)$$

$$H_2S(aq) \rightleftharpoons H^+(aq) + HS^-(aq)$$

$$HS^-(aq) \rightleftharpoons H^+(aq) + S^{2-}(aq)$$

$$H_2O(l) \rightleftharpoons H^+(aq) + OH^-(aq)$$

The overall reaction occurring in the reactive stripping column is:

$$NaHS(aq) + CO_2(g) + H_2O(l) \rightleftharpoons NaHCO_3(aq) + H_2S(g)$$

2.2. The Reactive Stripping Process

Figure 1 shows a schematic drawing of the reactive stripping column:

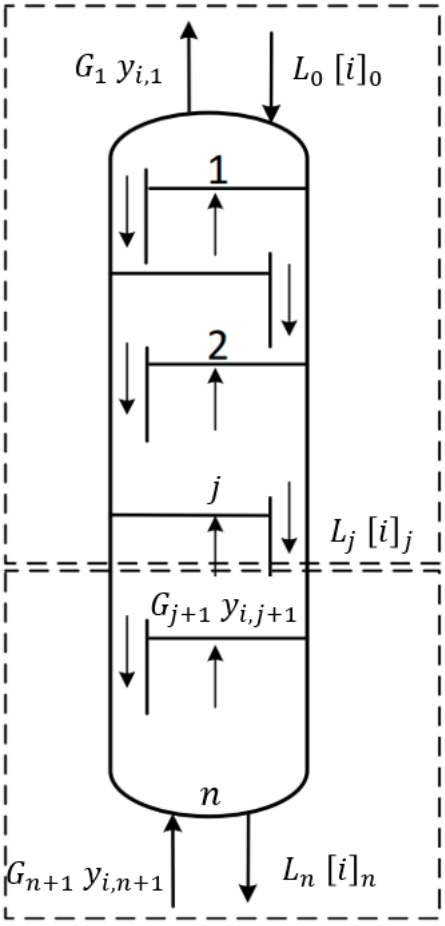


Figure 1: *Schematic Drawing of the Reactive Stripping Column*

The symbols L_j and G_j indicates respectively liquid and vapour streams flowing from stage j in the column. The symbols $[i]_j$ and $y_{i,j}$ indicate concentrations of specie i in the respective liquid or vapour stream flowing from stage j.

We also note that in *Figure 1*, the stages are labelled from stage 1 to stage n, starting at the top of the stripping column. Likewise, we will solve the stripping column starting at the top – this will greatly simplify the route toward the solution.

2.3. Liquid-Side Mass Transfer Coefficients

We obtain the Sherwood number from (Nock, et al., 2015):

$$Sh = \frac{k_L\, d_e}{D_L} \Longrightarrow k_L = \frac{Sh\, D_L}{d_e} \qquad (Eqn.1)$$

Where k_L is the liquid-side mass transfer coefficient in [m/s], d_e is the bubble diameter in [m], and D_L is the liquid diffusivity of the gas in the liquid in [m^2/s].

From (Nock, et al., 2015) we make use of the Higbie assumption for a mobile bubble surface moving through a volume of fluid:

$$Sh = 1.13\sqrt{Re\, Sc} \qquad (Eqn.2)$$

$$Re = \frac{d_e\, u_b\, \rho_L}{\mu_L} \qquad (Eqn.3)$$

$$Sc = \frac{\mu_L}{\rho_L\, D_L} \qquad (Eqn.4)$$

Where u_b is the rising velocity of the bubble in [m/s], ρ_L is the liquid density in [kg/m^3], μ_L is the dynamic viscosity of the liquid in [Pa.s], and Re and Sc are the dimensionless Reynold's and Scmidt numbers, respectively.

We now obtain an expression for k_L:

$$k_L = 2\sqrt{\frac{D_L\, u_b}{\pi\, d_e}} \qquad (Eqn.5)$$

From (Guo, et al., 2018), we obtain an expression for the rising velocity for a bubble in a volume of liquid. We also assume that the bubble diameter will be greater than 1.3 mm and will therefore be of an ellipsoidal shape when it moves through the liquid.

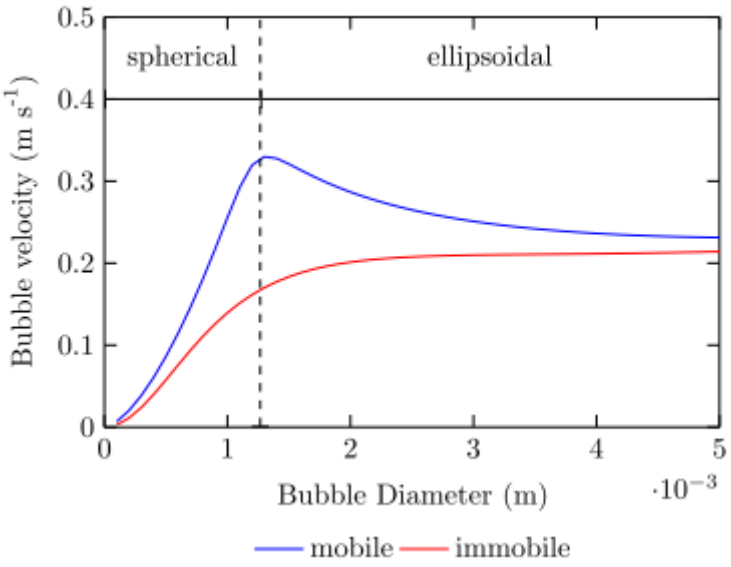


Figure 2: *Bubble Rise Velocity vs. Diameter for Mobile and Immobile Bubbles*

$$u_b = \sqrt{2.14\frac{\sigma_L}{\rho_L\, d_e} + 0.505\, g\, d_e} \qquad for \quad d_e > 1.3\, mm \qquad (Eqn.6)$$

Where u_b is the rising velocity of the bubble in [m/s], ρ_L is the liquid density in [kg/m^3], d_e is the bubble diameter in [m], σ_L is the surface tension for the liquid in [N/m], and g is the gravitational acceleration constant (which is equal to 9.81 m/s^2).

We now look at the specific interfacial area, a. The expression for a is determined as follows:

$$a = \frac{total\ interfacial\ area}{total\ volume\ of\ the\ liquid}$$

The units for a will therefore be in [m^{-1}]. For a system with bubbles of droplets, a can be expressed as:

$$a = \frac{6\,\varepsilon}{d_e} \qquad (Eqn.7)$$

Where ε represents the gas-holdup or fraction of the reactor (or tray, for this project) volume occupied by gas. This term is also known as the gas fraction for the reacting system.

We are now able to write an expression for the volumetric mass transfer coefficient in the form of:

$$(k_L a) = 2a\sqrt{\frac{D_L\, u_b}{\pi\, d_e}} \qquad (Eqn.8)$$

Where the volumetric mass transfer coefficient, $k_L a$, has units of [s^{-1}].

2.4. Temperature Dependent Variables

In order to correctly simulate a model for the reactive stripping column at a specific temperature and eventually perform a sensitivity analysis on the model at various temperatures, we must make use of temperature dependent functions for the constants that govern the reaction equilibria, reaction rates, and mass transfer processes in the system.

The following temperature dependent relationships are gathered:

2.4.1. Dissociation and Ionization Constants

In Soli & Byrne, 2002, the first dissociation constant (K_{1C} in [mol/L]) for H_2CO_3 as a function of temperature in [K] (for 15 °C ≤ T ≤ 32.5 °C, at P = 1 atm) is determined as:

$$logK_{1C}(T) = -0.994 - \frac{610.5}{T} \qquad (Eqn.\,9)$$

In Millero, et al., 2002, the second dissociation constant (K_{2C} in [mol/L]) for H_2CO_3 as a function of temperature in [K] (for 0 °C ≤ T ≤ 40 °C, at P = 1 atm) and salinity is determined as:

$$\begin{aligned} -logK_{2C}(T,S) &= pK_{2C}(T,S) \\ &= -452.0940 + 13.142162\,S - (8.101 \times 10^{-4})S^2 \\ &+ \frac{21263.61}{T} + 68.483143\,lnT \\ &+ \frac{(-581.4428\,S + 0.259601\,S^2)}{T} - 1.967035\,S\,lnT \end{aligned} \qquad (Eqn.\,10)$$

For this relationship we will take S = 0 to obtain a temperature only dependent relationship for K_{2C}:

$$\begin{aligned} -logK_{2C}(T) &= pK_{2C}(T) \\ &= -452.0940 + \frac{21263.61}{T} + 68.483143\,lnT \end{aligned} \qquad (Eqn.\,11)$$

In Millero, 2003, the first dissociation constant (K_{1S} in [mol/L]) for H_2S as a function of temperature in [K] (for 0 °C ≤ T ≤ 300 °C, at P = 1 atm) is determined as:

$$\begin{aligned} -logK_{1S}(T) &= pK_{1S}(T) \\ &= 32.55 + \frac{1519.44}{T} - 15.672\,logT + 0.02722\,T \end{aligned} \qquad (Eqn.\,12)$$

In Stephens & Cobble, 1970, the second dissociation constant (K_{2S} in [mol/L]) for H_2S as a function of temperature in [K] (for 0 °C ≤ T ≤ 100 °C, at P = 1 atm) is determined as:

$$-logK_{2S}(T) = pK_{2S}(T) = \frac{4500}{T} + 12.6\ log\left(\frac{T}{298.15}\right) - 1.29 \qquad (Eqn.\ 13)$$

In Olofsson & Hepler, 1975, the ionization constant for water (K_W in [mol^2/L^2]) as a function of temperature in [K] (for 0 °C ≤ T ≤ 300 °C, at P = 1 atm) is determined as:

$$\begin{aligned} -logK_W(T) =\ & pK_W(T) \\ =\ & \frac{142613.6}{T} + 4229.195\ logT - 9.7384\ T \\ & + 0.0129638\ T^2 - (1.15068 \times 10^{-5})T^3 \\ & + (4.602 \times 10^{-9})T^4 - 8909.483 \end{aligned} \qquad (Eqn.\ 14)$$

2.4.2. Reaction Rate Constant

In Soli & Byrne, 2002, the forward reaction rate constant (k_f in [s^{-1}]) for the rate limiting equilibrium reaction of dissolved CO_2 with water as a function of temperature in [K] (for 15 °C ≤ T ≤ 32.5 °C) is determined as:

$$lnk_f(T) = 22.66 - \frac{7799}{T} \qquad (Eqn.\ 15)$$

Therefore, the first order rate expression for the rate limiting equilibrium reaction of dissolved CO_2 with water would be:

$$r_f = k_f[CO_2(aq)] \qquad (Eqn.\ 16)$$

2.4.3. Liquid Properties

In Perry's Chemical Engineering Handbook, 9th Edition, p.2-280, the dynamic viscosity (μ_L in [Pa.s]) for liquid water as a function of temperature in [K] (at P = 1 atm) is determined as:

$$\mu_L(T) = e^{\left[-52.843 + \frac{3703.6}{T} + 5.866\ lnT - (5.879 \times 10^{-29})T^{10}\right]} \qquad (Eqn.\ 17)$$

In Perry's Chemical Engineering Handbook, 9th Edition, p.2-99, the molar density (ρ_L in [mol/dm^3] later converted to [kg/m^3]) of liquid water as a function of temperature in [K] (at P = 1 atm) for a temperature range of 273.16 K ≤ T ≤ 353.15 K (0.01 °C ≤ T ≤ 80 °C) is determined as:

$$\rho_L(T) = -13.851 + 0.64038\ T - 0.0019124\ T^2 + 1.8211 \times 10^{-6}\ T^3 \qquad (Eqn.\ 18)$$

In Vargaftik, et al., 1983, the surface tension (σ_L in [N/m]) of liquid water as a function of temperature in [K] (at P = 1 atm) is determined as:

$$\sigma_L(T) = (235.8 \times 10^{-3}) \left[\frac{647.15 - T}{647.15}\right]^{1.256} \left[1 - 0.625 \left(\frac{647.15 - T}{647.15}\right)\right] \qquad (Eqn.\ 19)$$

2.4.4. Water Vapor Pressure

Form NIST Webbook, the Antoine coefficients (A, B, C) for the Antoine equation needed to obtain the vapor pressure (P^* in [bar] later converted to [atm]) of water as a function of temperature in [K] (for -17 °C ≤ T ≤ 100 °C) is determined as:

$$A = 4.6543$$

$$B = 1435.264$$

$$C = -64.848$$

$$\log(P^*) = A - \frac{B}{T + C} \qquad (Eqn.\ 20)$$

2.4.5. Henry Constants

Form NIST Webbook, the Henry constant for CO_2 (k'_{H,CO_2} in [mol/kg.bar] later converted to [mol/kg.atm]) in water as a function of temperature in [K] (for 0 °C ≤ T ≤ 30 °C, at P = 1 atm) is determined as:

$$k'_{H,CO_2}(T) = 0.034\ e^{2600\left(\frac{1}{T} - \frac{1}{298.15}\right)} \qquad (Eqn.\ 21)$$

Form NIST Webbook, the Henry constant for H_2S (k_{H,H_2S} in [mol/kg.bar] later converted to [mol/kg.atm]) in water as a function of temperature in [K] (for 0 °C ≤ T ≤ 30 °C, at P = 1 atm) is determined as:

$$k_{H,H_2S}(T) = 0.10\ e^{2300\left(\frac{1}{T} - \frac{1}{298.15}\right)} \qquad (Eqn.\ 22)$$

2.4.6. Liquid Diffusivities

From the 9th Edition of Perry's Chemical Engineering Handbook, it is stated that we must use the following expression to determine the effect of temperature on the liquid diffusivity of a gas x in the liquid L:

$$\frac{D_{L,x}(T)\,\mu_L(T)}{T} = \lambda_x = constant \qquad (Eqn.\ 23)$$

In Perry's Chemical Engineering Handbook, 9th Edition, p.2-285, we obtain the expression for the liquid diffusivity for CO_2 (D_{L,CO_2} in [cm²/s] later converted to [m²/s]) in water as a function of temperature in [K] (at P = 1 atm) as:

$$D_{L,CO_2}(T) = \frac{\lambda_{CO_2}\,T}{\mu_L(T)} \qquad (Eqn.\ 24)$$

where,

$$\lambda_{CO_2} = \frac{D_{L,CO_2}(@25\ ℃)\,\mu_L(@25\ ℃)}{(25\ ℃ + 273.15)} = 6 \times 10^{-15} \qquad (Eqn.\ 25)$$

and,

$$D_{L,CO_2}(@25\ ℃) = 1.96 \times 10^{-5}\ cm^2/s$$

In Perry's Chemical Engineering Handbook, 9th Edition, p.2-285, we also obtain the expression for the liquid diffusivity for H_2S (D_{L,H_2S} in [cm²/s] later converted to [m²/s]) in water as a function of temperature in [K] (at P = 1 atm) as:

$$D_{L,H_2S}(T) = \frac{\lambda_{H_2S}\,T}{\mu_L(T)} \qquad (Eqn.\ 26)$$

where,

$$\lambda_{H_2S} = \frac{D_{L,H_2S}(@25\ ℃)\,\mu_L(@25\ ℃)}{(25℃ + 273.15)} = 4.928 \times 10^{-11} \qquad (Eqn.\ 27)$$

and,

$$D_{L,H_2S}(@25\ ℃) = 1.61 \times 10^{-5}\ cm^2/s$$

2.5. Predicting K_{eq} at Various Temperatures

Instead of researching a possible temperature dependent relationship to calculate the equilibrium constant for the rate limiting equilibrium reaction of dissolved CO_2 with water, a well-fitted approximation was obtained. This was done by making use of experimental data points that were used by a chemical engineering student, Mr. Gontse Mokonyane, from the University of Pretoria in the previous year (2024). These values were obtained from Soli & Byrne, 2002.

Table 1: *Data points for $(1/K_{eq})$ vs Temperature*

Temperature (°C)	15.0	17.5	20.0	22.5	25.0	27.5	30.0	32.5
$1/K_{eq}$	841	840	840	843	846	850	866	878

These data points were then plotted, and a simple Python program was written to closely fit an exponential function to the data points.

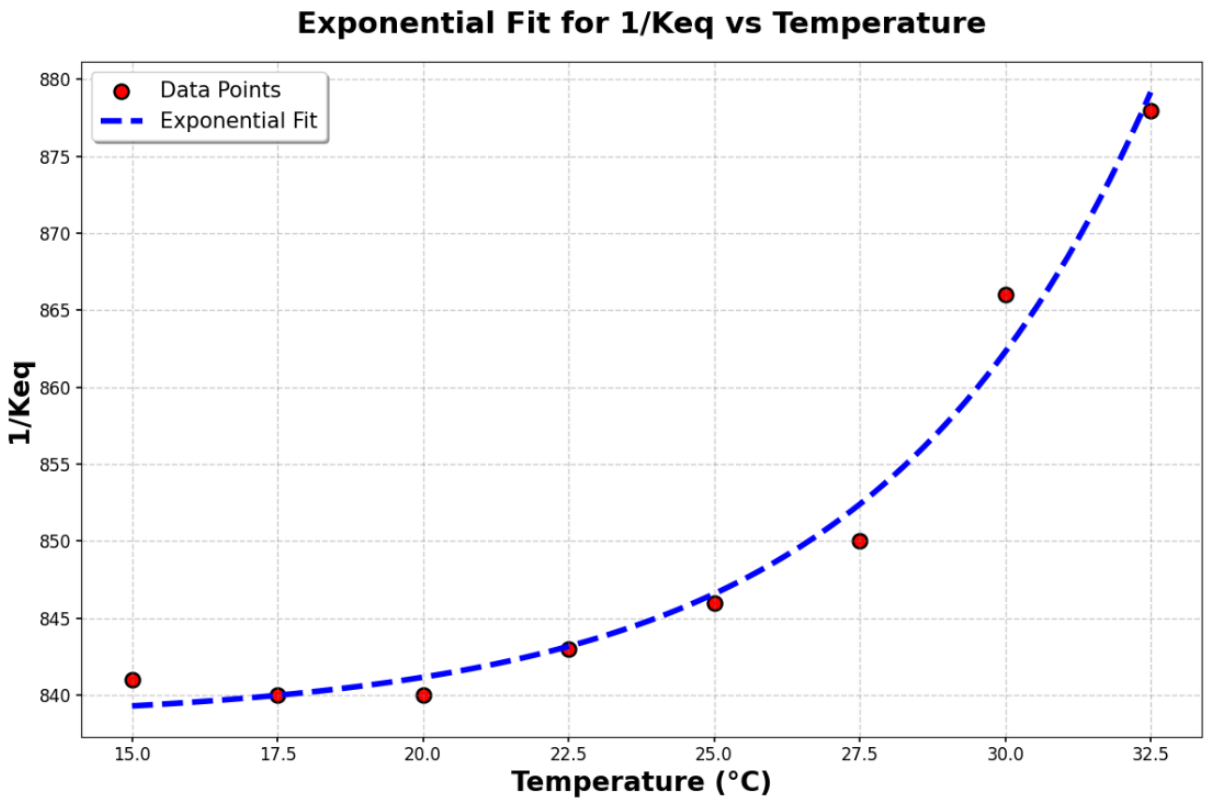


Figure 3: *Exponential Fit for $(1/K_{eq})$ vs Temperature*

The exponential fitted relationship for $1/K_{eq}$ as a function of temperature in [°C] was determined to be:

$$\frac{1}{K_{eq}(T)} = 0.040209\, e^{0.213053\,\mathrm{T}} + 838.300799 \qquad (Eqn.\ 28)$$

The Python program used to determine this relationship, is presented in detail in the Appendix *A3. Python Code for Determining K_{eq} as a Function of Temperature*.

2.6. The Reactive Stripping Column Model

The following equations and balances were derived to solve for the reactive stripping column, starting at the first stage at the top of the column.

For the first stage of the column only, we have the overall H_2S recovery for the column which predicts the amount of H_2S leaving the column in the vapor stream:

$$G_1 y_{H_2S,1} = L\,[NaHS]_0\, x_{H_2S} \qquad (Eqn.\ 29)$$

We have the CO_2 balance:

$$G_{j+1}\, y_{CO_2,j+1} + L\big([CO_2(aq)]_{j-1} + [H_2CO_3]_{j-1} + [HCO_3^-]_{j-1} + [CO_3^{2-}]_{j-1}\big)$$
$$= G_j\, y_{CO_2,j} + L\big([CO_2(aq)]_j + [H_2CO_3]_j + [HCO_3^-]_j + [CO_3^{2-}]_j\big) \qquad (Eqn.\ 30)$$

We have the sulphur balance:

$$G_{j+1}\, y_{H_2S,j+1} + L\big([H_2S(aq)]_{j-1} + [HS^-]_{j-1} + [S^{2-}]_{j-1}\big)$$
$$= G_j\, y_{H_2S,j} + L\big([H_2S(aq)]_j + [HS^-]_j + [S^{2-}]_j\big) \qquad (Eqn.\ 31)$$

We have the charge balance:

$$[H^+]_j + [Na^+] = [OH^-]_j + [HCO_3^-]_j + 2[CO_3^{2-}]_j + [HS^-]_j + 2[S^{2-}]_j \qquad (Eqn.\ 32)$$

We describe the CO_2 reaction rate with:

$$k_f\left([CO_2(aq)]_j - \frac{[H_2CO_3]_j}{K_{eq}}\right)V(1-\varepsilon) = L\left([H_2CO_3]_j + [HCO_3^-]_j + [CO_3^{2-}]_j\right) - L\left([H_2CO_3]_{j-1} + [HCO_3^-]_{j-1} + [CO_3^{2-}]_{j-1}\right) \quad (Eqn.\ 33)$$

We know that for any number of vapor mole fractions:

$$y_{CO_2,j+1} + y_{H_2S,j+1} + y_{H_2O} = 1 \quad (Eqn.\ 34)$$

For the mass transfer of CO_2 in water we have:

$$k_{L,CO_2}a\,V\left([CO_2(i)]_j - [CO_2(aq)]_j\right) = G_{j+1}\,y_{CO_2,j+1} - G_j\,y_{CO_2,j} \quad (Eqn.\ 35)$$

Similarly for the mass transfer of H_2S in water we have:

$$k_{L,H_2S}a\,V\left([H_2S(i)]_j - [H_2S(aq)]_j\right) = G_{j+1}\,y_{H_2S,j+1} - G_j\,y_{H_2S,j} \quad (Eqn.\ 36)$$

If we add the two mass transfer equations, we have:

$$G_{j+1} = G_j + \frac{k_{L,CO_2}a\,V\left([CO_2(i)]_j - [CO_2(aq)]_j\right) + k_{L,H_2S}a\,V\left([H_2S(i)]_j - [H_2S(aq)]_j\right)}{1 - y_{H_2O}} \quad (Eqn.\ 37)$$

For the equilibrium expression of the reaction, $H_2CO_3(aq) \rightleftharpoons H^+(aq) + HCO_3^-(aq)$, we have:

$$K_{1C} = \frac{[H^+]_j\,[HCO_3^-]_j}{[H_2CO_3]_j} \quad (Eqn.\ 38)$$

For the equilibrium expression of the reaction, $HCO_3^-(aq) \rightleftharpoons H^+(aq) + CO_3^{2-}(aq)$, we have:

$$K_{2C} = \frac{[H^+]_j\,[CO_3^{2-}]_j}{[HCO_3^-]_j} \quad (Eqn.\ 39)$$

For the equilibrium expression of the reaction, $H_2S(aq) \rightleftharpoons H^+(aq) + HS^-(aq)$, we have:

$$K_{1S} = \frac{[H^+]_j \, [HS^-]_j}{[H_2S(aq)]_j} \qquad (Eqn.\ 40)$$

For the equilibrium expression of the reaction, $HS^-(aq) \rightleftharpoons H^+(aq) + S^{2-}(aq)$, we have:

$$K_{2S} = \frac{[H^+]_j \, [S^{2-}]_j}{[HS^-]_j} \qquad (Eqn.\ 41)$$

For the equilibrium expression of the reaction, $H_2O(l) \rightleftharpoons\ H^+(aq) + OH^-(aq)$, we have:

$$K_W = [H^+]_j \, [OH^-]_j \qquad (Eqn.\ 42)$$

For the equilibrium between the CO_2 gas and the dissolved CO_2 at the gas-liquid interface:

$$[CO_2(i)]_j = k'_{H,CO_2} \, P_{CO_2} = k'_{H,CO_2} \, y_{CO_2,j} \, P_{tot} \qquad (Eqn.\ 43)$$

For the equilibrium between the H_2S gas and the dissolved H_2S at the gas-liquid interface:

$$[H_2S(i)]_j = k_{H,H_2S} \, P_{H_2S} = k_{H,H_2S} \, y_{H_2S,j} \, P_{tot} \qquad (Eqn.\ 44)$$

Important to mention, we calculate the total hydrogen ion concentration in a stage from pH of that stage using:

$$[H^+]_j = 10^{-pH_j} \qquad (Eqn. 45)$$

3. Method

3.1. Degrees of Freedom Analysis

If the sets of equations (from $Eqn.30$ to $Eqn.44$, excluding $Eqn.34$ because it is used to derive $Eqn.37$) that must be solved for each stage, e.g. stage j, starting from the top of the reactive stripping column, all the gas flow leaving stage j and its composition are known and the liquid flow rate from stage j-1 and its composition is known. *Table 2* shows the variables that must be calculated:

Table 2: *Variables to be Calculated for the System*

1	G_{j+1}
2	$[CO_2(aq)]_j$
3	$[H^+]_j$
4	$[OH^-]_j$
5	$[H_2CO_3]_j$
6	$[HCO_3^-]_j$
7	$[CO_3^{2-}]_j$
8	$[H_2S(aq)]_j$
9	$[HS^-]_j$
10	$[S^{2-}]_j$
11	$[CO_2(i)]_j$
12	$[H_2S(i)]_j$
13	$y_{CO_2,j+1}$
14	$y_{H_2S,j+1}$

We note that all $[Na^+]$ and y_{H_2O} are constant and known for each stage because we know $[NaHS]_{in}$ and we also know $T_{operating}$ and $P_{operating}$ for the system.

$$DOF = \#unknown\ variables - \#equations$$

$$DOF = 14 - 14 = 0$$

Thus, the problem is exactly specified.

3.1. Methodology for the Reactive Stripping Model

We will use the following methodology to solve the first stage for the reactive stripping column. Once we have the solution to the first stage, will be able to solve the second stage in the column, and so forth, until we reach the 'stopping criterion' which we will define in *3.2. Stopping Criterion*.

In order to solve for the first stage in the reactive stripping column we must:

Step-1)

We define values for $T_{operating}$, $P_{operating}$, L, V, $[NaHS]_0$, ε, G_1, d_e, and x_{H2S}.

Step-2)

We calculate all temperature dependent constants for the system at a specified $T_{operating}$. These constants are K_{eq}, K_{1C}, K_{2C}, K_{1S}, K_{2S}, K_W, P_{H_2O}, $k_{L,CO_2}a$, $k_{L,H_2S}a$, k_f, k'_{H,CO_2}, and k_{H,H_2S}.

Step-3)

We use Dalton's Law to calculate y_{H_2O}:

$$y_{H_2O} = \frac{P_{H_2O}}{P_{operating}} \qquad (Eqn.\ 46)$$

Step-4)

Using $Eqn.\ 29$, we calculate $y_{H_2S,1}$ which can be seen as $y_{H_2S,j}$. $Eqn.\ 29$ rearranged is:

$$y_{H_2S,1} = \frac{L\ [NaHS]_0\ x_{H_2S}}{G_1}$$

Step-5)

We use $Eqn.\ 34$ calculate $y_{CO_2,1}$ which can be seen as $y_{CO_2,j}$. $Eqn.\ 34$ rearranged is:

$$y_{CO_2,j+1} = 1 - y_{H_2S,j+1} - y_{H_2O}$$

Step-6)

We guess appropriate values for the pH, $[H_2CO_3]_j$, and $[H_2S(aq)]_j$. The values of eleven of the fourteen variables can be calculated if the values of these three variables are known.

Step-7)

Using $Eqn.38$, we calculate $[HCO_3^-]_j$. $Eqn.38$ rearranged is:

$$[HCO_3^-]_j = \frac{K_{1C}\,[H_2CO_3]_j}{[H^+]_j}$$

Step-8)

Using $Eqn.39$, we calculate $[CO_3^{2-}]_j$. $Eqn.39$ rearranged is:

$$[CO_3^{2-}]_j = \frac{K_{2C}\,[HCO_3^-]_j}{[H^+]_j}$$

Step-9)

Using $Eqn.40$, we calculate $[HS^-]_j$. $Eqn.40$ rearranged is:

$$[HS^-]_j = \frac{K_{1S}\,[H_2S(aq)]_j}{[H^+]_j}$$

Step-10)

Using $Eqn.41$, we calculate $[S^{2-}]_j$. $Eqn.41$ rearranged is:

$$[S^{2-}]_j = \frac{K_{2S}\,[HS^-]_j}{[H^+]_j}$$

Step-11)

Using $Eqn.45$, we calculate $[H^+]_j$.

Step-12)

Using $Eqn.\,42$, we calculate $[OH^-]_j$. $Eqn.\,42$ rearranged is:

$$[OH^-]_j = \frac{K_W}{[H^+]_j}$$

Step-13)

Using $Eqn.\,43$, we calculate $[CO_2(i)]_j$.

Step-14)

Using $Eqn.\,44$, we calculate $[H_2S(i)]_j$.

Step-15)

Using $Eqn.\,33$, we calculate $[CO_2(aq)]_j$. $Eqn.\,33$ rearranged is:

$$[CO_2(aq)]_j = \frac{[H_2CO_3]_j}{K_{eq}} + \frac{L}{k_f\, V(1-\epsilon)}\left\{\left([H_2CO_3]_j + [HCO_3^-]_j + [CO_3^{2-}]_j\right) - \left([H_2CO_3]_{j-1} + [HCO_3^-]_{j-1} + [CO_3^{2-}]_{j-1}\right)\right\}$$

Step-16)

Using $Eqn.\,37$, we calculate G_{j+1}.

Step-17)

Using $Eqn.\,35$, we calculate $y_{CO_2,j+1}$. $Eqn.\,35$ rearranged is:

$$y_{CO_2,j+1} = \frac{G_j\, y_{CO_2,j} + k_{L,CO_2}\, a\, V\left([CO_2(i)]_j - [CO_2(aq)]_j\right)}{G_{j+1}}$$

Step-18)

Using $Eqn.36$, we calculate $y_{H_2S,j+1}$. $Eqn.36$ rearranged is:

$$y_{H_2S,j+1} = \frac{G_j\, y_{H_2S,j} + k_{L,H_2S} a\, V\left([H_2S(i)]_j - [H_2S(aq)]_j\right)}{G_{j+1}}$$

Step-19)

We now use the sum of the squares of the CO_2 balance ($Eqn.30$) residual, the sulphur balance ($Eqn.31$) residual, and the charge balance ($Eqn.32$) residual to verify whether we have obtained the solution to the stage in the column.

CO_2 balance residual (CDB):

$$CDB = G_j\, y_{CO_2,j} + L\left([CO_2(aq)]_j + [H_2CO_3]_j + [HCO_3^-]_j + [CO_3^{2-}]_j\right) - G_{j+1}\, y_{CO_2,j+1} - L\left([CO_2(aq)]_{j-1} + [H_2CO_3]_{j-1} + [HCO_3^-]_{j-1} + [CO_3^{2-}]_{j-1}\right) \quad (Eqn.47)$$

Sulphur balance residual (SB):

$$SB = G_j\, y_{H_2S,j} + L\left([H_2S(aq)]_j + [HS^-]_j + [S^{2-}]_j\right) - G_{j+1}\, y_{H_2S,j+1} - L\left([H_2S(aq)]_{j-1} + [HS^-]_{j-1} + [S^{2-}]_{j-1}\right) \quad (Eqn.48)$$

Charge balance residual (CB):

$$CB = [OH^-]_j + [HCO_3^-]_j + 2\,[CO_3^{2-}]_j + [HS^-]_j + 2[S^{2-}]_j - [H^+]_j - [Na^+] \quad (Eqn.\ 49)$$

Sum of the squares of the balance residuals (SS):

$$SS = CDB^2 + SB^2 + CB^2 \quad (Eqn.\ 50)$$

The goal is to finally minimize $Eqn.50$ for a range of guess values for pH, $[H_2CO_3]_j$, and $[H_2S(aq)]_j$ – once the value for SS is minimized we would have finally solved the stage in the reactive stripping column.

We note that when solving the CO_2 balance, the sulphur balance, and the charge balance, convergence problems may be experienced because the value of $[H^+]_j$ may be several orders of magnitude lower than the values of $[H_2CO_3]_j$ and $[H_2S(aq)]_j$. This is known as a set of 'stiff' equations. In an attempt to lessen the 'stiffness' of the system of equations, we will instead guess values of pH, $1000 \times [H_2CO_3]_j$, and $1000 \times [H_2S(aq)]_j$ and account for the scaling in the equations.

3.2. Stopping Criterion

To ensure that we do not produce solutions to infinite stages in the reactive stripping column, we must introduce a 'stopping criterion'. We will look toward completing the overall mass balance using the H_2S recovery.

We will define the target value for the total sulphur concentration remaining in the liquid phase leaving the column as:

$$target\ sulphur\ remaining = [NaHS]_0 \left(1 - x_{H_2S}\right) \qquad (Eqn. 51)$$

The overall mass balance using the H_2S recovery will then be complete.

After iteratively solving a stage in the reactive stripping column, we will compute the total sulphur leaving the stage in the liquid phase as:

$$total\ sulphur\ out = [H_2S(aq)]_j + [HS^-]_j + [S^{2-}]_j \qquad (Eqn. 52)$$

After a solution is found at a stage in the column, we will compare $Eqn. 51$ and $Eqn. 52$ with the inequality:

$$total\ sulphur\ out \leq total\ sulphur\ out$$

In the program the moment that this inequality is no longer true, we know that the overall mass balance using the H_2S recovery is sufficiently complete, since solving a next stage will result in removing more sulphur that than is available to be removed. Therefore, the reactive stripping column is now solved

3.3. Basis and Assumptions

3.3.1. Basis

- The system will be analyzed on a per second basis, i.e. liquid and vapor flowrates will be in units of [L/s] and [mol/s], respectively.
- The reactive stripping column operates under isobaric conditions (P = 1 atm).
- The reactive stripping column operates under isothermal conditions, and will be initially simulated at T = 25 °C.

- The reactive stripping column will be initially simulated with the liquid feed stream consisting of 1 L/s of water containing 0.8 mol $NaHS$.
- There is no carbonate present in the liquid feed stream.
- There is no H_2S or S^{2-} present in the liquid feed stream.
- There is no H_2S present in the gas feed stream.
- The CO_2 fed to the column is saturated with water vapour thus y_{H_2O} = 0.0314
- The reactive stripping column will be simulated with a gas hold-up of ε = 5 %.
- The reactive stripping column will be simulated with a bubble diameter of d_e = 5 mm.
- The reactive stripping column will be initially simulated with an outlet vapor flowrate of G_1 = 0.9 mol/s.
- The reactive stripping column will be initially simulated with a stage volume of V = 180 L.
- The reactive stripping column will be initially simulated with a H_2S recovery of x_{H_2S} = 99.99 % in the outlet vapor stream.

3.3.2. Assumptions

- The volumetric flow rate of liquid phase flowing down the column is constant and equal to the volume of liquid fed to the column (L = 1 L/s).
- The vapour pressure of water in the vapour stream is constant throughout the column.
- The vapour pressure of water is not affected by changes in the concentrations of the different dissolved species in the liquid stream.
- The sum of the molar concentrations of sulphur compounds i.e. $H_2S(aq)$, HS^-, and S^{2-}, in the liquid feed stream is equal to the molar concentration of Na^+ in the liquid feed stream, and therefore:

$$L([H_2S(aq)]_0 + [HS^-]_0 + [S^{2-}]_0) = L\,[Na^+]_0 = L\,[NaHS]_0 \qquad (Eqn.\ 53)$$

$$\therefore [HS^-]_0 = [Na^+]_0 = [NaHS]_0 \qquad (Eqn.\ 54)$$

- The equilibrium, dissociation, and ionization constants are not affected by changes in the concentrations of the different dissolved species in the liquid stream.
- The volume of each stage in the reactive stripping column is the same for all stages.
- The gas hold-up for each stage in the reactive stripping column is the same for all stages (ε = 5 %).
- The bubbles rising through the liquid phase in each stage in the reactive stripping column have a constant average diameter (d_e = 5 mm).

3.4. Computational Methods

The simulation of the H_2S stripping column employs a stage-wise equilibrium model with kinetic limitations for the CO_2 hydration reaction. The computational approach involves solving a system of nonlinear algebraic equations for each stage, representing mass transfer and equilibrium phenomena. The model framework incorporates interfacial equilibrium through Henry's law relationships for gas-liquid interface concentrations of H_2S and CO_2, while simultaneously considering multiple acid-base equilibria including carbonic acid dissociation (K_{1C} and K_{2C}), hydrogen sulfide dissociation (K_{1S} and K_{2S}), water autoionization (K_W), and CO_2 hydration equilibrium (K_{eq}). A critical aspect of the model is the inclusion of kinetic limitations through the forward rate constant (k_f) for the slow CO_2 hydration reaction, which cannot be assumed to be at equilibrium.

The column is solved sequentially from top to bottom using a counter-current flow configuration, where each stage requires the simultaneous solution of three fundamental balance equations – charge balance ensuring electroneutrality of the liquid phase, carbon dioxide material balance accounting for CO_2 transfer between phases and chemical reactions, and hydrogen sulfide material balance tracking H_2S stripping and speciation changes. The convergence strategy employs variable transformation using pH, scaled H_2CO_3 concentration (×1000), and scaled H_2S concentration (×1000) as primary variables to improve numerical stability. Multiple initial guess combinations are tested to ensure robust convergence, with solutions validated against physical constraints including realistic pH ranges (4 – 14) and positive concentration requirements. The simulation features automatic stage determination, continuing until the target H_2S recovery (99.99 %) is achieved, with convergence criteria based on remaining sulphur content in the liquid phase.

3.4.1. Software and Tools

The model development and simulation are implemented using *Python 3.13.3* as the primary programming language, with *Jupyter Notebook* serving as the interactive development environment for iterative model refinement and analysis. The core scientific computing is supported by several essential libraries, with *SciPy* providing the fundamental numerical solver through its *scipy.optimize.fsolve* function for solving systems of nonlinear equations. *NumPy* serves as the foundation for numerical computations, array operations, and mathematical functions, while *Pandas* handles data manipulation and analysis, particularly for organizing stage-wise results into structured *DataFrames* that facilitate post-processing and analysis.

Visualization and results analysis are accomplished using *Matplotlib*, which provides comprehensive plotting capabilities for concentration profiles, equilibrium validations, and convergence diagnostics. The model incorporates extensive mathematical and

physical property calculations through custom-built functions that handle temperature-dependent correlations for liquid surface tension, density, viscosity, mass transfer coefficients for CO_2 and H_2S, Henry's law constants, vapor pressure calculations, and bubble dynamics including rising velocity calculations. These physical property functions are essential for accurate representation of the mass transfer phenomena occurring in the stripping column.

3.4.2. Numerical Methods Used in Python

The core numerical method employed is the *scipy.optimize.fsolve* function, which utilizes a hybrid Powell method that combines the efficiency of Newton's method with the global convergence properties of the Levenberg-Marquardt algorithm. High precision convergence criteria are implemented with tolerance settings of 10^{-12} to ensure accurate mass and energy balance closure. The solution strategy involves residual minimization, where the best solution is selected based on minimizing the sum of squares of equation residuals across multiple initial guesses.

The nonlinear equation system for each stage is formulated through a systematic approach where scaled variables are converted back to physical concentrations, derived concentrations are calculated using equilibrium relationships, and balance equations are formulated and returned as residuals. Numerical stability is enhanced through several techniques including variable scaling where concentrations are multiplied by factors of 1000 to improve numerical conditioning, logarithmic transformations using pH instead of H^+ concentration to handle the wide range of hydrogen ion concentrations, and systematic testing of different starting points to avoid local minima and ensure global convergence.

The implementation features robust error handling through try-catch blocks around the equation solver to manage convergence failures, comprehensive solution validation checking for physical reasonableness and mass balance closure, and detailed convergence diagnostics that track and report residual values for each balance equation. Data management is accomplished through dictionary-based storage of liquid and gas phase properties for each stage, with subsequent conversion to *Pandas DataFrames* for efficient analysis and visualization, enabling systematic tracking of concentration profiles, flow rates, and convergence diagnostics throughout the column height.

4. Results

Table 1 shows the input and system variables that were used to obtain the following results from the developed Python program (attached in Appendix *A4. Python Program for Reactive Stripping Column Model*) for the reactive stripping column model:

Table 3: *Input and System Variables to Generate Results*

Variable	Value	Units
$T_{operating}$	25	°C
$P_{operating}$	1	atm
V_{stage}	180	L
L	1	L/s
G_1	0.9	mol/s
ε	5	%
d_e	5	mm
$[NaHS]_0$	0.8	mol/L
x_{H_2S}	99.99	%

4.1. Results for Equilibrium and System Constants

Table 4: *Results for Equilibrium and System Constants*

Constant	Value	Units
K_{eq}	1.18123 x10^{-3}	unitless
K_{1C}	9.08600 x10^{-4}	mol/L
K_{2C}	3.85631 x10^{-10}	mol/L
K_{1S}	1.04105 x10^{-7}	mol/L
K_{2S}	1.57371 x10^{-14}	mol/L
K_W	1.00194 x10^{-14}	mol^2/L^2
k_f	0.030259	s^{-1}
a	60	m^{-1}
σ_L	0.071976	N/m
ρ_L	997.0408	kg/m^3
μ_L	9.12531 x10^{-4}	Pa.s
u_b	0.23255	m/s
$k_{L,CO_2}\,a$	0.020441	s^{-1}

$k_{L,H_2S}a$	0.018527	s^{-1}
k'_{H,CO_2}	0.034349	mol/L.atm
k_{H,H_2S}	0.10103	mol/L.atm
P_{vap,H_2O}	0.031378	atm
y_{H_2O}	0.031378	unitless

4.2. Total Sulphur and Carbon Species Concentrations

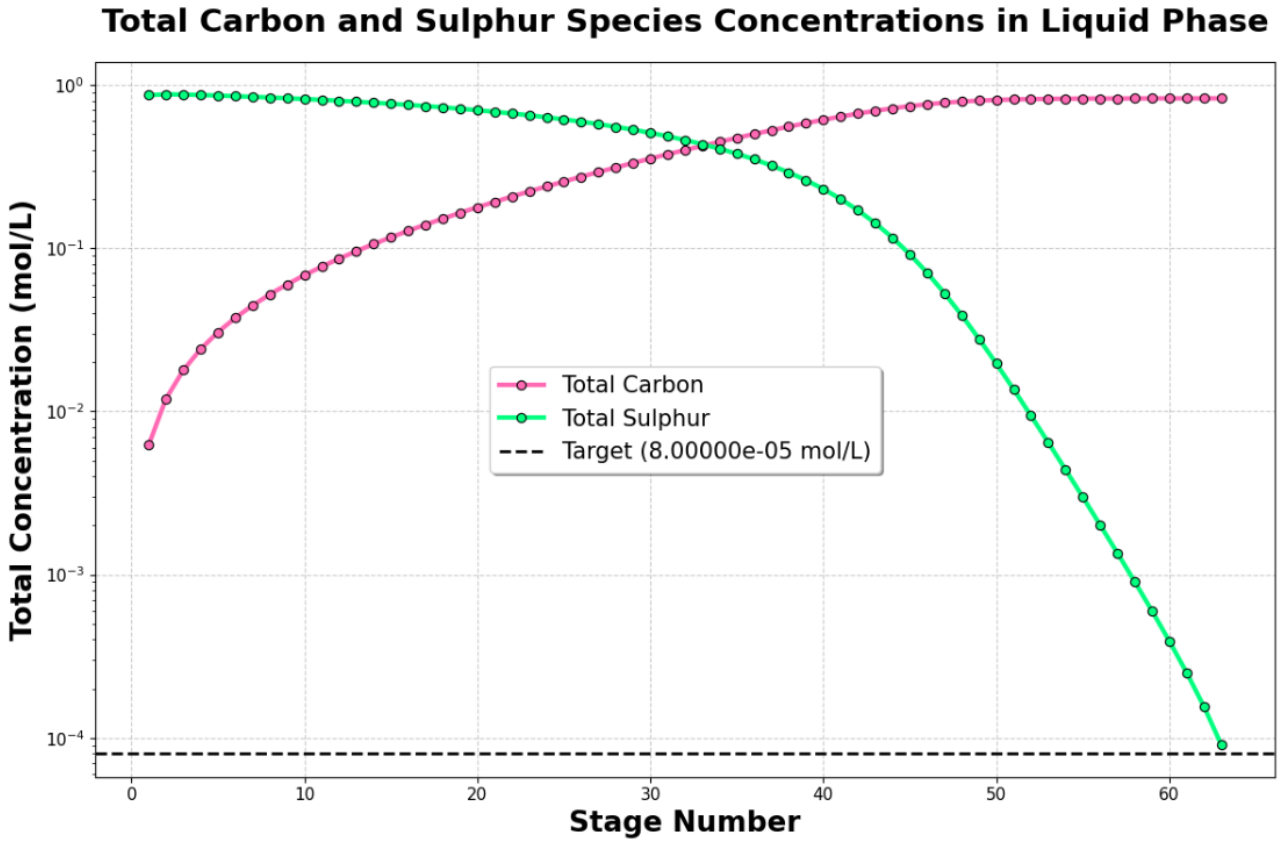


Figure 4: *Total Carbon an Sulphur Species Concentrations*

Figure 4 demonstrates the fundamental mass transfer behaviour in the stripping column. The total sulphur concentration (green line) decreases from approximately 0.8 mol/L at the feed stage to the target recovery level of 8×10^{-5} mol/L around stage 60, representing the progressive stripping of H_2S from the liquid phase into the gas phase. Conversely, the total carbon concentration (pink line) increases down the column from near zero to approximately 0.8 mol/L, indicating CO_2 absorption from the stripping gas into the liquid phase. This counter-current behaviour is expected since CO_2 is being used as the stripping agent, and as it contacts the alkaline $NaHS$ solution, it dissolves and forms carbonic species while simultaneously facilitating H_2S desorption.

4.3. Individual Sulphur and Carbon Species Concentrations

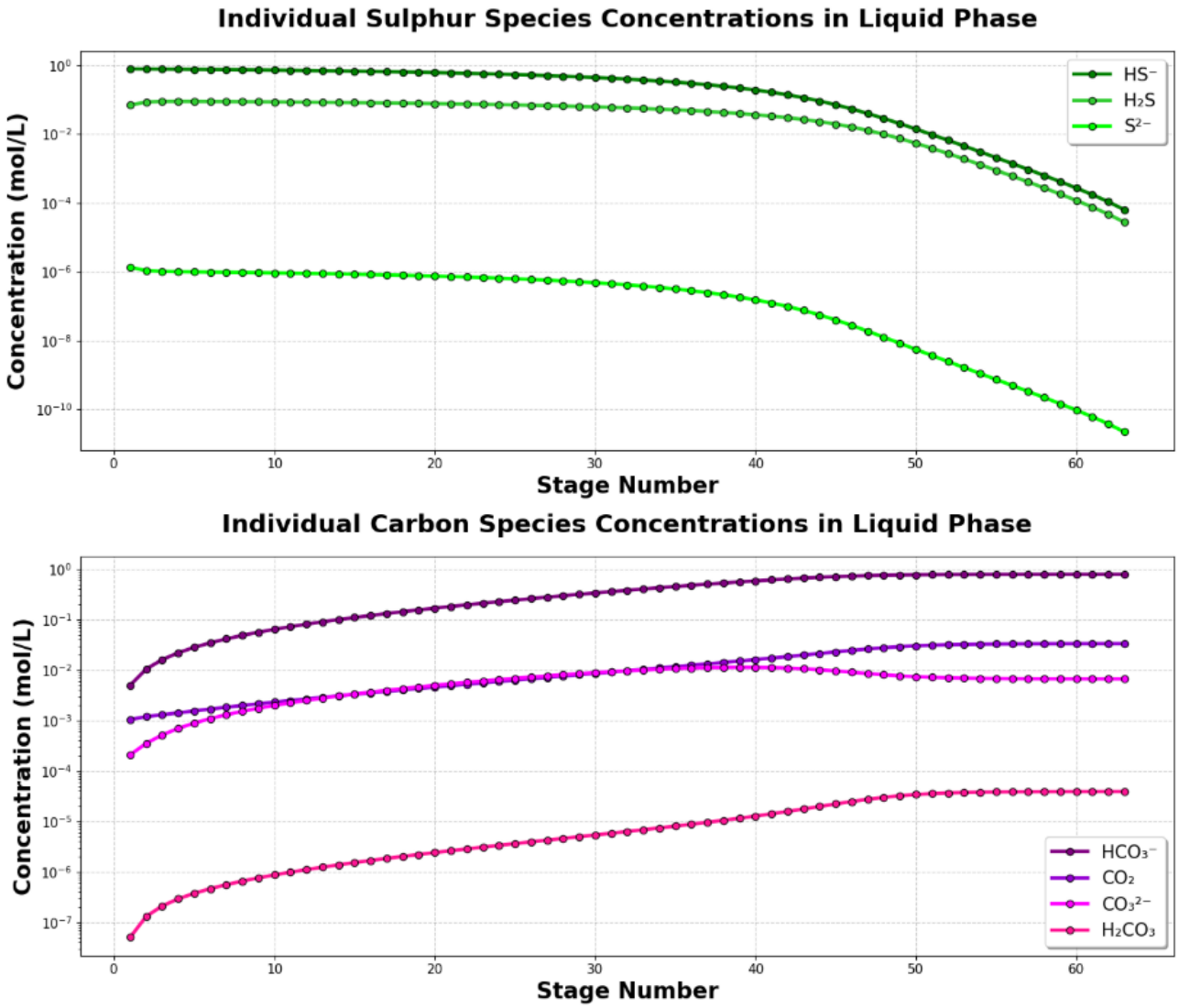


Figure 5: *Individual Sulphur and Carbon Species Concentrations*

The individual species profiles in *Figure 5* reveal the complex chemical equilibria occurring throughout the column. For sulphur species, HS^- dominates initially (around 0.8 mol/L from the $NaHS$ feed) and decreases steadily, while H_2S and S^{2-} remain at much lower concentrations due to the solution pH conditions. For carbon species, HCO_3^- becomes the dominant species as CO_2 dissolves and dissociates in the alkaline solution, increasing from zero to nearly 1 mol/L. The CO_3^{2-} and H_2CO_3 concentrations remain relatively constant at lower levels, while dissolved CO_2 shows a gradual increase. This speciation pattern is consistent with the pH-dependent equilibria of both the carbonic acid and hydrogen sulphide systems, where the kinetically limited CO_2 hydration reaction (modelled with rate constant k_f) controls the conversion between dissolved CO_2 and carbonic acid species.

4.4. Vapor Flowrate and Vapor Fractions

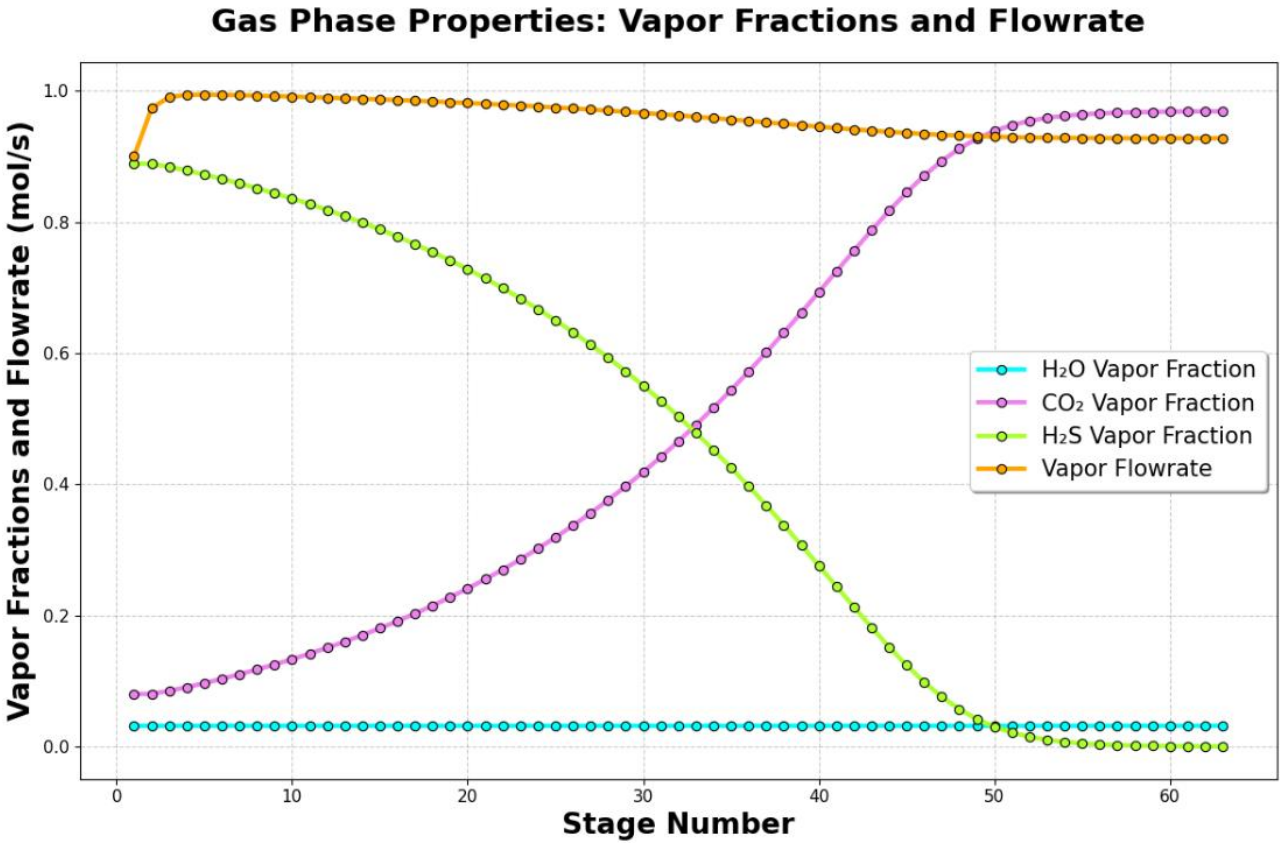


Figure 6: *Vapor Flowrate and Vapor Fractions*

In *Figure 6* the vapor fraction profiles show the expected behaviour for a stripping operation. The H_2S vapor fraction (green line) decreases dramatically from about 0.9 at the top to nearly zero around stage 50, indicating efficient stripping of H_2S from the liquid phase. Conversely, the CO_2 vapor fraction (purple line) increases from approximately 0.08 at the top to about 0.95 at the bottom, demonstrating that CO_2 is being absorbed into the liquid phase as it contacts the alkaline NaHS solution. The water vapor fraction remains essentially constant at a low level (~0.03) throughout the column, controlled by the vapor pressure equilibrium. The vapor flowrate (orange line) shows a slight increase from top to bottom due to the net mass transfer of CO_2 into the gas phase, which is typical for reactive absorption/stripping systems.

4.5. pH of the Model

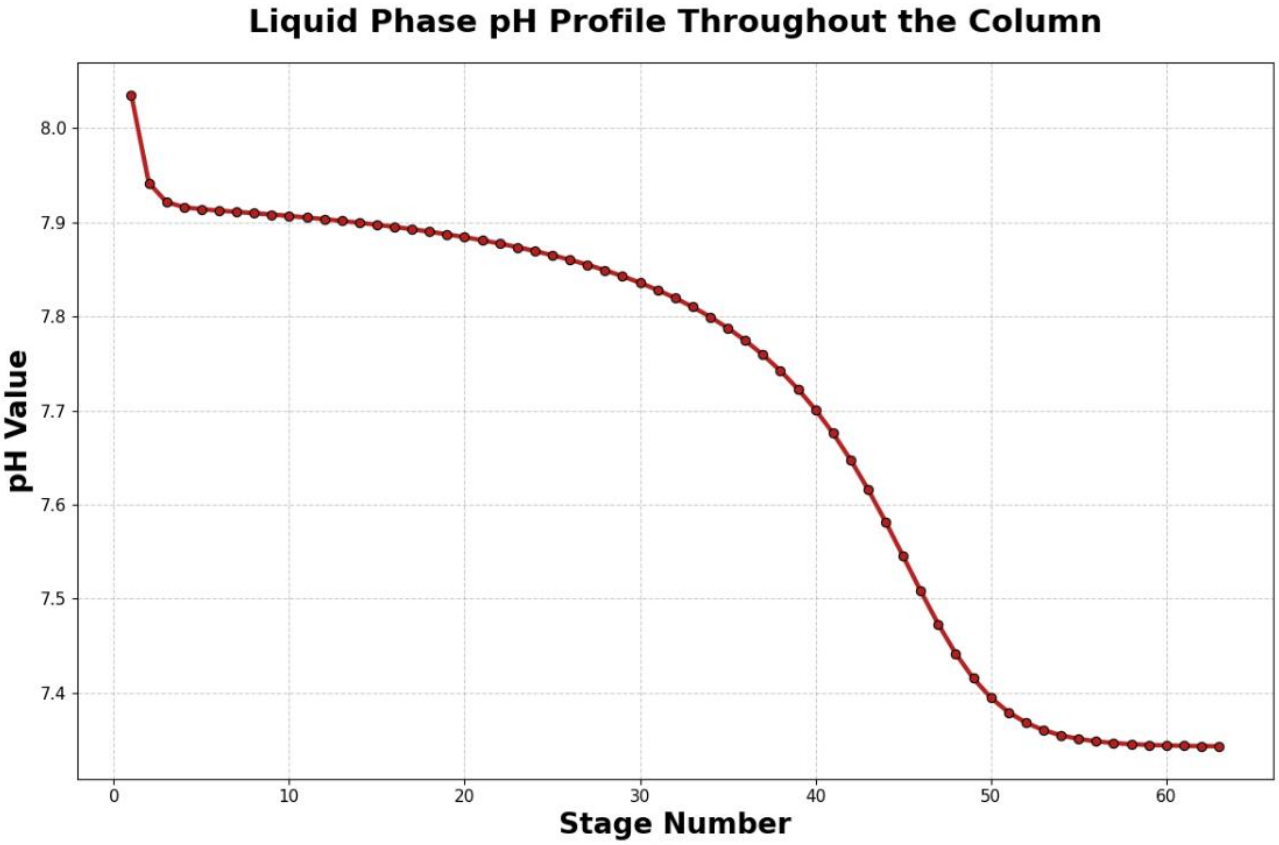


Figure 7: *pH-Values Throughout the Stripping Column*

The pH profile in *Figure 7* demonstrates the chemical driving force behind the stripping process. Starting at approximately pH 8.05 in the feed stage, the pH gradually decreases to about 7.35 at the bottom of the column. This pH decline occurs because CO_2 absorption forms carbonic acid species (H_2CO_3 and HCO_3^-) that consume hydroxide ions and lower the solution pH. The higher pH at the top favours the conversion of dissolved H_2S species (HS^-) to volatile H_2S gas, while the lower pH at the bottom reduces the driving force for further H_2S stripping. This pH gradient is crucial for process optimization, as it shows that the alkalinity of the feed solution is being neutralized by CO_2 absorption, which could limit the stripping efficiency if not properly managed. The relatively smooth pH transition indicates good stage-to-stage equilibrium is being approached despite the kinetic limitations of the CO_2 hydration reaction.

4.6. Accuracy of the Model

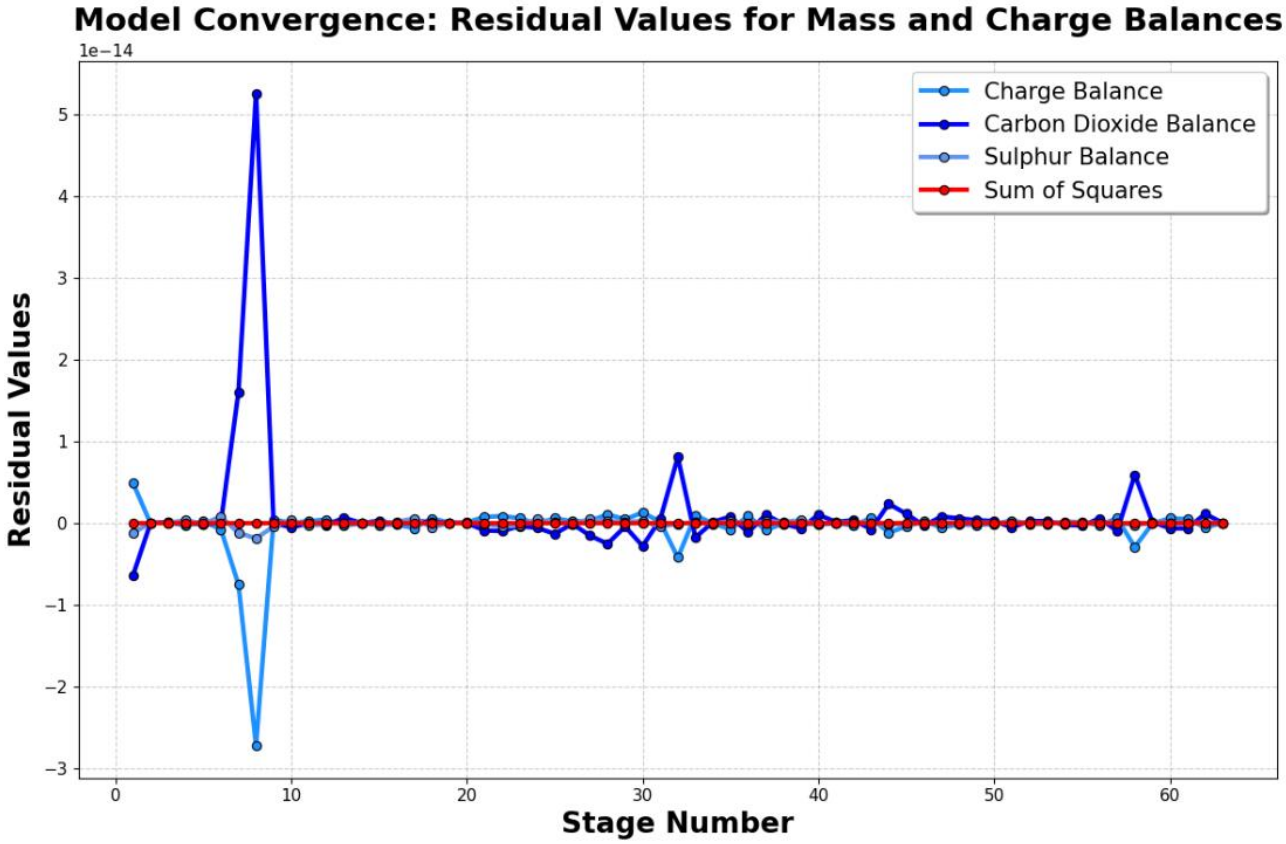


Figure 8: *Residual Values Indicating the Accuracy of the Solutions*

The charge balance residuals oscillate around zero with occasional spikes, indicating that electroneutrality is generally well-maintained but experiences temporary deviations at certain stages where ionic species concentrations change rapidly due to shifting acid-base equilibria.The carbon dioxide balance shows the most dramatic behaviour, with a massive spike around stage 8 reaching 5×10^{-14}. This occurs because the CO_2 hydration reaction has slow kinetics explicitly modelled in the system. At this stage, there's a significant driving force for CO_2 transfer, but the kinetic limitation prevents instantaneous equilibration, causing large mass balance residuals as the model accounts for the finite reaction rate.

The sulphur balance residuals remain stable near zero throughout the column because sulphide equilibria are much faster than CO_2 hydration kinetics, allowing these species to maintain near-equilibrium conditions without significant kinetic limitations. The sum of squares closely follows the CO_2 pattern, confirming that CO_2 kinetics dominate the numerical challenges. The return to near-zero residuals after the spike demonstrates that the solver successfully navigates the complex interplay between kinetic and equilibrium effects, achieving convergence that captures the true physical behaviour of this reactive mass transfer system.

5. Sensitivity Analyses on the Model

5.1. Varying Operating Temperature

Table 5 shows the input and system variables that were used to obtain the following results from the developed Python program (attached in Appendix *A5. Python Program for Operating Temperature Sensitivity Analysis*) for the reactive stripping column model during an operating temperature sensitivity analysis:

Table 5: *Input and System Variables to Operating Temperature Sensitivity Analysis*

Variable	Value	Units
$T_{operating}$	***15 - 47***	°C
$P_{operating}$	1	atm
V_{stage}	180	L
L	1	L/s
G_1	0.9	mol/s
ε	5	%
d_e	5	mm
$[NaHS]_0$	0.8	mol/L
x_{H_2S}	99.99	%

5.1.1. Number of Stages required

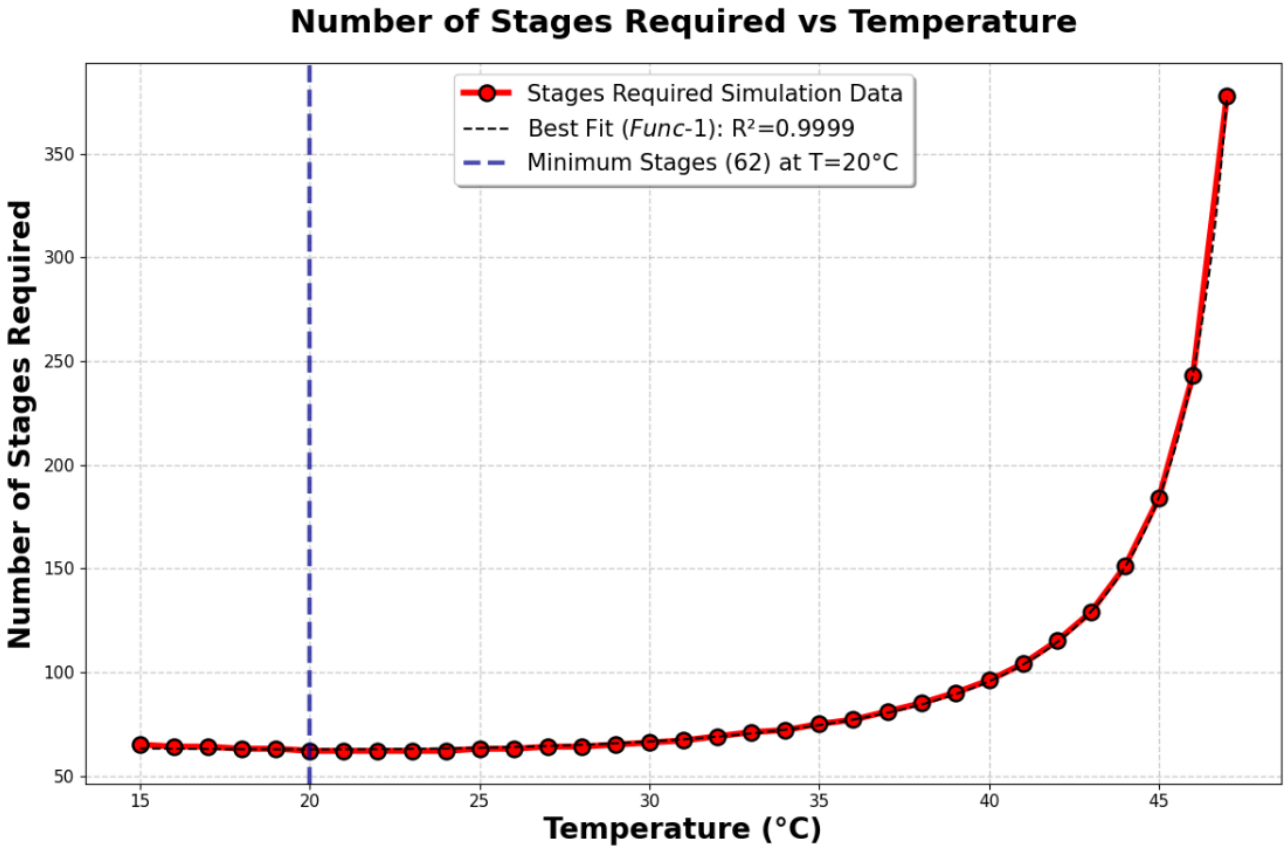


Figure 9: *Number of Stages Required with Varying Operating Temperature*

The increase in required stages at higher temperatures reflects unfavourable thermodynamic shifts that outweigh kinetic benefits. While higher temperatures improve mass transfer through enhanced diffusivity, they also cause water vapor pressure to rise, reducing the driving force for volatile component transfer. Additionally, decreasing Henry's law constants at elevated temperatures create less favourable gas-liquid equilibrium distributions.

The minimum at 20°C (62 stages) represents the optimal balance between kinetic and thermodynamic effects. Below this temperature, the process becomes kinetically limited by poor mass transfer, while above it, unfavourable vapor-liquid equilibria increasingly dominate, requiring more stages to achieve the same separation efficiency.

The accurately fitted empirical function, *Func-1*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*)

5.1.2. Bottom Stage pH Value

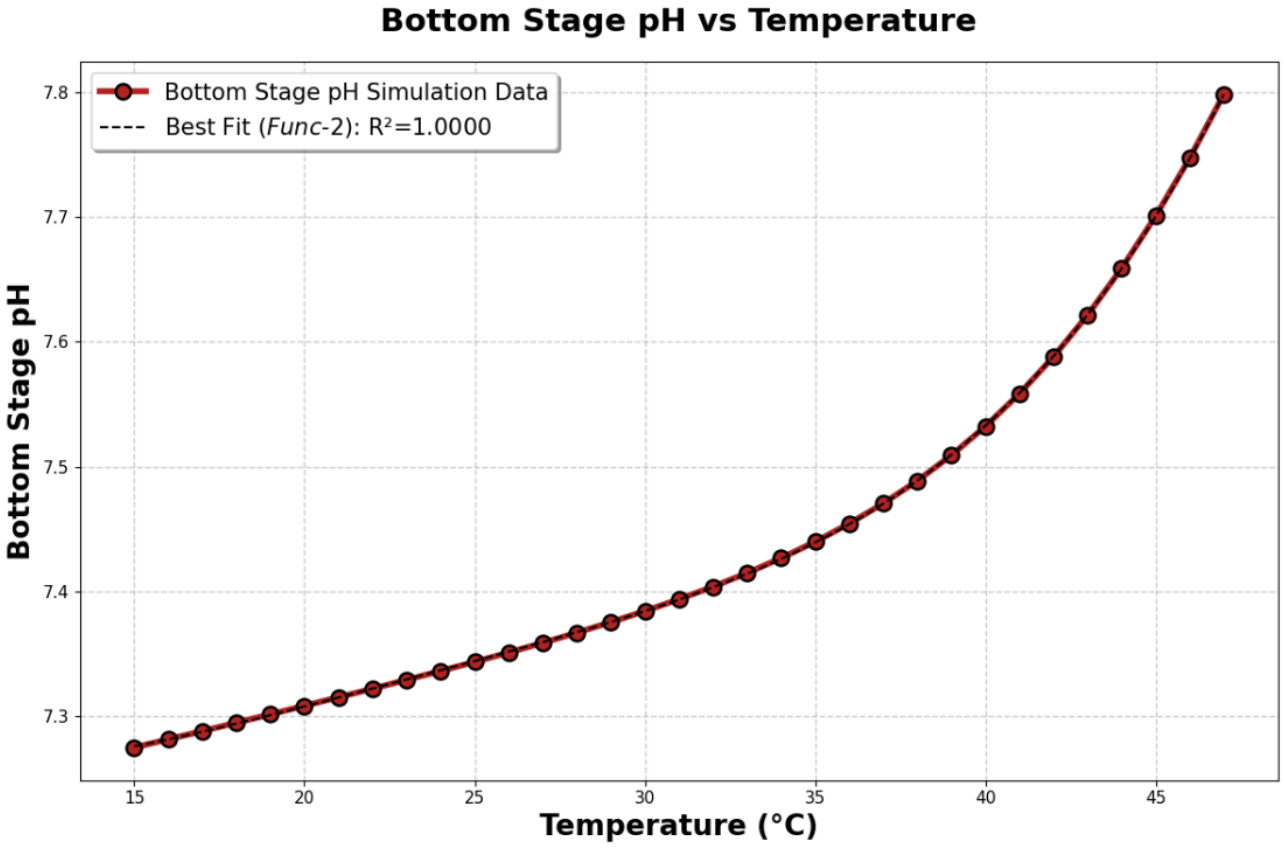


Figure 10: *Bottom Stage pH Against Varying Operating Temperature*

In *Figure 10* the steady increase in final pH with temperature is consistent with the temperature dependence of the various equilibrium constants in the system. As temperature rises, the ionization constants for H_2S (K_{1S} and K_{2S}) and the carbonic acid system (K_{1C} and K_{2C}) change in ways that favour more basic conditions. The water ionization constant (K_W) also increases with temperature, contributing to higher hydroxide concentrations. This pH shift affects the speciation of sulphur and carbon species, influencing the overall mass transfer driving forces.

The accurately fitted empirical function, *Func-2*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.1.3. Bottom Gas Flowrate and Mole Fractions

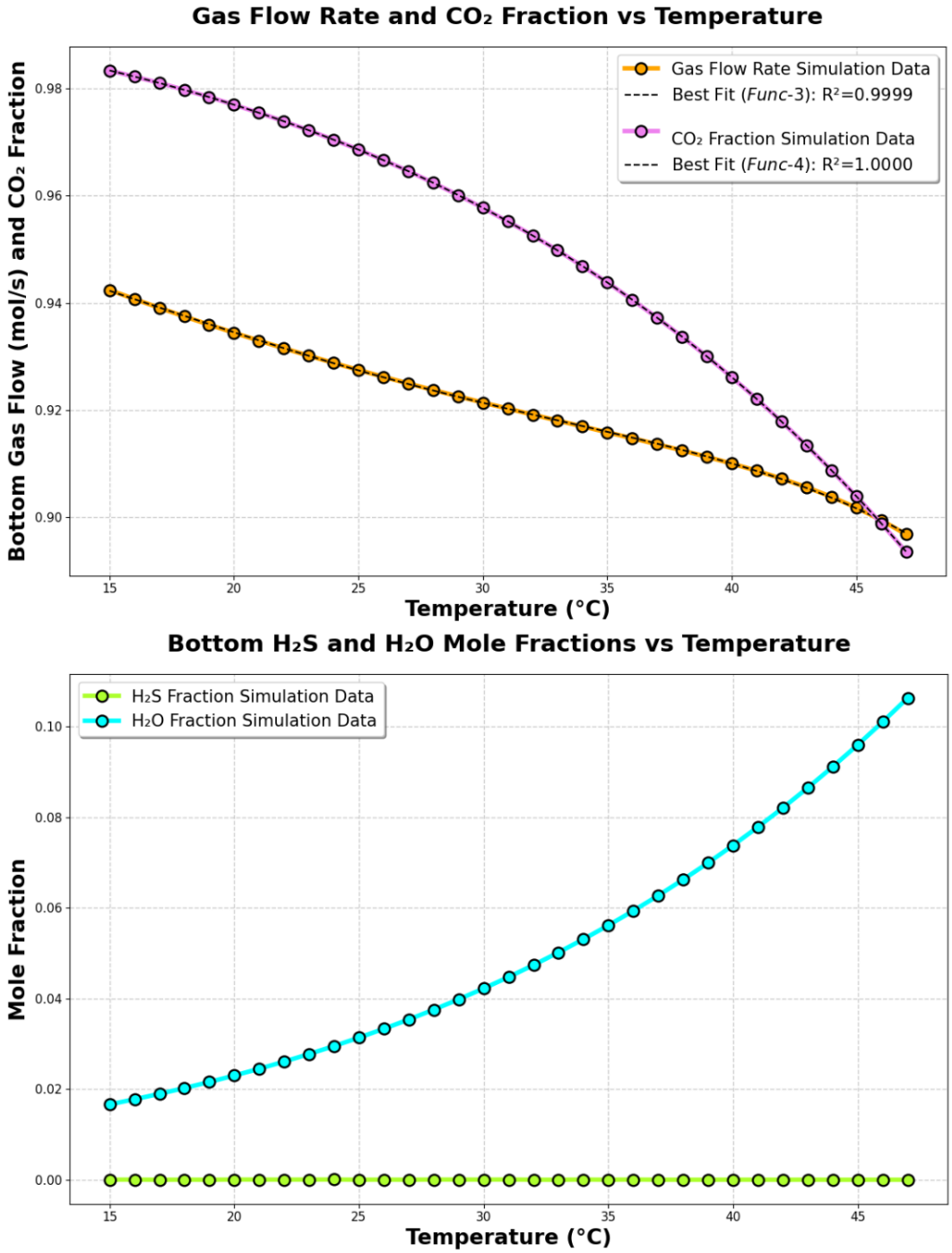


Figure 11: *Bottom Gas Analysis with Varying Operating Temperature*

In *Figure 11* the decreasing bottom gas flow rate and CO_2 fraction with increasing temperature reflects the changing vapor-liquid equilibrium and mass transfer characteristics. Higher temperatures reduce gas solubility, but they also increase water vapor pressure significantly, changing the overall gas composition. The CO_2 fraction decreases because more water vapor enters the gas phase, diluting other components.

The H_2S fraction remains relatively constant while the H_2O fraction increases dramatically, confirming that water vaporization becomes the dominant factor at higher temperatures, fundamentally altering the separation efficiency and requiring more stages to achieve the same H_2S recovery target.

For visualization of the system temperature dependent system properties and constants during the temperature sensitivity analysis, see Appendix *A2. System Properties and Constants Produced for Varying Temperature*.

The accurately fitted empirical functions, *Func-3* and *Func-4*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.2. Varying Operating Pressure

Table 6 shows the input and system variables that were used to obtain the following results from the developed Python program (attached in Appendix *A6. Python Program for Operating Pressure Sensitivity Analysis*) for the reactive stripping column model during an operating pressure sensitivity analysis:

Table 6: *Input and System Variables to Operating Pressure Sensitivity Analysis*

Variable	Value	Units
$T_{operating}$	25	°C
$P_{operating}$	***0.3 - 5***	atm
V_{stage}	180	L
L	1	L/s
G_1	0.9	mol/s
ε	5	%
d_e	5	mm
$[NaHS]_0$	0.8	mol/L
x_{H_2S}	99.99	%

5.2.1. Number of Stages required

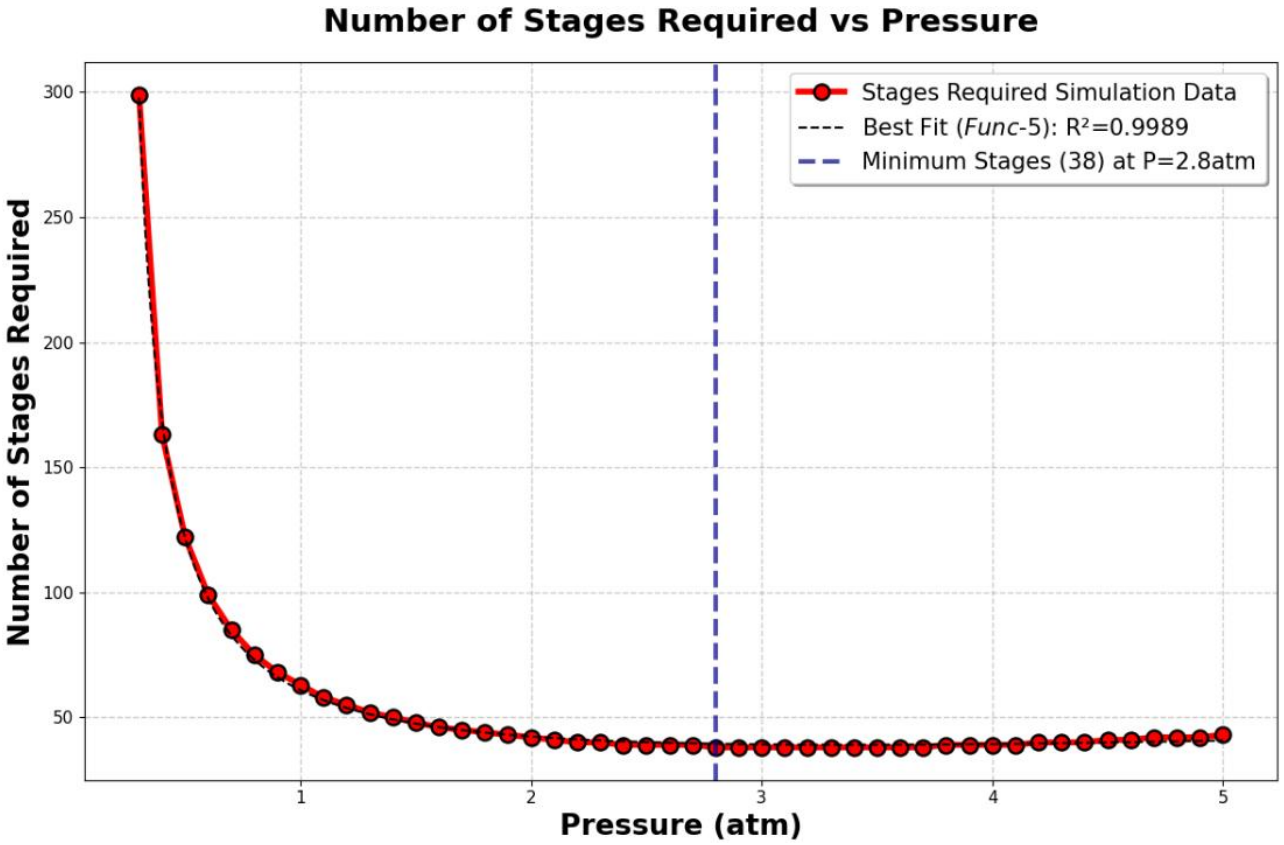


Figure 12: *Number of Stages Required with Varying Operating Pressure*

The dramatic decrease in required stages in *Figure 12* from nearly 300 stages at 0.3 atm to a minimum of 38 stages at 2.8 atm reflects the strong influence of pressure on gas-liquid equilibrium. At lower pressures, the Henry's law constants for both H_2S and CO_2 result in higher gas-phase concentrations, but the overall driving force for mass transfer is reduced due to lower interfacial concentrations. The minimum at 2.8 atm represents an optimal balance where the pressure is high enough to maintain reasonable interfacial concentrations while still providing sufficient driving force for stripping. Above this pressure, additional stages are needed because higher pressure reduces the volatility of both components, requiring more contact stages to achieve the same separation efficiency.

The accurately fitted empirical function, *Func-5*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.2.2. Bottom Stage pH Value

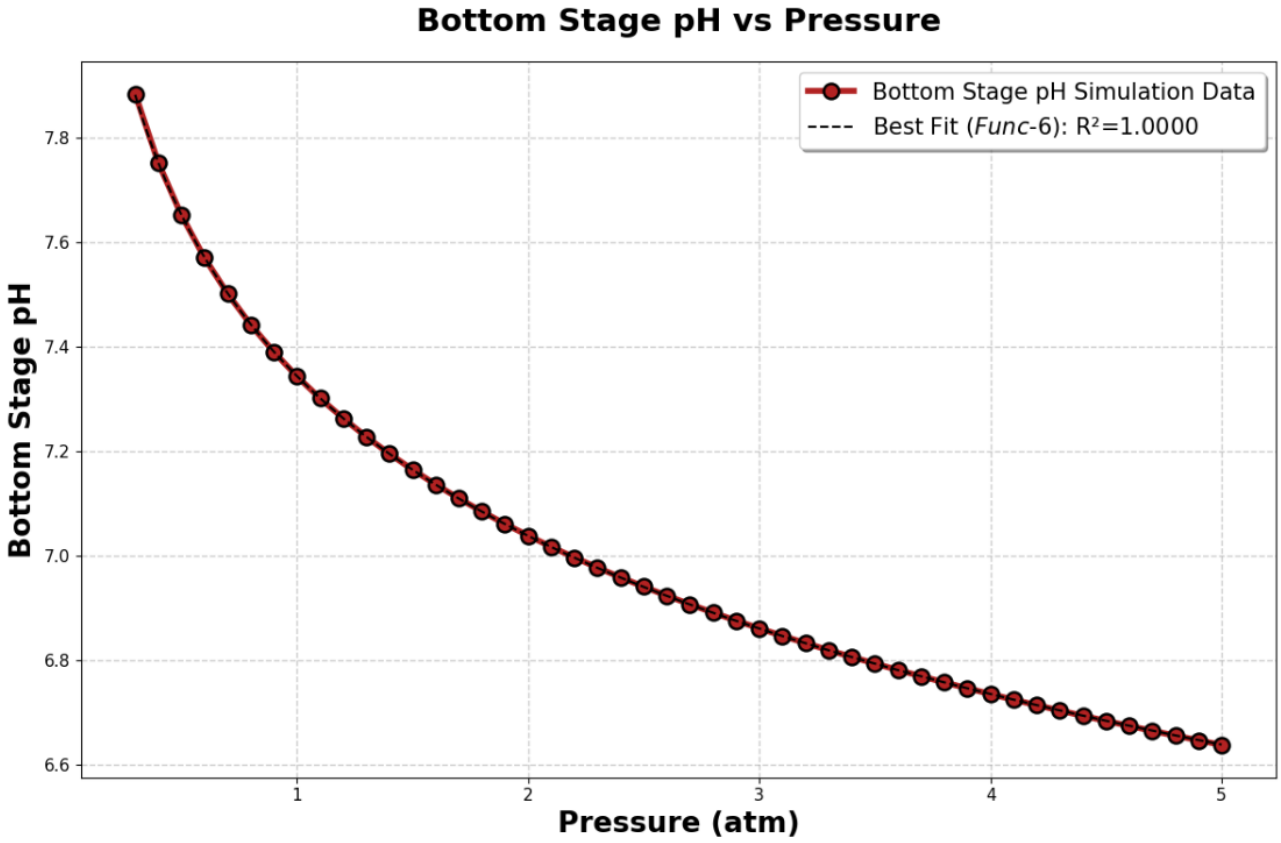


Figure 13: *Bottom Stage pH Against Varying Operating Pressure*

In *Figure 13* the steady decrease in final pH from 7.9 to 6.6 as pressure increases from 0.3 atm to 5 atm demonstrates the pressure-dependent nature of the carbonic acid system equilibrium. Higher pressures increase CO_2 solubility according to Henry's law, leading to greater carbonic acid formation and subsequent dissociation, which lowers the solution pH. This pH reduction also affects the H_2S/HS^- equilibrium, potentially impacting the overall stripping efficiency by shifting more sulfur species toward the ionic forms that are less volatile.

The accurately fitted empirical function, *Func-6*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.2.3. Bottom Gas Flowrate and Mole Fractions

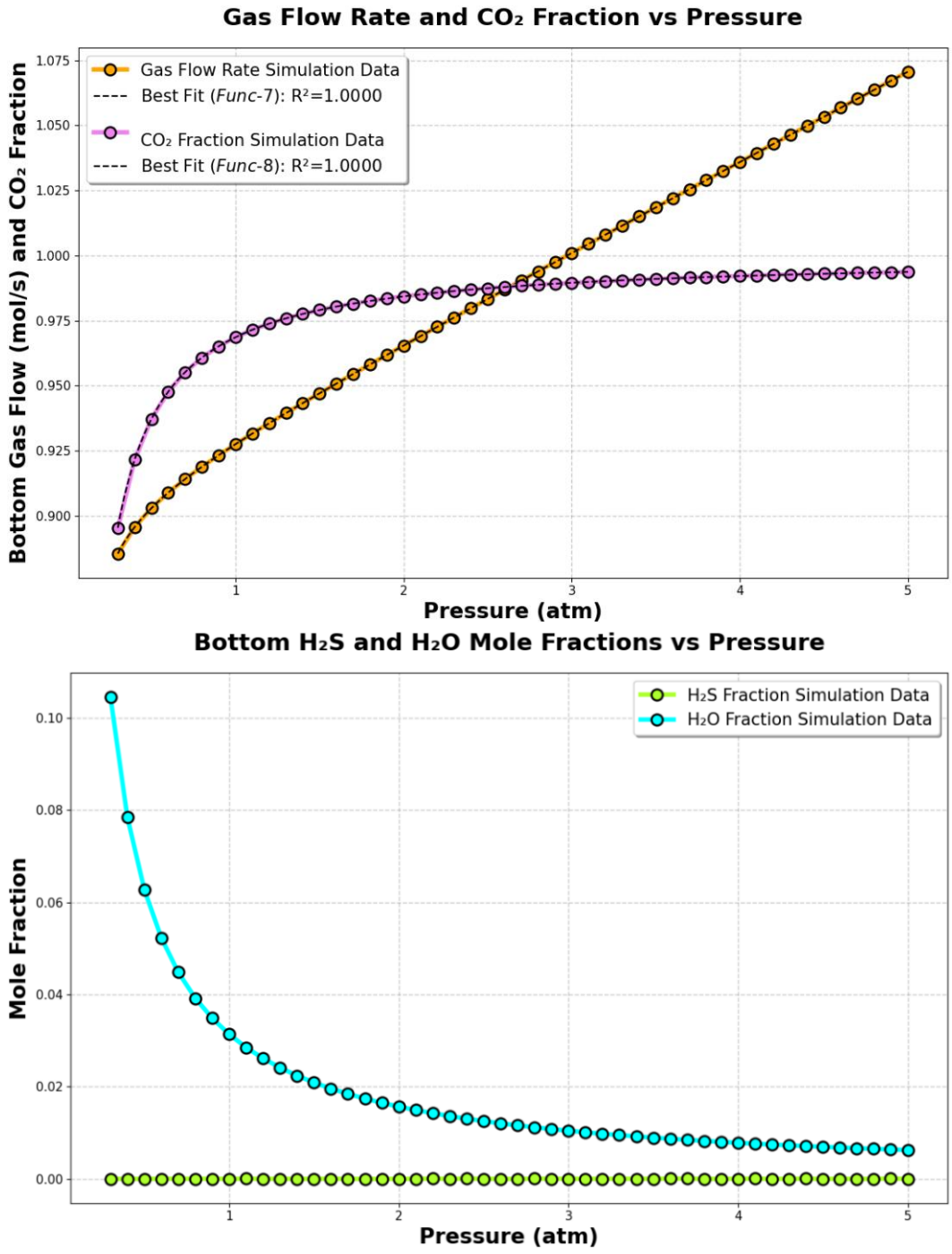


Figure 14: *Bottom Gas Analysis with Varying Operating Pressure*

The bottom gas flow rate in *Figure 14* increases monotonically with pressure due to enhanced mass transfer driving forces at higher interfacial concentrations, while the CO_2 mole fraction plateaus around 0.99 above 3 atm, indicating complete saturation of the stripping capacity for CO_2. Conversely, the H_2O mole fraction decreases significantly with pressure because water's vapor pressure remains constant while total pressure

increases, reducing its relative volatility. The H_2S fraction remains minimal throughout, confirming that the specified 99.99 % recovery target is being met across all pressures, though the absolute amounts stripped vary with the changing gas flow rates and equilibrium conditions.

The accurately fitted empirical functions, *Func-7* and *Func-8*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.3. Varying Stage Volume

Table 7 shows the input and system variables that were used to obtain the following results from the developed Python program (attached in Appendix *A7. Python Program for Stage Volume Sensitivity Analysis*) for the reactive stripping column model during a stage volume sensitivity analysis:

Table 7: *Input and System Variables to Stage Volume Sensitivity Analysis*

Variable	Value	Units
$T_{operating}$	25	°C
$P_{operating}$	1	atm
V_{stage}	***50 - 550***	L
L	1	L/s
G_1	0.9	mol/s
ε	5	%
d_e	5	mm
$[NaHS]_0$	0.8	mol/L
x_{H_2S}	99.99	%

5.3.1. Number of Stages Required

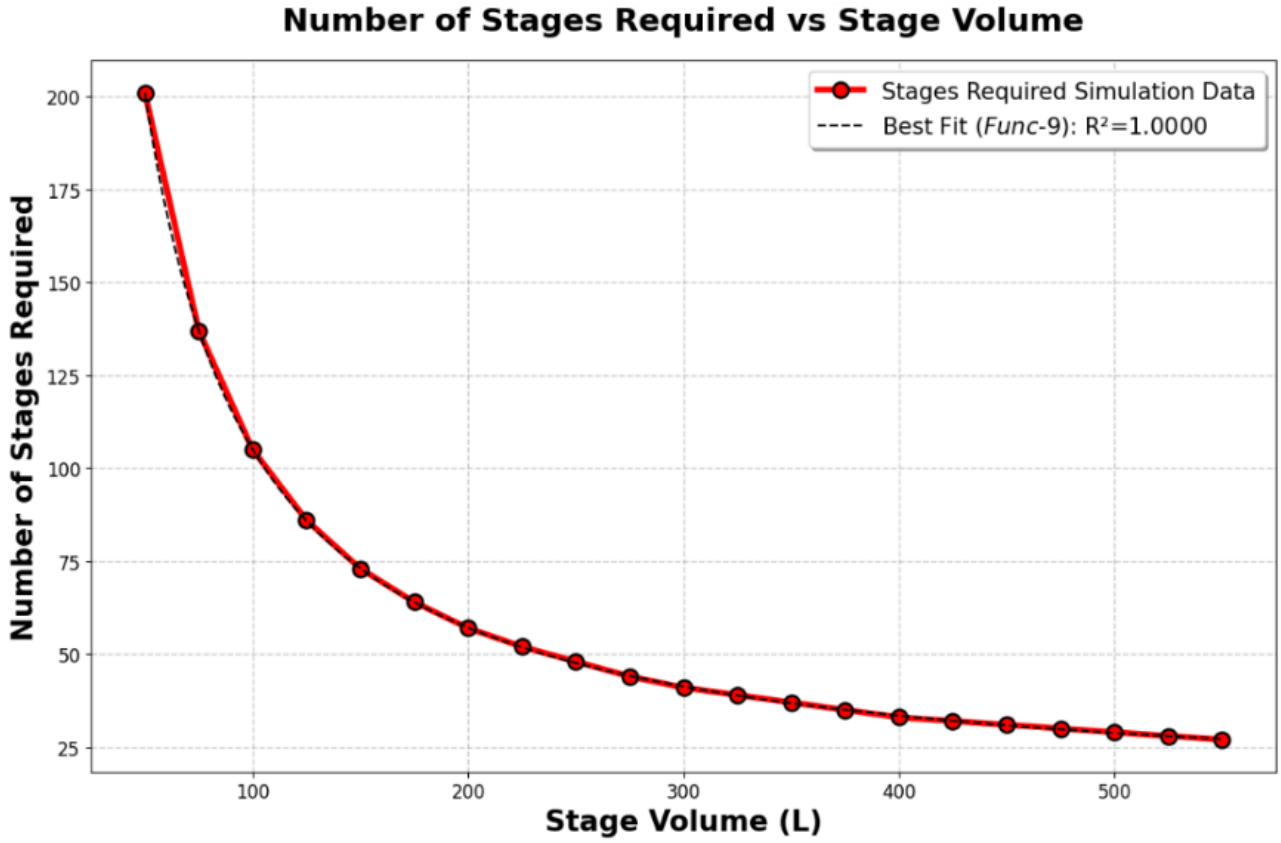


Figure 15: *Number of Stages Required with Varying Stage Volume*

Figure 15 reveals a sharp inverse relationship between stage volume and the number of required stages, with the curve exhibiting decaying characteristics. At smaller stage volumes (50 L – 150 L), the system requires dramatically more stages (200 stages down to approximately 70 stages) because insufficient residence time per stage limits the extent of the kinetically controlled CO_2 hydration reaction and mass transfer processes. As stage volume increases beyond 200 L, the curve flattens significantly, indicating diminishing returns where additional volume per stage provides progressively smaller reductions in stage count. This behaviour reflects the fundamental trade-off between providing adequate residence time for slow kinetic processes versus the economic penalty of excessive equipment size.

The accurately fitted empirical function, *Func-9*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.3.2. Reactive Stripping Column Size

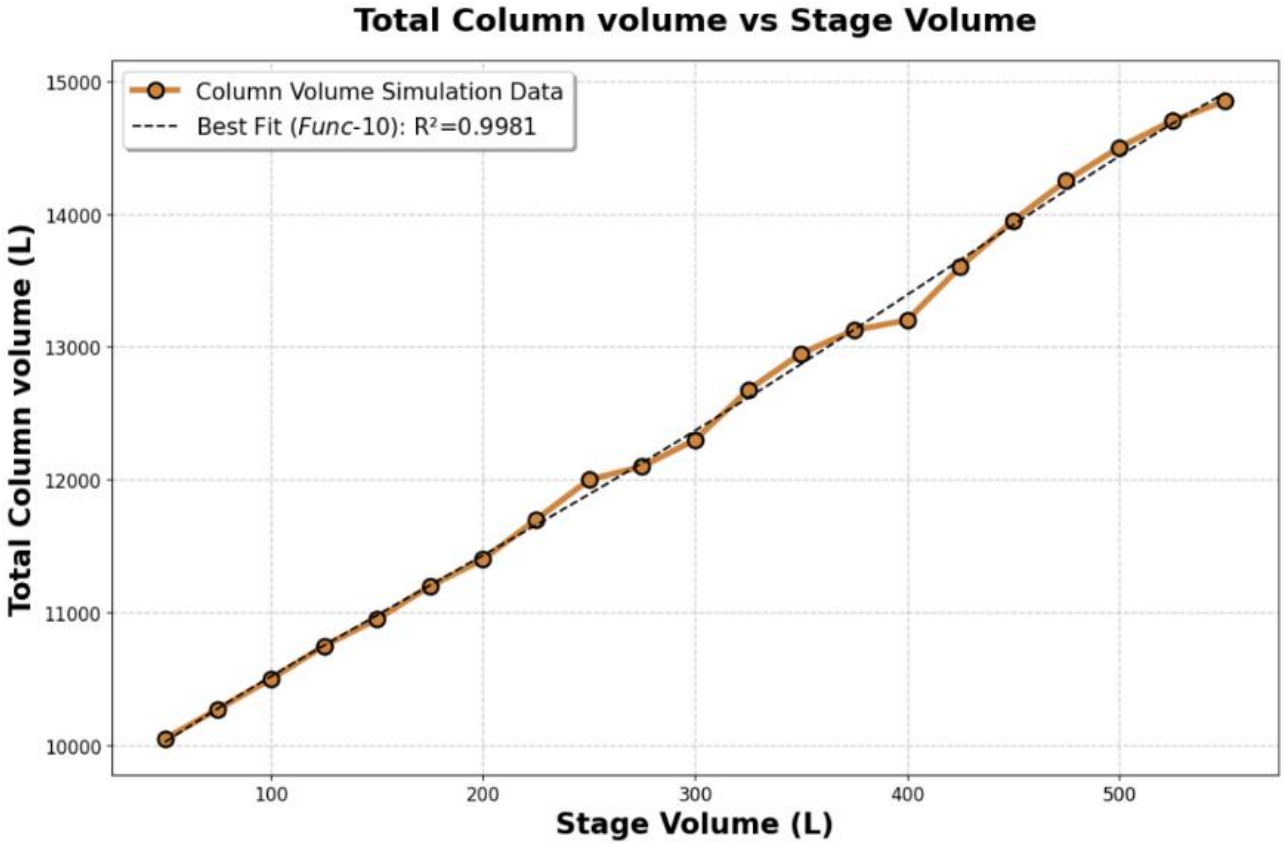


Figure 16: *Reactive Stripping Column Size Against Varying Stage Volume*

Figure 16 shows that total column volume increases monotonically with stage volume, displaying a consistently upward trend from approximately 10000 L at 50 L stage volume to nearly 15000 L at 550 L stage volume. The data clearly indicates that smaller stage volumes consistently result in lower total column volumes throughout the entire range examined.

However, while the plot suggests that the smallest feasible stage volume would be optimal from a total volume perspective, the actual optimal design would need to balance kinetic limitations with equipment costs. Very small stage volumes may not provide sufficient residence time for the slow hydration kinetics to reach completion, potentially compromising separation performance despite the lower total volume. Additionally, smaller stages typically involve higher equipment costs per unit volume due to increased surface area-to-volume ratios, more complex piping networks, and greater number of individual vessels required. Therefore, the true optimum likely exists at a stage volume larger than what this total volume analysis alone would suggest, where adequate kinetic performance is maintained while minimizing the combined impact of capital and operating costs.

The accurately fitted empirical function, *Func-10*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.4. Varying Outlet Gas Flowrate

Table 8 shows the input and system variables that were used to obtain the following results from the developed Python program (attached in Appendix *A8. Python Program for Outlet Gas Flowrate Sensitivity Analysis*) for the reactive stripping column model during an outlet gas flowrate sensitivity analysis:

Table 8: *Input and System Variables to Outlet Gas Flowrate Sensitivity Analysis*

Variable	Value	Units
$T_{operating}$	25	°C
$P_{operating}$	1	atm
V_{stage}	180	L
L	1	L/s
G_1	***0.825 - 0.9***	mol/s
ε	5	%
d_e	5	mm
$[NaHS]_0$	0.8	mol/L
x_{H_2S}	99.99	%

5.4.1. Number of Stages Required

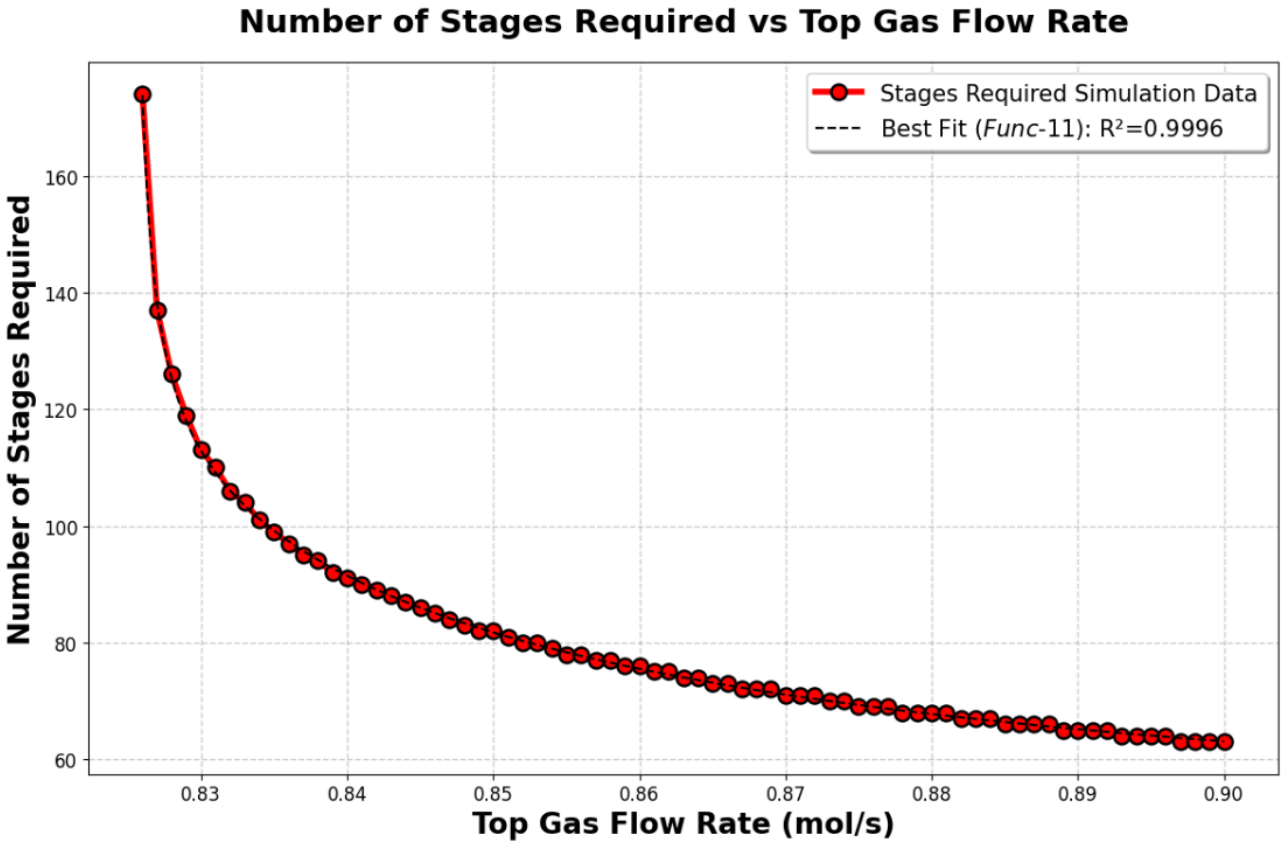


Figure 17: *Number of Stages Required with Varying Outlet Gas Flowrate*

Figure 17 demonstrates a strong inverse relationship between top gas flow rate and the number of stages required for achieving 99.99 % H_2S recovery. At lower gas flow rates (around 0.825 mol/s), approximately 170 stages are needed, while increasing the flow rate to 0.9 mol/s reduces the requirement to about 63 stages. This dramatic reduction occurs because higher gas flow rates provide greater driving force for mass transfer, enhancing the stripping efficiency per stage. The relationship follows a characteristic decay pattern, indicating diminishing returns as gas flow increases beyond optimal levels.

The accurately fitted empirical function, *Func-11*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.4.2. Percent Excess CO_2 Required

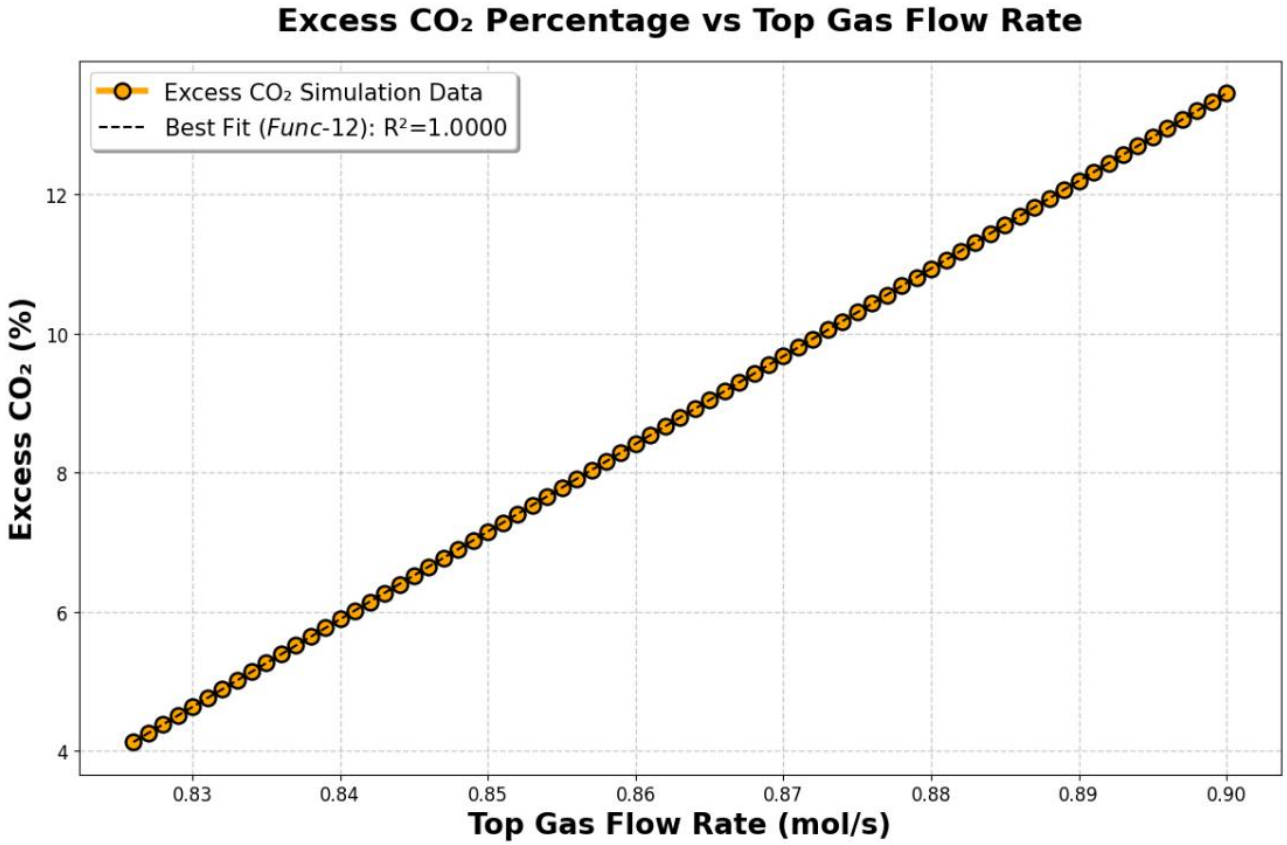


Figure 18: *% Excess CO_2 Requires with Varying Outlet Gas Flowrate*

Figure 18 illustrates the direct linear relationship between top gas flow rate and excess CO_2 percentage. As the gas flow rate increases from 0.825 mol/s to 0.9 mol/s, the excess CO_2 increases linearly from approximately 4 % to 13 %. This trend reflects the process design constraint where higher gas flows inherently provide more CO_2 than the stoichiometric minimum required for the acid-base reactions. The linear relationship suggests that the excess CO_2 is primarily determined by the gas flow rate rather than kinetic limitations, indicating that the process operates in a regime where mass transfer and thermodynamic driving forces dominate over reaction kinetics for the range of conditions studied.

The accurately fitted empirical function, *Func-12*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.4.3. Number of Stages Required for Levels of Excess CO_2

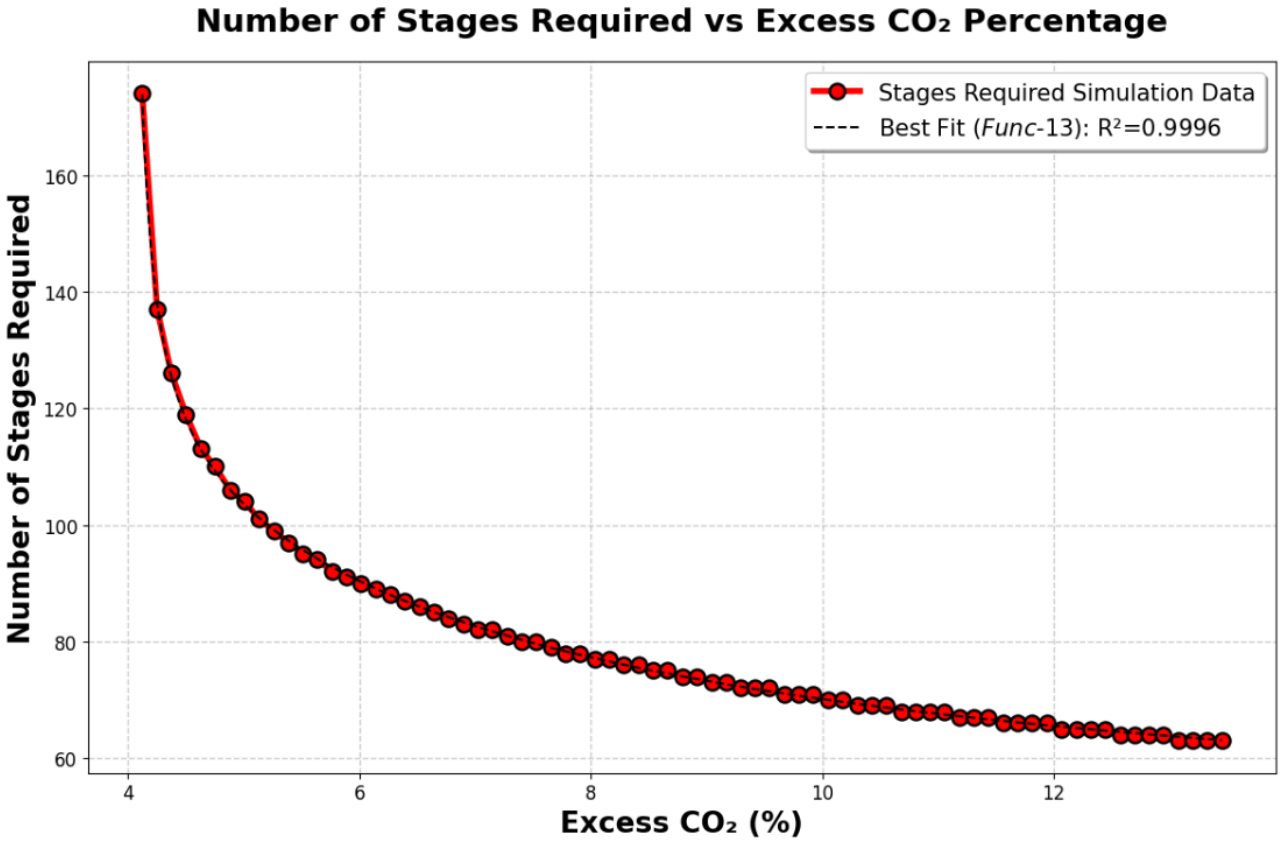


Figure 19: *Number of Stages Required with Varying % Excess CO_2 Required*

Figure 19 shows the relationship between excess CO_2 percentage and stage requirements. The excess CO_2 is calculated relative to the minimum CO_2 needed for complete conversion of sulphide species. Higher excess CO_2 percentages (ranging from about 4 % to 13 %) correspond to lower stage requirements, following the same decay pattern. This relationship exists because excess CO_2 shifts the chemical equilibrium favourably, promoting more complete conversion of H_2S and its ionic forms (HS^- and S^{2-}) to the strippable H_2S gas phase, while also providing additional driving force for mass transfer.

The accurately fitted empirical function, *Func-13*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.5. Varying $NaHS$ Feed Concentration

Table 9 shows the input and system variables that were used to obtain the following results from the developed Python program (attached in Appendix *A9. Python Program for $NaHS$ Feed Concentration Sensitivity Analysis*) for the reactive stripping column model during a $NaHS$ feed concentration sensitivity analysis:

Table 9: *Input and System Variables to $NaHS$ Feed Concentration Sensitivity Analysis*

Variable	Value	Units
$T_{operating}$	25	°C
$P_{operating}$	1	atm
V_{stage}	180	L
L	1	L/s
G_1	0.9	mol/s
ε	5	%
d_e	5	mm
$[NaHS]_0$	***0.5 - 0.87***	mol/L
x_{H_2S}	99.99	%

5.5.1. Number of Required Stages

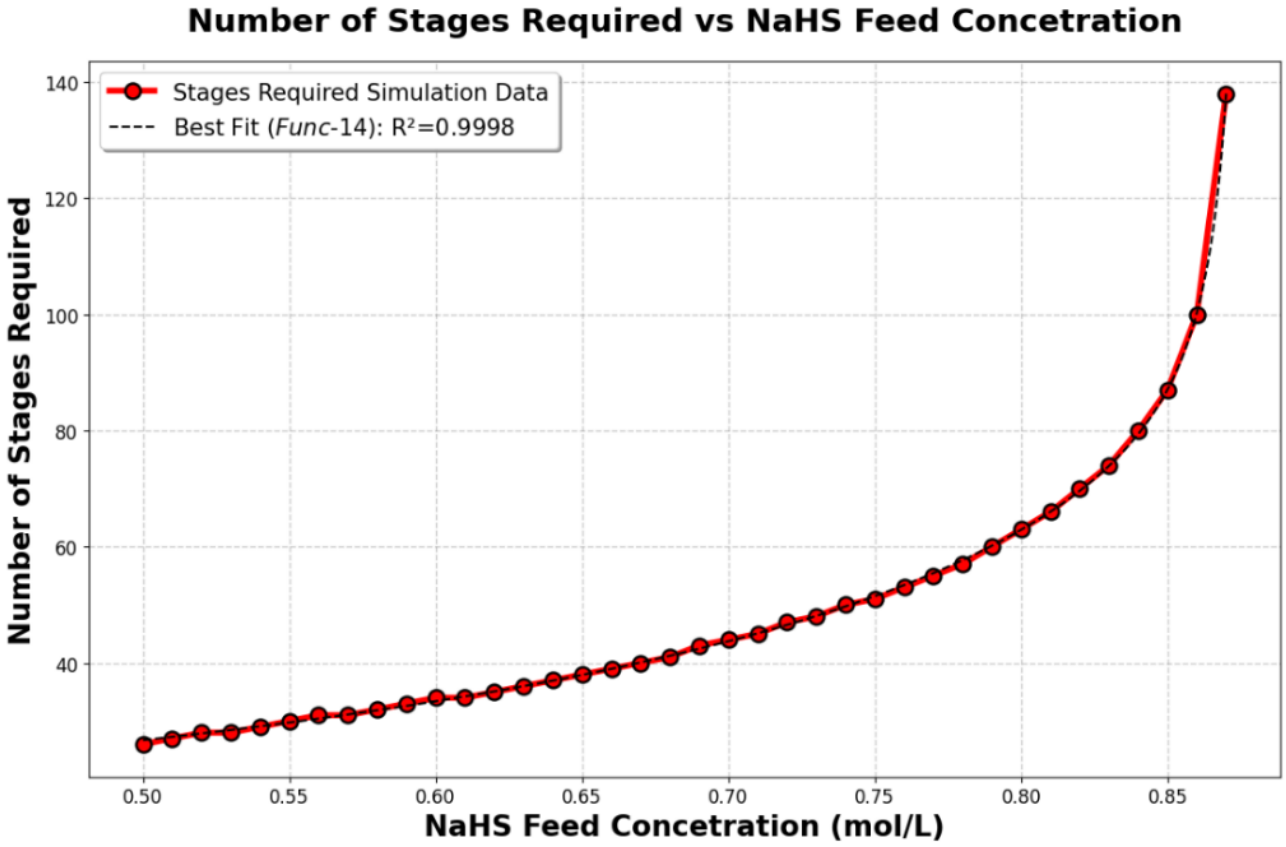


Figure 20: *Number of Stages Required with Varying $NaHS$ Feed Concentration*

Figure 20 showing the number of stages required versus $NaHS$ feed concentration, there is a clear proportional relationship where higher $NaHS$ concentrations dramatically increase the column complexity. At low concentrations around 0.5 mol/L, approximately 28 stages are sufficient, but this requirement grows to over 135 stages at 0.87 mol/L. This behaviour occurs because higher $NaHS$ concentrations create more alkaline conditions that favour the formation of HS^- and S^{2-} species over the H_2S form needed for stripping. The increased alkalinity shifts the sulphur speciation equilibrium away from the easily stripped H_2S, requiring significantly more contact stages to achieve the target 99.99 % H_2S recovery. Additionally, the kinetic limitations of the CO_2 hydration reaction becomes more pronounced at higher concentrations, as the system struggles to maintain the carbonic acid equilibrium needed to provide sufficient H^+ ions for converting HS^- back to strippable H_2S.

The accurately fitted empirical function, *Func-14*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.5.2. Bottom Stage pH Value

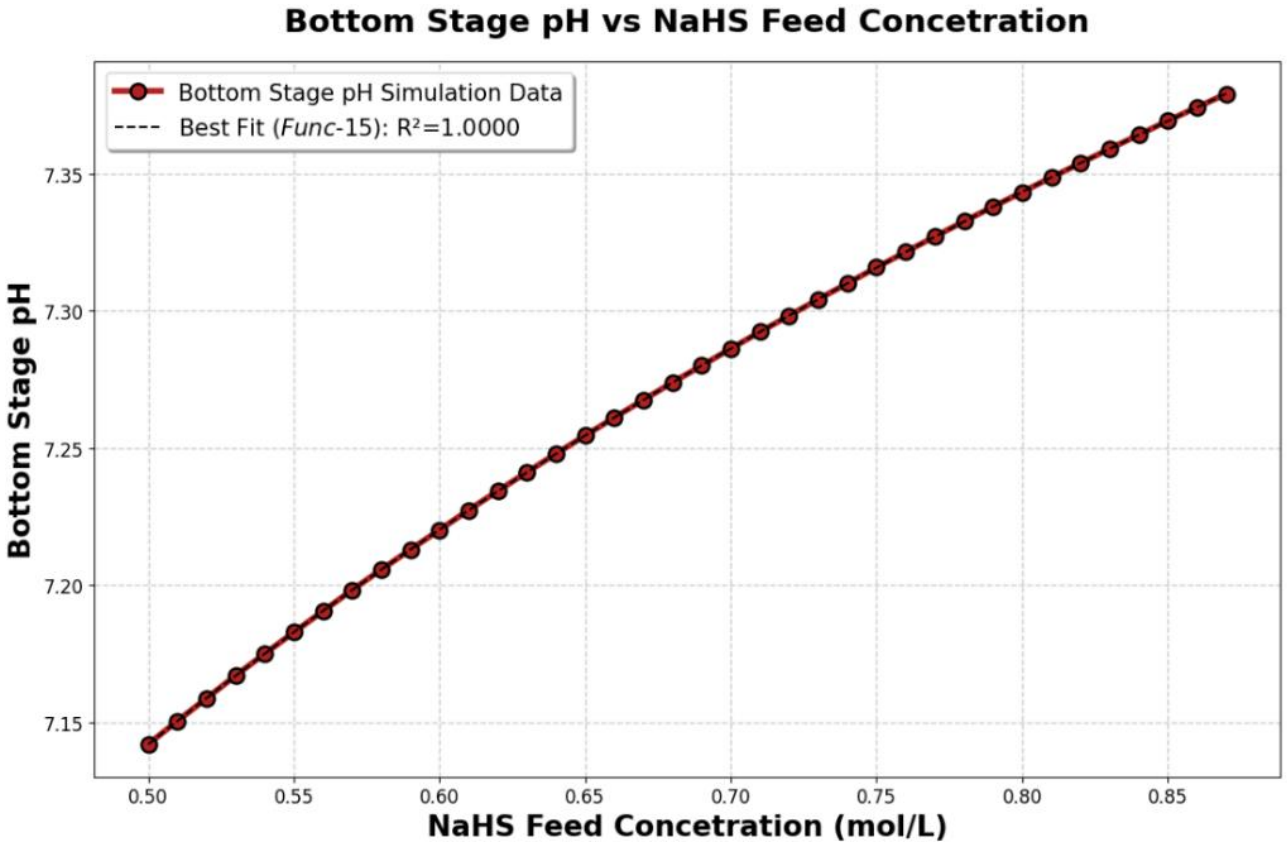


Figure 21: *Bottom Stage pH Against Varying $NaHS$ Feed Concentration*

Figure 21 demonstrates how the bottom stage pH increases linearly with $NaHS$ feed concentration, rising from approximately 7.14 at 0.5 mol/L to 7.38 at 0.87 mol/L. This pH increase directly explains the growth in stage requirements shown in the first plot. As $NaHS$ concentration increases, more HS^- ions are introduced into the system, creating a stronger buffering effect that maintains higher pH values throughout the column. The higher pH conditions thermodynamically favour the retention of sulphur in ionic forms (HS^- and S^{2-}) rather than the molecular H_2S form required for gas-phase stripping. This pH-dependent speciation shift, combined with the kinetic constraints of CO_2 hydration that limit the system's ability to generate sufficient H^+ ions for neutralization, creates increasingly challenging conditions for efficient H_2S stripping as the feed concentration rises.

The accurately fitted empirical function, *Func-15*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.5.3. Bottom Gas Flowrate

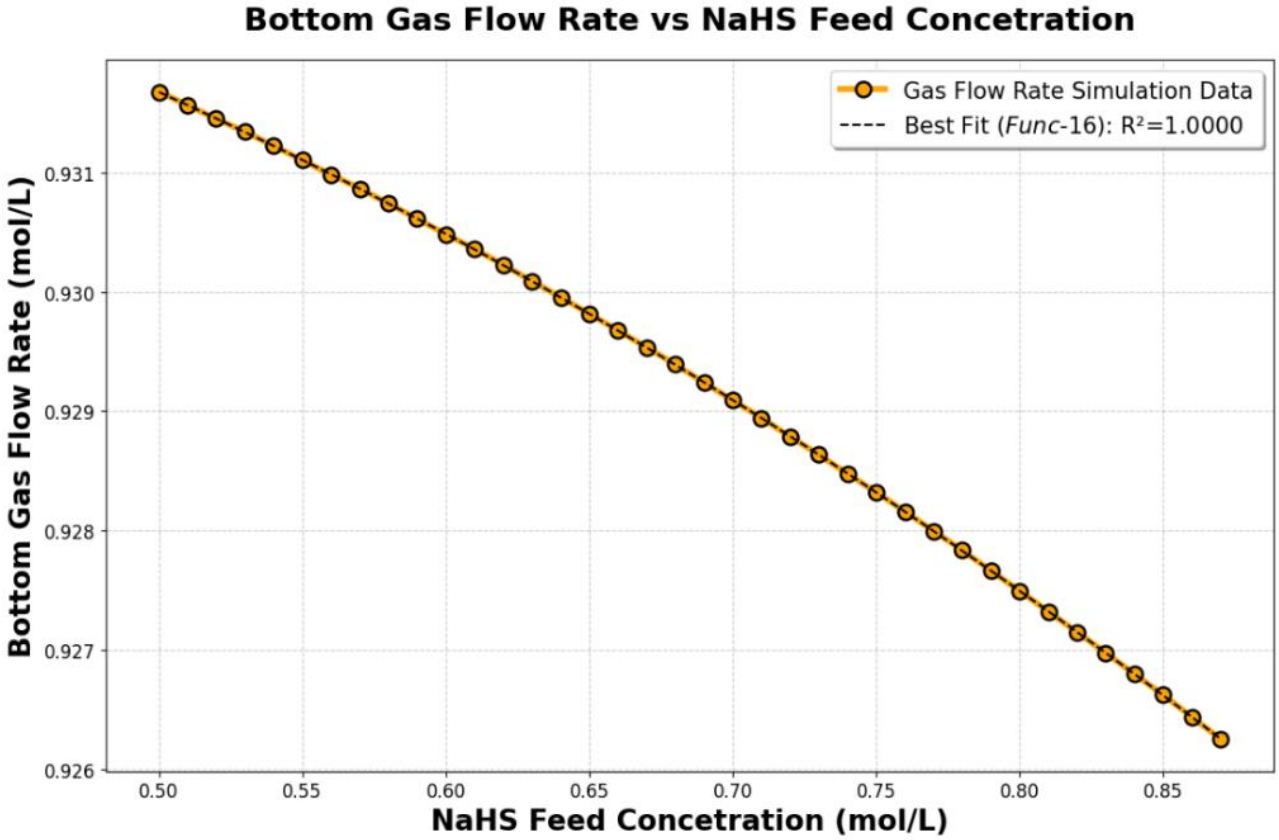


Figure 22: *Bottom Gas Flowrate with Varying $NaHS$ Feed Concentration*

In *Figure 22* the bottom gas flow rate shows a clear decreasing trend as $NaHS$ feed concentration increases, dropping from approximately 0.9315 mol/s at 0.5 mol/L $NaHS$ to about 0.9262 mol/s at 0.86 mol/L $NaHS$. This behaviour reflects the complex interplay between mass transfer driving forces and the kinetic limitations identified. At lower $NaHS$ concentrations, there's less H_2S available for stripping, but the mass transfer coefficients and interfacial area effects create conditions that require higher gas flow rates to achieve the target 99.99 % H_2S recovery. As $NaHS$ concentration increases, the higher dissolved sulphide content provides a stronger driving force for H_2S transfer to the gas phase, allowing the same recovery target to be met with slightly lower gas flow rates. This trend also reflects the slow CO_2 hydration kinetics affecting the overall system equilibrium – at higher $NaHS$ concentrations, the solution chemistry shifts in ways that make the stripping process marginally more efficient from a gas flow perspective.

The accurately fitted empirical function, *Func-16*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.5.4. Minimum CO_2 Required

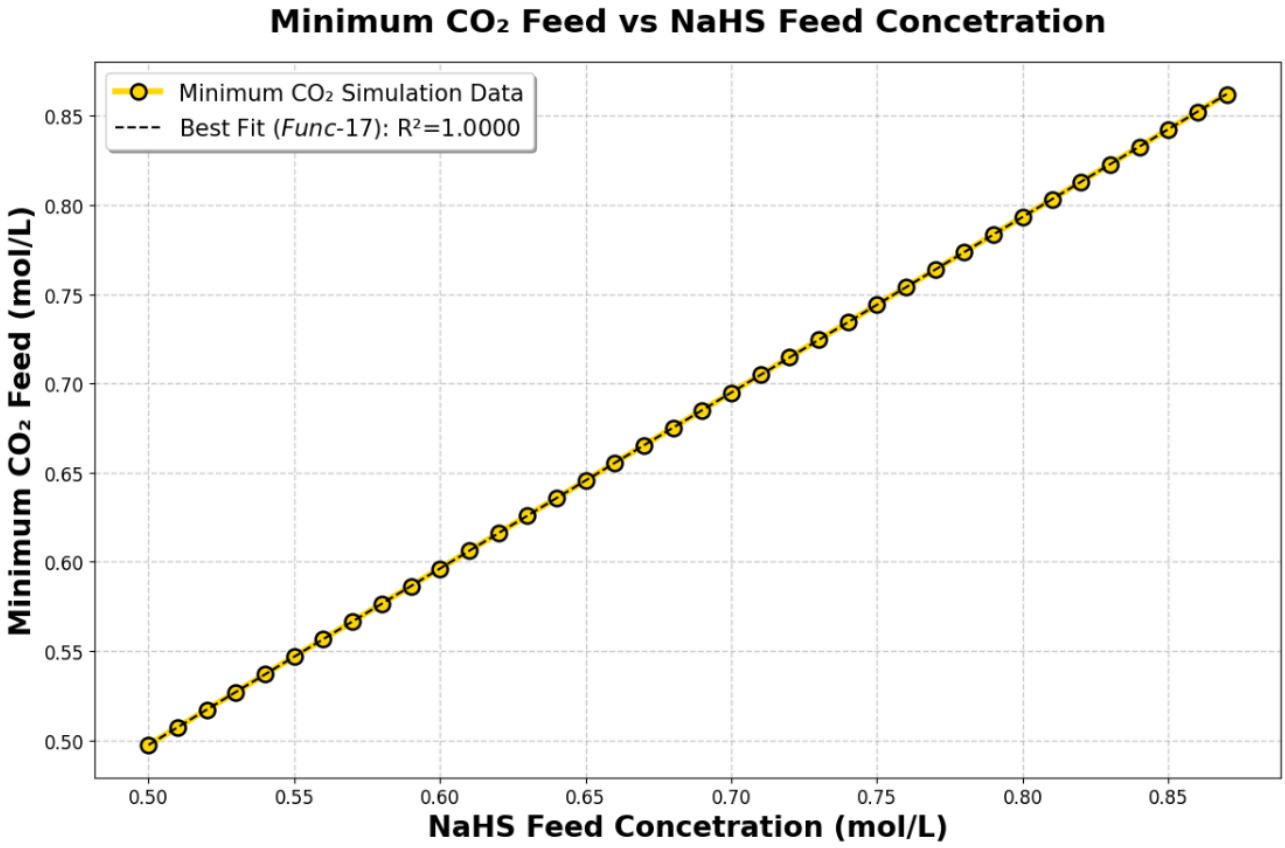


Figure 23: *Minimum CO_2 Required with Varying $NaHS$ Feed Concentration*

In *Figure 23* the minimum CO_2 feed shows a strong linear increase with $NaHS$ concentration, rising from approximately 0.5 mol/L at low $NaHS$ feeds to about 0.86 mol/L at the highest $NaHS$ concentration tested. This relationship is fundamentally driven by the stoichiometric and equilibrium constraints of the system. Higher $NaHS$ concentrations introduce more sulphide species that must be converted to H_2S for stripping. The CO_2 serves as the acidifying agent that shifts the sulphide equilibrium toward the molecular H_2S form, which has much higher volatility and can be effectively stripped. The linear relationship suggests that each mole of additional $NaHS$ requires approximately one mole of additional CO_2 to maintain the proper pH conditions for efficient H_2S formation and stripping. This trend highlights why previous equilibrium-based models were insufficient – the slow CO_2 hydration kinetics create bottlenecks that affect how quickly the system can respond to changes in feed composition, making the CO_2 requirement higher than simple equilibrium calculations would predict.

The accurately fitted empirical function, *Func-17*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.6. Varying H_2S Recovery

Table 10 shows the input and system variables that were used to obtain the following results from the developed Python program (attached in Appendix *A10. Python Program for H_2S Recovery Sensitivity Analysis*) for the reactive stripping column model during a H_2S recovery sensitivity analysis:

Table 10: *Input and System Variables to H_2S Recovery Sensitivity Analysis*

Variable	Value	Units
$T_{operating}$	25	°C
$P_{operating}$	1	atm
V_{stage}	180	L
L	1	L/s
G_1	0.9	mol/s
ε	5	%
d_e	5	mm
$[NaHS]_0$	0.8	mol/L
x_{H_2S}	***97 - 99.999***	%

5.6.1. Number of Stages Required

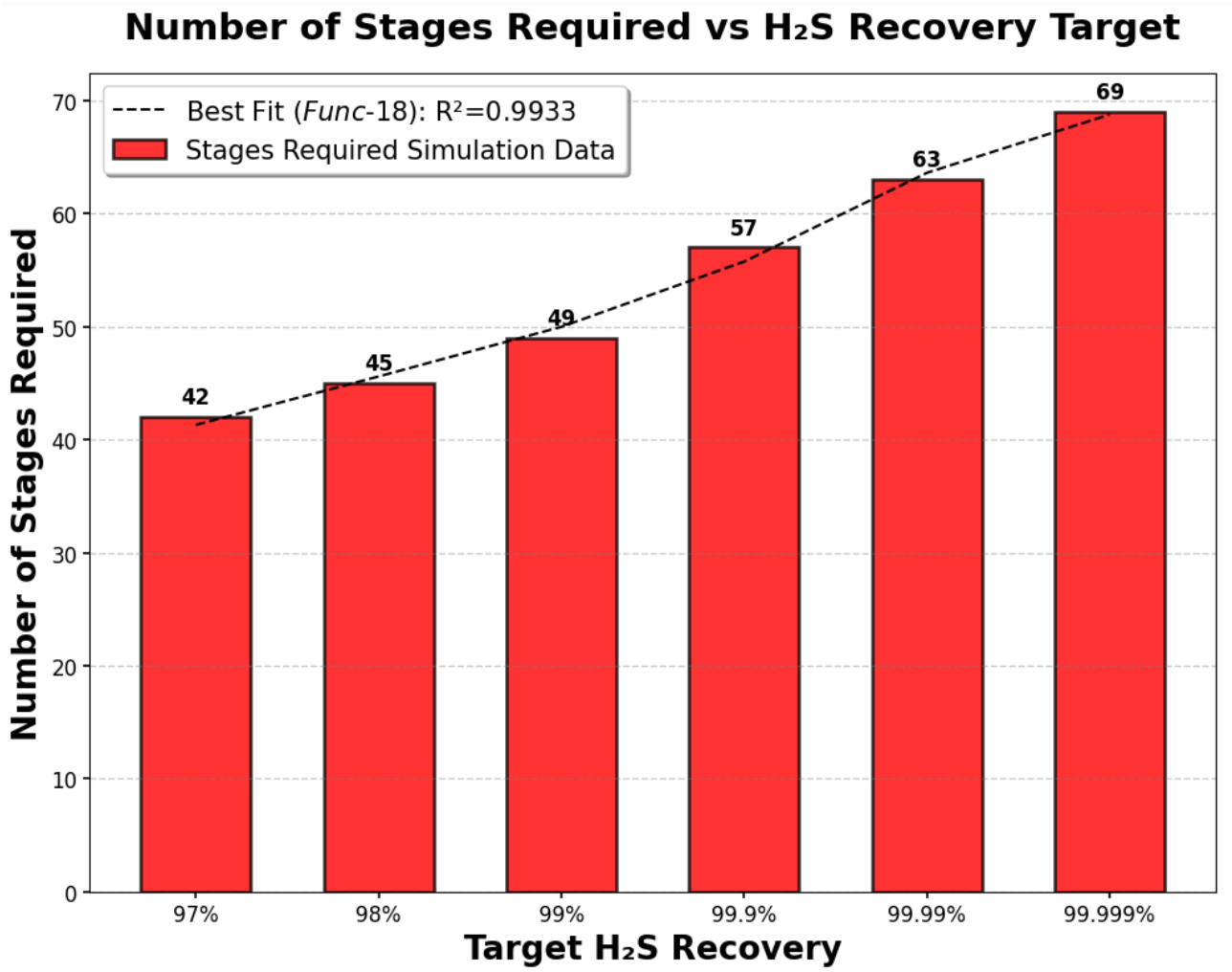


Figure 24: *Number of Satges Required with Varying H_2S Recovery*

The increase in required stages *in Figure 24* (from 42 stages at 97 % recovery to 69 stages at 99.999 % recovery) demonstrates the diminishing returns effect inherent in mass transfer operations. Each additional stage provides progressively smaller improvements in H_2S removal as the driving force decreases. The steep rise at higher recoveries is exacerbated by the kinetic limitations of CO_2 hydration, which reduce mass transfer efficiency and require more contact time (additional stages) to achieve the desired separation. This trend illustrates why industrial H_2S stripping processes typically operate at moderate recovery targets rather than pursuing maximum theoretical efficiency.

The accurately fitted empirical function, *Func-18*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.6.2. Bottom Stage pH Value

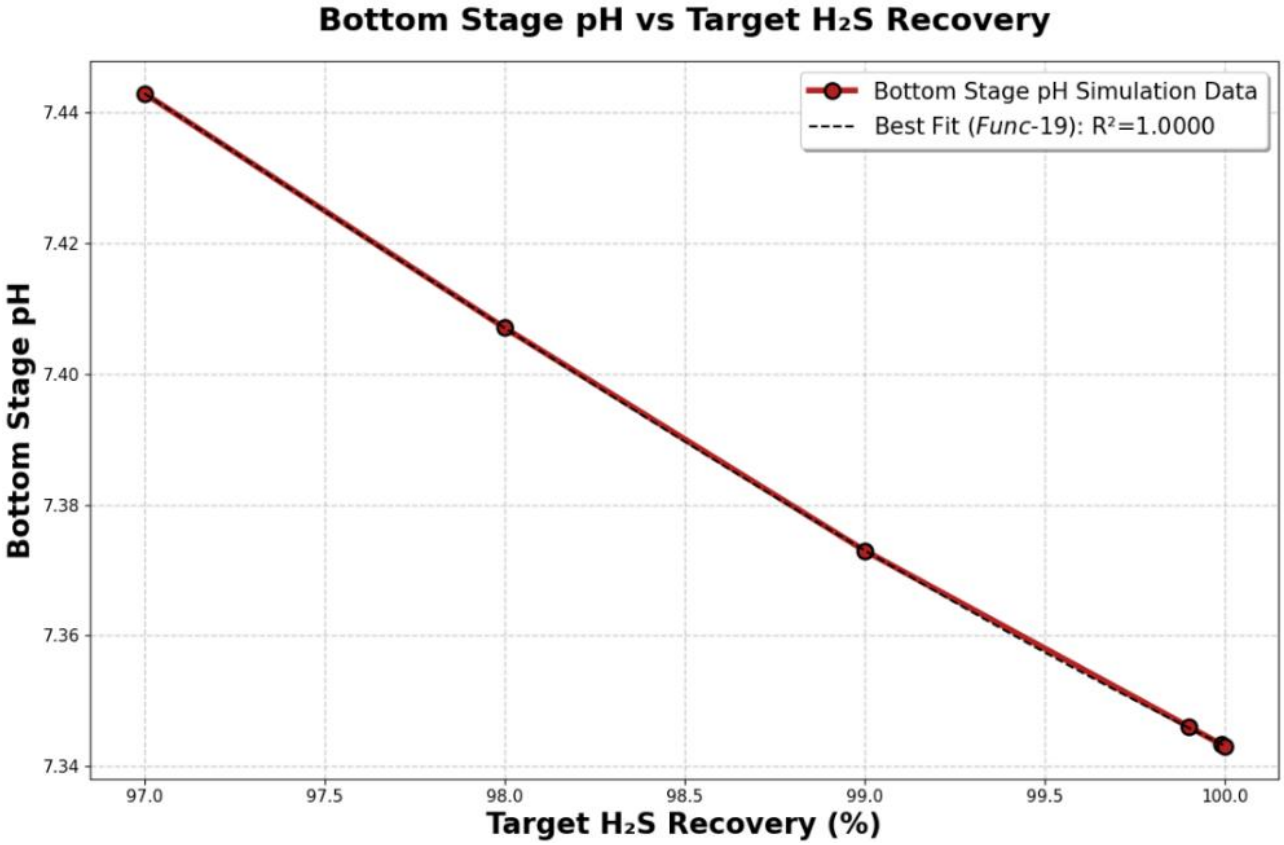


Figure 25: *Bottom Satge pH Against Varying H_2S Recovery*

In *Figure 25* the decreasing pH trend from 7.44 to 7.34 with increasing recovery targets reveals the chemical equilibrium shifts occurring within the stripping column. As more H_2S is stripped from solution, the remaining bisulfide (HS^-) and sulphide (S^{2-}) species undergo protonation reactions to maintain equilibrium, consuming hydroxide ions and lowering the solution pH. This pH depression becomes more pronounced at higher recoveries because the system must maintain increasingly lower dissolved sulphur concentrations, requiring greater acidification to shift the H_2S/HS^- equilibrium toward the more volatile molecular H_2S form.

The accurately fitted empirical function, *Func-19*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.6.3. Actual and Target H_2S Recovery

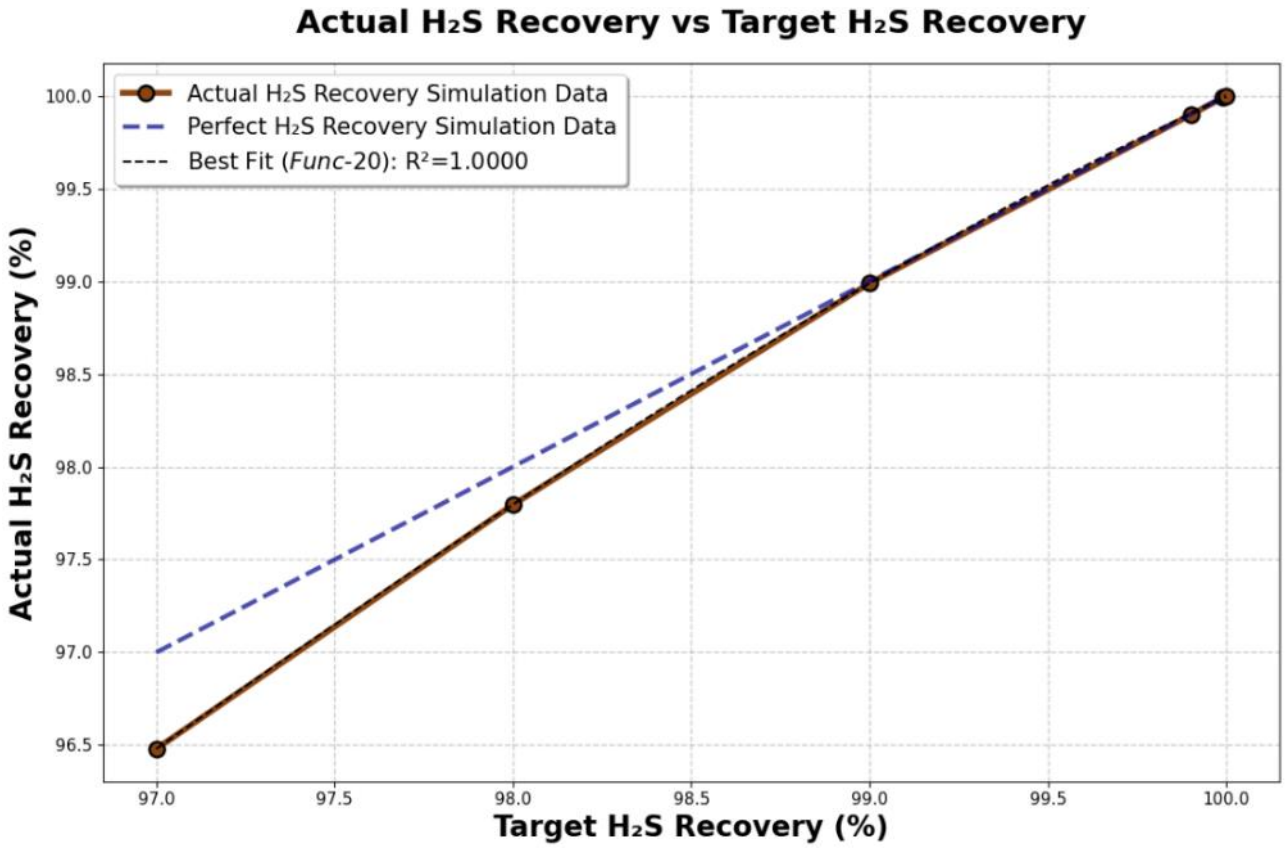


Figure 26: *Actual H_2S Recovery Against Target H_2S Recovery*

In the figure above, the target H_2S recovery represents the desired separation efficiency that is set as an input parameter (such as 97 %, 98 %, 99 %, etc.), which the Python program uses to calculate the theoretical H_2S concentration needed in the top gas stream and determine other important outputs. In contrast, the actual H_2S recovery is what the simulation realistically achieves after solving the complete system of chemical equilibrium equations, mass balances, and mass transfer relationships for each individual stage in the stripping column.

The close alignment in *Figure 26* between actual and target recovery values, particularly at higher recovery targets, validates the robustness of the stage-wise calculation methodology. The slight underprediction at lower recoveries (96.5 % actual vs 97 % target) suggests that the discrete stage approximation becomes more accurate as the number of stages increases and the system approaches true equilibrium behaviour. This convergence indicates that the kinetic limitations and mass transfer resistances are properly captured in the model, allowing for reliable prediction of column performance across the entire recovery range studied.

The accurately fitted empirical function, *Func-20*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.6.4. Minimum CO_2 Required

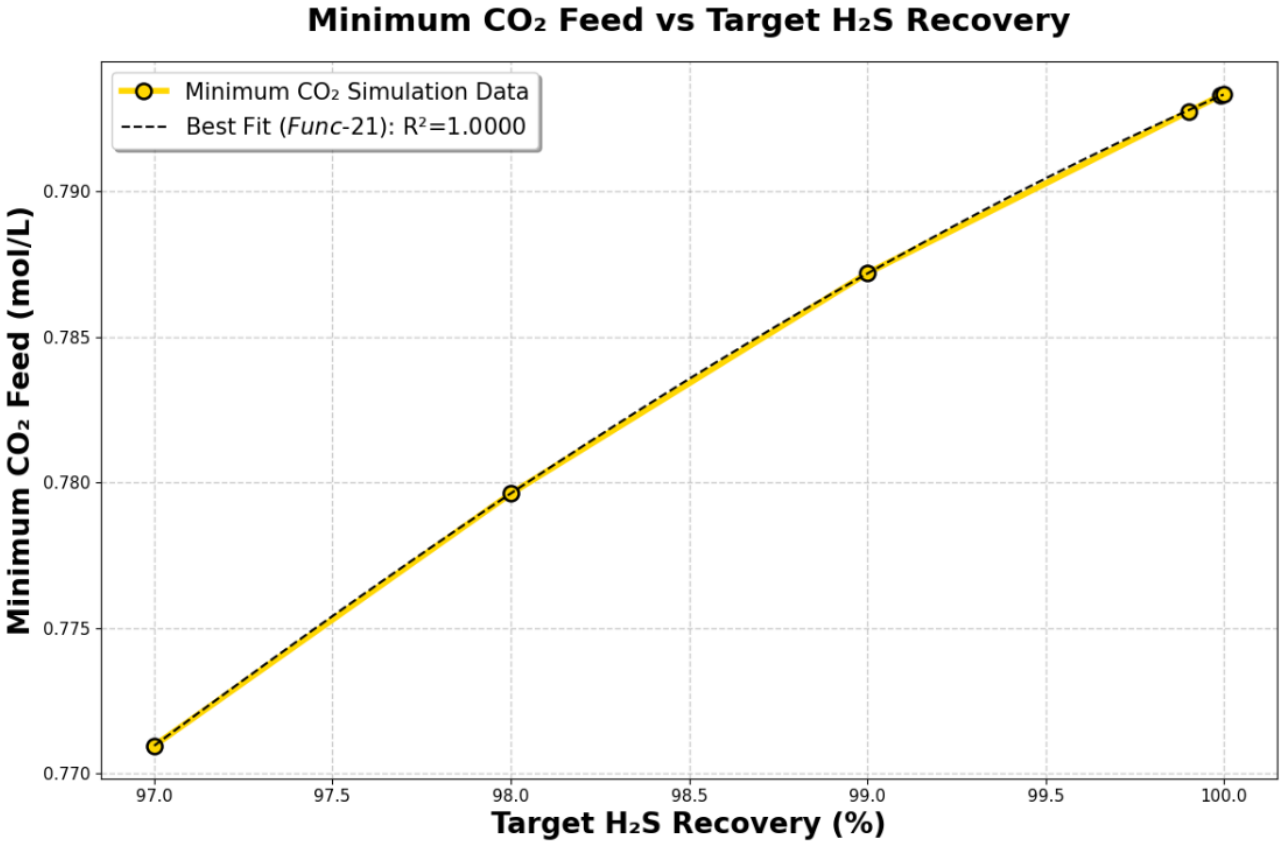


Figure 27: *Minimum CO_2 Feed Required with Vaying H_2S Recovery*

In *Figure 27* the linear increase in minimum CO_2 feed requirement (from ~0.771 mol/L to 0.793 mol/L) with higher H_2S recovery targets reflect the carbonic acid equilibrium system's role in pH buffering. As more H_2S is stripped from solution, the remaining bisulfide ions (HS^-) create a more alkaline environment. To maintain the necessary pH conditions for efficient H_2S volatilization and overcome the kinetic limitations of the CO_2 hydration reaction, additional CO_2 must be fed to form carbonic acid species ($H_2CO_3$and HCO_3^-) that provide sufficient acidic buffering capacity. This relationship demonstrates how the slow kinetics of CO_2 hydration becomes increasingly limiting at higher recovery targets.

The accurately fitted empirical function, *Func-21*, was obtained using the *curve_fit* function from the submodule *scipy.optimize* (see Appendix *A1. Empirical Equations for the Relevant Sensitivity Analyses*).

5.6.5. Actual and Theoretical H_2S Partial Pressure

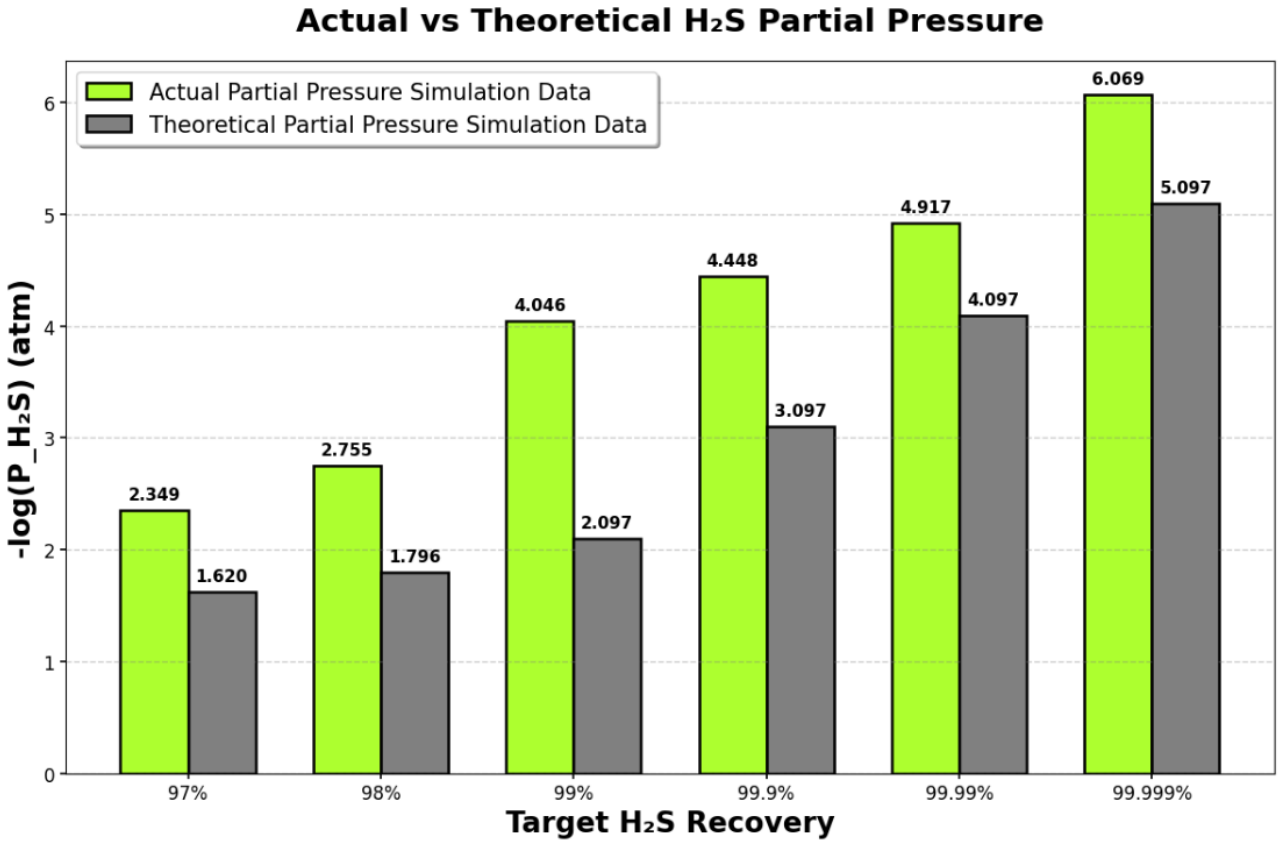


Figure 28: *H_2S Partial Pressure with Varying H_2S Recovery*

In the above figure, the difference between actual and theoretical H_2S partial pressure represents two different modelling approaches. The theoretical value uses a simple mass balance assumption – if a certain recovery percentage is required, the amount of sulphur remaining in the liquid phase can be calculated, assuming perfect stripping efficiency without considering complex chemical phenomena. The actual H_2S partial pressure comes from the rigorous stage-by-stage simulation that accounts for chemical equilibria between sulphur species, phase equilibria, mass transfer limitations, charge balance, and pH effects on speciation. This represents what happens in the real system with all its complexities.

In *Figure 28* the comparison reveals that actual H_2S partial pressures consistently exceed theoretical predictions, with the deviation increasing dramatically at higher recovery targets (from 2.349 atm vs 1.620 atm at 97 % to 6.069 atm vs 5.097 atm at 99.999 %). This discrepancy highlights the impact of kinetic limitations that aren't captured in equilibrium-based models. The slow CO_2 hydration kinetics prevent the system from reaching true thermodynamic equilibrium, resulting in incomplete pH control and higher-than-expected H_2S volatility. The nature of this deviation at extreme recovery targets indicates that kinetic effects become dominant over equilibrium considerations.

6. Discussion

The developed computational model provides a detailed and accurate representation of a reactive H_2S stripping column that accounts for the slow hydration kinetics of CO_2 – something traditional equilibrium-based models fail to capture. Across all simulations and sensitivity analyses, the model demonstrates that neglecting CO_2 hydration kinetics introduces significant errors in predicting key performance metrics such as H_2S recovery, pH profiles, and gas-phase compositions. For instance, targeting very high H_2S recovery levels (e.g., 99.99 %) results in increases in the number of required stages, highlighting the diminishing returns associated with pushing mass transfer operations toward thermodynamic limits. This is further exacerbated by the kinetic bottleneck introduced by the CO_2 hydration reaction, which slows down acid formation and limits the rate of H_2S conversion to the gas phase.

Temperature was shown to have a complex influence on the process. While higher temperatures generally improve mass transfer due to reduced viscosity and increased diffusivity, they also reduce CO_2 solubility and shift chemical equilibria in unfavourable directions. Similarly, feed concentration significantly affects the process. Higher NaHS concentrations increase the system's alkalinity, which in turn raises the pH and favours the retention of sulphur species in ionic form rather than the volatile H_2S form needed for effective stripping. This results in a steep rise in the number of stages required and underscores the importance of pH management in such systems.

Optimizing operating pressure and outlet gas flowrate presents opportunities for improving efficiency and reducing equipment size. The simulations identified a practical range of operating pressures (2 atm – 3 atm) that minimize the total number of stages in the column and thus minimizing capital cost. Likewise, increasing CO_2 flowrate improves stripping performance up to a point, after which the returns diminish, and excess gas becomes wasteful.

7. Recommendations

- Conduct experimental validation to confirm model accuracy under real-world conditions and compare with industrial plant data.
- Investigate non-isothermal effects as temperature changes along the column could influence reaction rates and mass transfer behaviour.
- Use the developed Python program and empirical functions as a basis to develop an integrated optimization and cost estimation tool that automatically determines ideal operating conditions while minimizing total costs.
- Adapt the model for scale-up to assist in pilot and full-scale industrial system design with appropriate safety factors
- Develop simplified correlations and design guidelines for industrial column sizing based on feed conditions and recovery targets
- Research catalyst addition or alternative process configurations to accelerate CO_2 hydration kinetics and reduce the kinetic bottleneck
- Investigate dynamic behaviour, control strategies, and the effects of impurities on model performance for robust industrial operation

8. Conclusions

- A rigorous, stage-wise *Python 3.13.3* model for simulating a reactive stripping column that specifically incorporates the kinetic limitations of hydration was successfully developed.
- A more realistic and predictive framework than traditional equilibrium-based approaches by integrating chemical equilibria, reaction kinetics, and gas-liquid mass transfer was created.
- The model's capability to simulate complex multicomponent behaviour across a wide range of operating conditions through extensive sensitivity analyses was validated.
- The critical impact of hydration kinetics on process performance was identified.
- A strong nonlinear response of the system to variables such as temperature and feed concentration was demonstrated.
- A practical design and operating windows that balance efficiency and capital cost was established.
- A simulation framework that captures true physical behaviour of the system and offers clear guidance for optimizing column design and operation was developed.
- A key gap in the modelling of reactive gas-liquid systems was addressed.
- A strong foundation for process optimization and scale-up was established.
- A significant step toward more accurate and efficient design of industrial processes for sodium carbonate production via stripping was represented.

9. References

- ChemicalBook, 2022. *Uses of Sodium Carbonate*. [Online] Available at: https://www.chemicalbook.com/article/uses-of-sodium-carbonate.htm [Accessed 10 April 2025].
- Co., C. S., n.d.. *Soda Ash (Sodium Carbonate) Applications in the Paper Industry*. [Online] Available at: https://causticsodaco.com/soda-ash-sodium-carbonate-applications-in-the-paper-industry/ [Accessed 10 April 2025].
- Dreybrodt, W. L. J. Z. L. S. U. a. B. D., 1996. The kinetics of the reaction $CO_2 + H_2O \rightarrow H^+ + HCO_3^-$ as one of the rate limiting steps for the dissolution of calcite in the system H_2O-CO_2-$CaCO_3$. *Geochimica et Cosmochimica Acta*, 60(18), p. 3375–3381.
- Guo, X. et al., 2018. The mechanism of inclusion removal from molten steel by dissolved gas flotation. *Ironmaking & Steelmaking*, pp. 648-654.
- Millero, F. J., 2003. *The thermodynamics and kinetics of the hydrogen sulfide system in natural waters*, Miami: ScienceDirect.
- Millero, F. J. et al., 2002. *Dissociation constants for carbonic acid determined from field measurements*, Miami: ScienceDirect.
- Mokonyane, G., 2024. Stripping Hydrogen Sulphide from an Aqueous Solution of Sodium Hydrosulphide, Pretoria: University of Pretoria
- Nock, W., Heaven, S. & Banks, C., 2015. *Mass transfer and gas–liquid interface properties of single CO2 bubbles rising in tap water*, United Kingdom: Elsevier.
- Olofsson, G. & Hepler, L. G., 1975. Thermodynamics of ionization of water over wide ranges of temperature and pressure. *J Solution Chem 4*, p. 127–143.
- Perry, R. G. D. a. M. J., 2018. *Perry's Chemical Engineering Handbook*. 9th ed. New York: McGraw-Hill Education.
- Schulz, K. et al., 2006. *Determination of the rate constants for the carbon dioxide to bicarbonate inter-conversion in pH-buffered seawater systems*, Bremerhaven: Elsevier.
- Soli, A. L. & Byrne, R. H., 2002. *CO2 system hydration and dehydration kinetics and the equilibrium CO2/H2CO3 ratio in aqueous NaCl solution*, Florida: Elsevier.

- Stephens, H. & Cobble, J. W., 1970. *Thermodynamic Properties of the Aqueous Sulfide and Bisulfide Ions and the Second Ionization Constant of Hydrogen Sulfide over Extended Temperatures*, Indiana: ACS Publications.
- Vargaftik, N., Volkov, B. & Voljak, L., 1983. *International Table of the Surface Tension of Water*, Moscow: s.n.

10. Appendices

A1. Empirical Equations for the Relevant Sensitivity Analyses

4th Degree Polynomial Empirical Functions

$$y = C_0x^4 + C_1x^3 + C_2x^2 + C_3x + C_4$$

Function	Variable x	Variable y	C_0	C_1	C_2	C_3	C_4
Func-2	$T_{operating}$ (°C)	pH_N	8.07×10^{-7}	-7.36×10^{-5}	2.55×10^{-3}	-3.23×10^{-2}	7.39×10^{0}
Func-3	$T_{operating}$ (°C)	G_N (mol/s)	-5.22×10^{-8}	5.24×10^{-6}	-1.75×10^{-4}	8.7×10^{-4}	9.53×10^{-1}
Func-4	$T_{operating}$ (°C)	$y_{CO_2,N}$	-6.63×10^{-9}	-4.55×10^{-8}	-2.12×10^{-5}	-3.47×10^{-4}	9.94×10^{-1}
Func-10	V_{stage} (L)	V_{total} (L)	-4.91×10^{-8}	5.85×10^{-5}	-2.14×10^{-2}	1.22×10^{1}	9.46×10^{3}
Func-12	G_1 (mol/s)	$excess\ CO_2$ (%)	-8.51×10^{2}	2.94×10^{3}	-3.8×10^{3}	2.32×10^{3}	-5.72×10^{2}
Func-15	$[NaHS]_0$ (mol/L)	pH_N	-4.18×10^{-1}	1.62×10^{0}	-2.61×10^{0}	2.46×10^{0}	6.39×10^{0}
Func-16	$[NaHS]_0$ (mol/L)	G_N (mol/s)	4.85×10^{-3}	-1.31×10^{-2}	2.82×10^{-3}	-6.33×10^{-3}	9.35×10^{-1}
Func-20	x_{H_2S} (%)	$x_{H_2S,actual}$ (%)	8.34×10^{-4}	-3.38×10^{-1}	5.12×10^{1}	-3.44×10^{3}	8.67×10^{4}

3rd Degree Polynomial Empirical Functions

$$y = C_0x^3 + C_1x^2 + C_2x + C_3$$

Function	Variable x	Variable y	C_0	C_1	C_2	C_3
Func-19	x_{H_2S} (%)	pH_N	3.75×10^{-4}	-1.09×10^{-1}	1.06×10^{1}	-3.33×10^{2}
Func-21	x_{H_2S} (%)	$min\ CO_2\ feed$ (mol/L)	-4.77×10^{-5}	1.35×10^{-2}	-1.26×10^{0}	3.95×10^{1}

Rational Empirical Functions

$$y = \frac{C_0 x^2 + C_1 x + C_2}{x^2 + C_3 x + C_4}$$

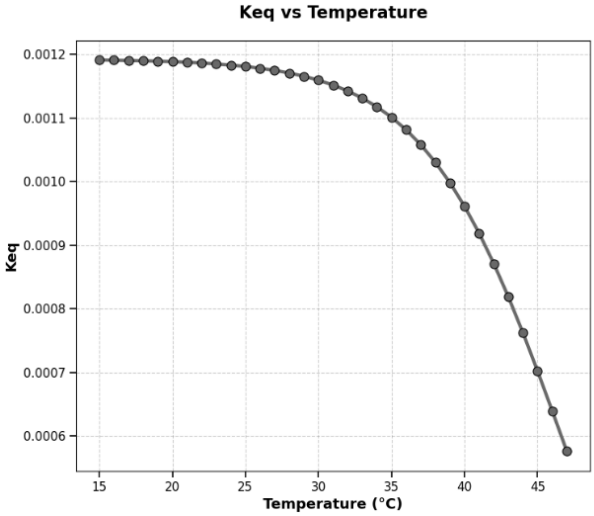
Function	Variable x	Variable y	C_0	C_1	C_2	C_3	C_4
Func-1	$T_{operating}$ (°C)	N	1.46×10^{7}	-1.89×10^{9}	6.93×10^{10}	-2.08×10^{7}	1.01×10^{9}
Func-5	$P_{operating}$ (atm)	N	2.53×10^{7}	1.44×10^{8}	2.31×10^{8}	8.08×10^{6}	-1.49×10^{6}
Func-7	$P_{operating}$ (atm)	G_N (mol/s)	3.69×10^{2}	9.6×10^{3}	-3.75×10^{2}	1.07×10^{4}	$-3.38 \times 10^{2}2$
Func-8	$P_{operating}$ (atm)	$y_{CO_2,N}$	1×10^{0}	2.3×10^{0}	-7.3×10^{-2}	2.33×10^{0}	2.3×10^{-4}
Func-9	V_{stage} (L)	N	1.07×10^{4}	5.26×10^{7}	5.82×10^{10}	6.05×10^{6}	4.37×10^{5}
Func-11	G_1 (mol/s)	N	4.38×10^{1}	-6.9×10^{1}	2.71×10^{1}	-1.62×10^{0}	6.56×10^{-1}
Func-13	$excess\ CO_2$ (%)	N	4.39×10^{1}	6.15×10^{1}	-9.14×10^{2}	-4.29×10^{0}	1.18×10^{0}
Func-14	$[NaHS]_0$ (mol/L)	N	-5.7×10^{0}	-7.9×10^{1}	1.15×10^{1}	-1.98×10^{0}	9.67×10^{-1}
Func-17	$[NaHS]_0$ (mol/L)	$min\ CO_2\ feed$ (mol/L)	1.02×10^{2}	-5.04×10^{1}	-1.61×10^{-2}	1.02×10^{2}	-5.05×10^{1}
Func-18	x_{H_2S} (%)	N	-1.3×10^{3}	2.44×10^{5}	-1.13×10^{7}	-5.15×10^{2}	4.15×10^{4}

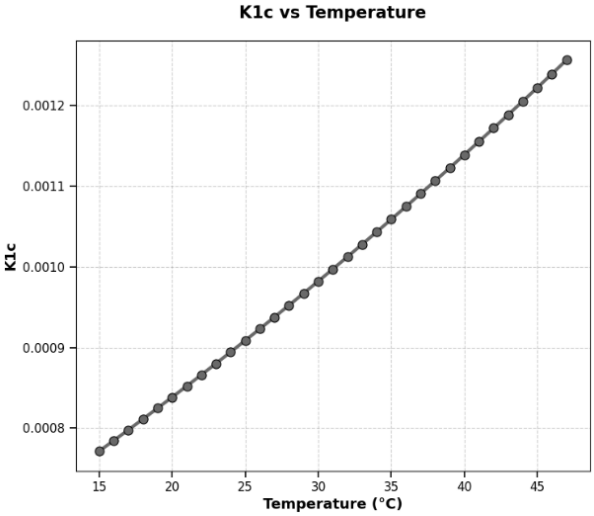
Power Law Empirical Functions

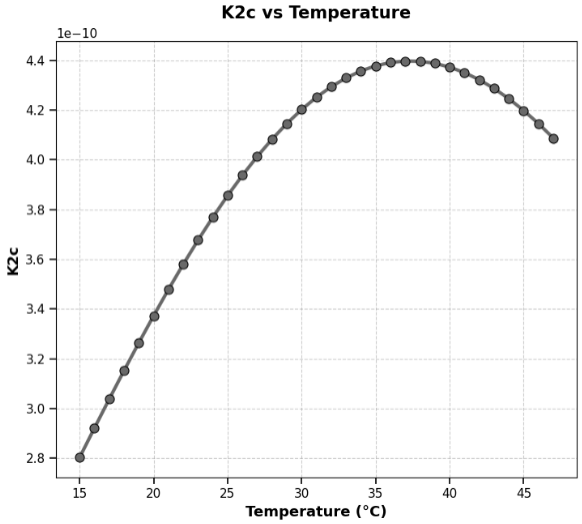
$$y = C_0 x^{C_1} + C_2$$

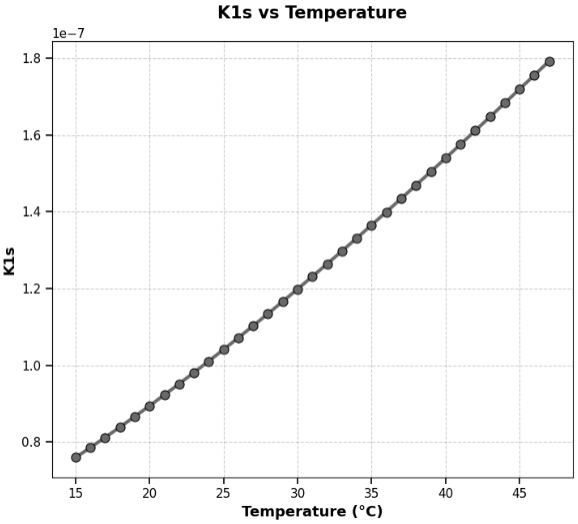
Function	Variable x	Variable y	C_0	C_1	C_2
Func-6	$P_{operating}$ (atm)	pH_N	3.14×10^{1}	-1.41×10^{-2}	-2.41×10^{1}

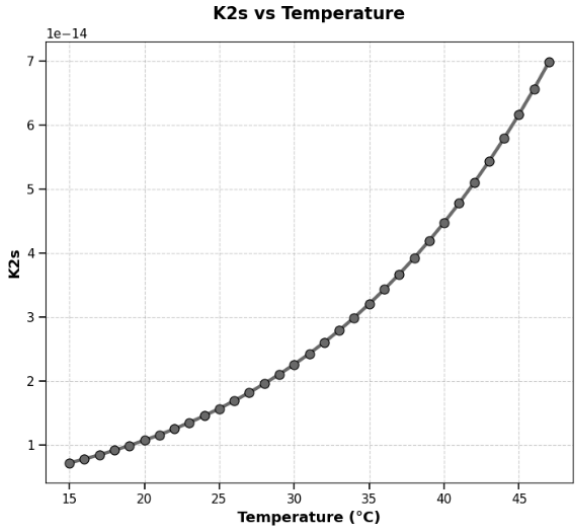
A2. System Properties and Constants Produced for Varying Temperature

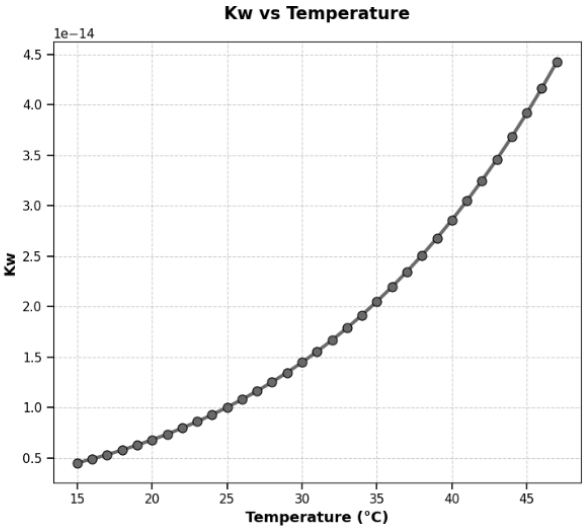


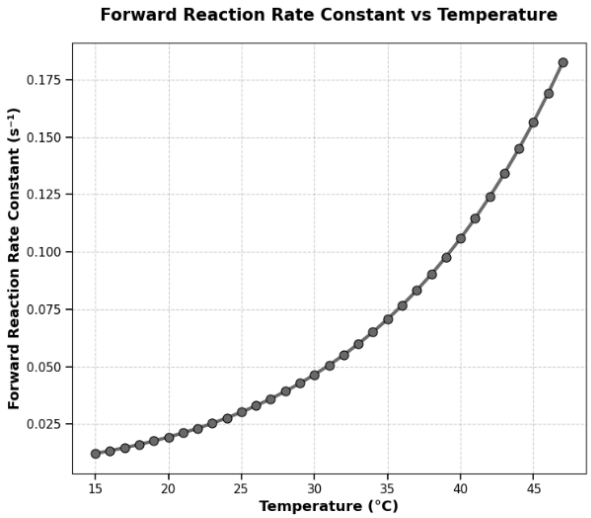


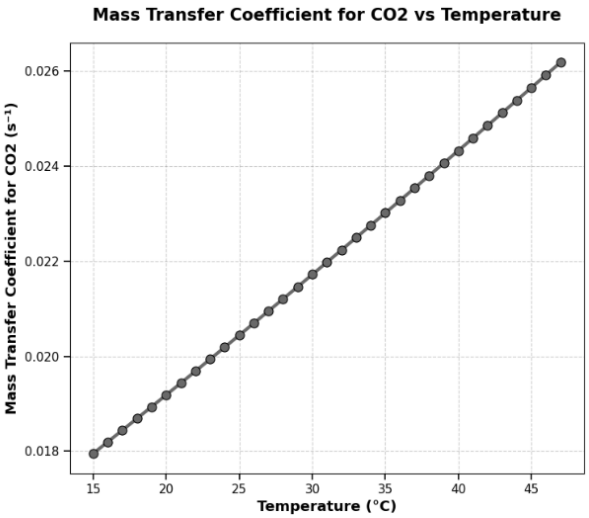


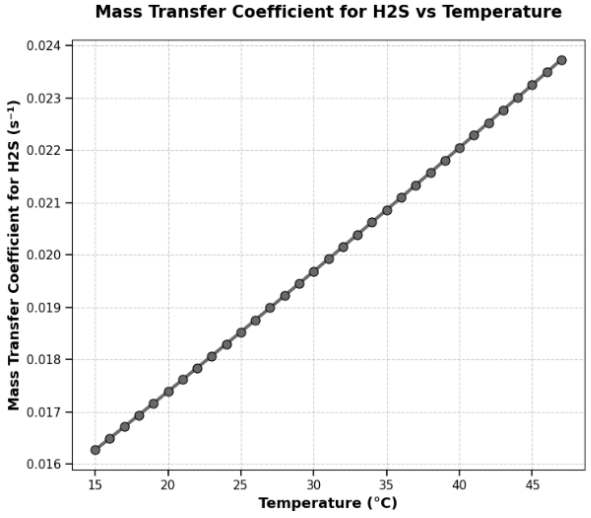


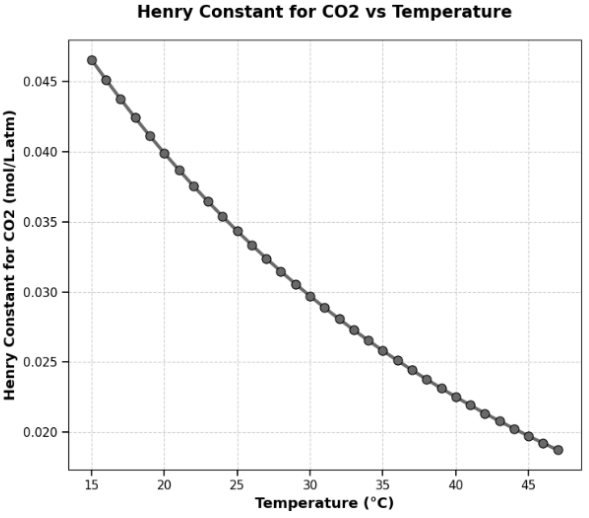


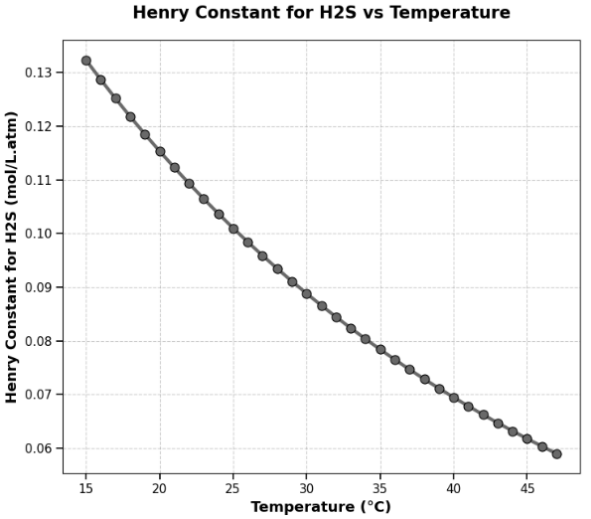


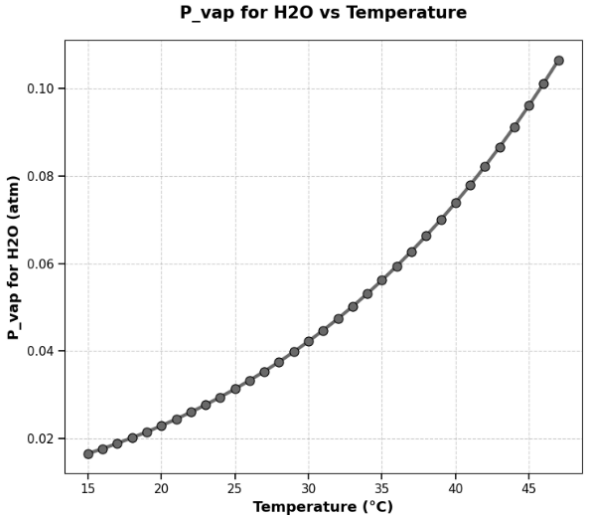

Forward Reaction Rate Constant vs Temperature
Forward Reaction Rate Constant (s⁻¹)
Temperature (°C)

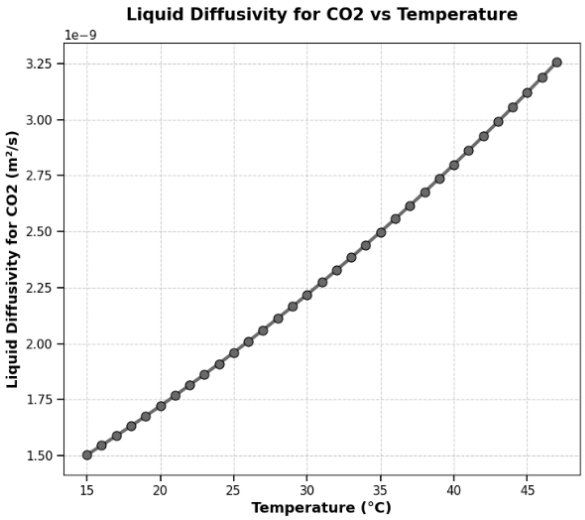

Mass Transfer Coefficient for CO2 vs Temperature
Mass Transfer Coefficient for CO2 (s⁻¹)
Temperature (°C)

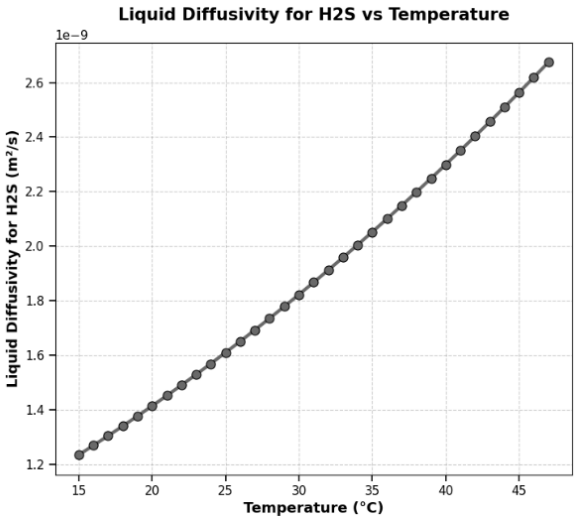

Mass Transfer Coefficient for H2S vs Temperature
Mass Transfer Coefficient for H2S (s⁻¹)
Temperature (°C)

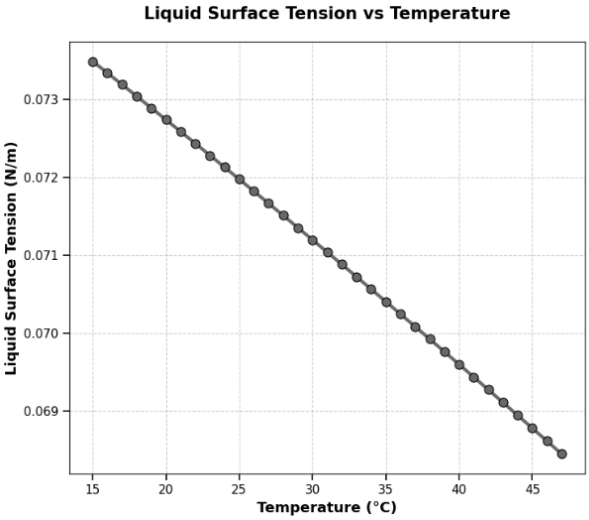

Henry Constant for CO2 vs Temperature
Henry Constant for CO2 (mol/L.atm)
Temperature (°C)

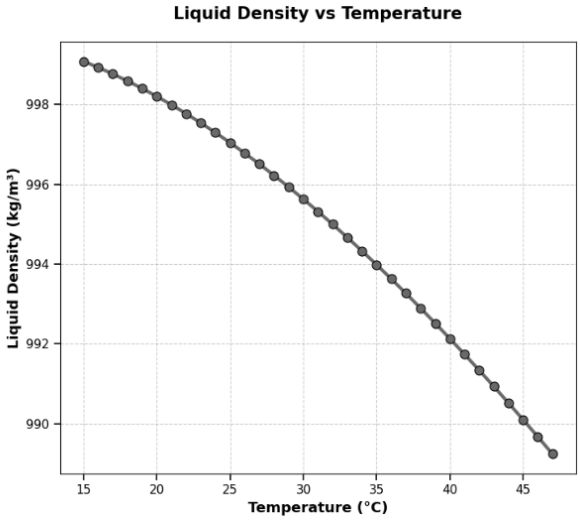

Henry Constant for H2S vs Temperature
Henry Constant for H2S (mol/L.atm)
Temperature (°C)

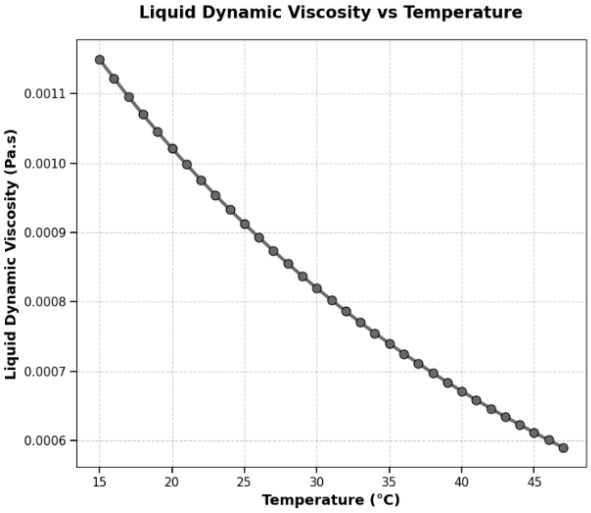

P_vap for H2O vs Temperature
P_vap for H2O (atm)
Temperature (°C)

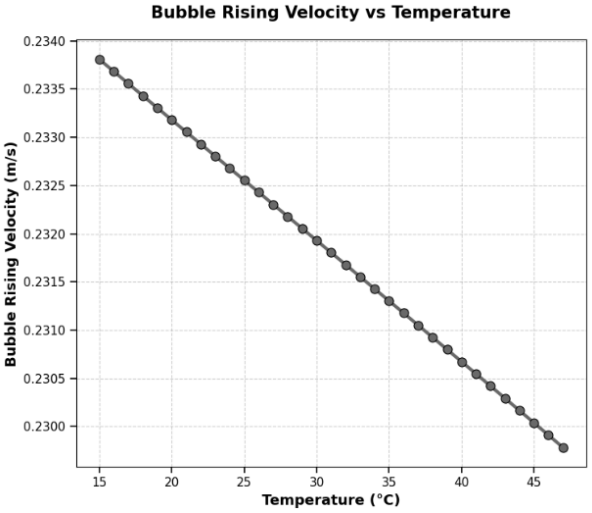

Liquid Diffusivity for CO2 vs Temperature
1e−9
Liquid Diffusivity for CO2 (m²/s)
Temperature (°C)


Liquid Diffusivity for H2S vs Temperature
1e−9
Liquid Diffusivity for H2S (m²/s)
Temperature (°C)


Liquid Surface Tension vs Temperature
Liquid Surface Tension (N/m)
Temperature (°C)


Liquid Density vs Temperature
Liquid Density (kg/m³)
Temperature (°C)


Liquid Dynamic Viscosity vs Temperature
Liquid Dynamic Viscosity (Pa.s)
Temperature (°C)


Bubble Rising Velocity vs Temperature
Bubble Rising Velocity (m/s)
Temperature (°C)

A3. Python Code for Determining K_{eq} as a Function of Temperature

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
e = np.e

temperature = np.array([15.0, 17.5, 20.0, 22.5, 25.0, 27.5, 30.0, 32.5])  # in °C
keq_inv = np.array([841, 840, 840, 843, 846, 850, 866, 878])              # 1/Keq values

# Define an exponential function for fitting
def exp_func(T, a, b, c):
    return a * np.exp(b * T) + c

# Perform curve fitting with an exponential function
exp_params, _ = curve_fit(exp_func, temperature, keq_inv, p0=(10, 0.01, 800))
a_exp, b_exp, c_exp = exp_params

# Generate fitted values using the exponential model
temperature_fit = np.linspace(min(temperature), max(temperature), 100)
keq_inv_exp_fit = exp_func(temperature_fit, a_exp, b_exp, c_exp)

# Display the exponential equation parameters
a_exp, b_exp, c_exp

plt.figure(figsize=(9, 6))
plt.scatter(temperature, keq_inv, s=100, marker='o', facecolors='red', edgecolors='black', linewidths=0.8, label='Data Points')
plt.plot(temperature_fit, keq_inv_exp_fit, color='blue', linestyle='--', linewidth=3, label=f'Exponential Fit')
plt.xlabel('Temperature (°C)', fontsize=14, fontweight='bold')
plt.ylabel('1/Keq', fontsize=14, fontweight='bold')
plt.title('Exponential Fit for 1/Keq vs Temperature', fontsize=16, fontweight='bold', pad=20)
plt.legend(loc='best', frameon=True, fancybox=True, shadow=True)
plt.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')
plt.tick_params(axis='both', which='major', labelsize=12, width=1.2)
plt.tight_layout()
plt.show()

# Display the exponential equation parameters
print("a = ", a_exp)
print("b = ", b_exp)
print("c = ", c_exp)
print("")
print("f ( T[°C] )  =  a exp(b T[°C]) + c")
print("")
print("1/Keq ( T[°C] ) =", a_exp, "exp(", b_exp, "T[°C]) +", c_exp)
```

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--MR. MOKONYANE
Keq(T) TREND 2024.h

A4. Python Program for Reactive Stripping Column Model

```
from scipy.optimize import fsolve
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Operating conditions
T_operating = 25        # °C
P_operating = 1         # atm
V_stage = 180           # L
L_flow = 1              # L/s
G_flow_out_top = 0.9    # mol/s (from top of column)
ε_gas_fraction = 5      # %
d_bubble_mm = 5         # mm
x_H2S_recovery = 99.99  # %
max_stages = 1000       # Maximum number of stages

# Feed concentrations (at the top of column)
c_NaHS_feed = 0.8        # mol/L
c_CO2_feed = 0           # mol/L
c_H2CO3_feed = 0         # mol/L
c_HCO3_feed = 0          # mol/L
c_CO3_feed = 0           # mol/L
c_H2S_feed = 0           # mol/L
c_S_feed = 0             # mol/L
```

```
# Equilibrium constants and property functions
def K_eq(T):
    K_eq = 1/(0.04020944211255422*(np.e**(0.2130531302959603*T)) + 838.3007987989027)
    return K_eq

def K_1c(T):
    K_1c = 10**(-0.994 - (610.5/(T + 273.15)))
    return K_1c

def K_2c(T):
    K_2c = 10**(452.094 - (21263.61/(T + 273.15)) - (68.483143*np.log((T + 273.15))))
    return K_2c

def K_1s(T):
    K_1s = 10**(-32.55 - (1519.44/(T + 273.15)) + (15.672*np.log10((T + 273.15))) - (0.02722*(T + 273.15)))
    return K_1s

def K_2s(T):
    K_2s = 10**(-(4500/(T + 273.15)) - (12.6*np.log10((T + 273.15)/298.15)) + 1.29)
    return K_2s

def K_w(T):
    K_w = 10**(-(142613.6/(T + 273.15)) - (4229.195*np.log10((T + 273.15))) + (9.7384*(T + 273.15)) -
               (0.0129638*(T + 273.15)**2) + (1.15068e-5 *(T + 273.15)**3) - (4.602e-9 *(T + 273.15)**4) + 8909.483)
    return K_w

def k_f(T):
    k_f = np.e**(22.66 - (7799/(T + 273.15)))
    return k_f

# Physical property functions
d_bubble_m = d_bubble_mm/1000
a = (6*(ε_gas_fraction/100))/d_bubble_m
g = 9.182
MM_H2O = 18.01528
```

```python
def liquid_surface_tension(T):
    factor = (647.15 - (T + 273.15))/647.15
    σ_L = (235.8e-3)*(1 - 0.625*factor)*(factor**1.256)
    return σ_L

def liquid_density(T):
    ρ_L = -13.851 + (0.64038*(T + 273.15)) - (0.0019124*((T + 273.15)**2)) + ((1.8211e-6)*((T + 273.15)**3))
    return ρ_L*(MM_H2O/1000)*(10**3)

def liquid_dynamic_viscosity(T):
    μ_L = np.e**(-52.843 + (3703.6/(T + 273.15)) + (5.866*np.log(T + 273.15)) - ((5.879e-29)*((T + 273.15)**10)))
    return μ_L

def bubble_rising_velocity(T):
    σ_L = liquid_surface_tension(T)
    ρ_L = liquid_density(T)
    u_b = (((2.14*σ_L)/(d_bubble_m*ρ_L)) + (0.505*g*d_bubble_m))**0.5
    return u_b

def liquid_diffusivity_CO2(T):
    D_L_25_CO2 = (1.96e-5)/(100**2)
    μ_L_25 = liquid_dynamic_viscosity(25)
    constant_25_CO2 = D_L_25_CO2*μ_L_25/(25 + 273.15)
    D_L_CO2 = constant_25_CO2*(T + 273.15)/liquid_dynamic_viscosity(T)
    return D_L_CO2

def liquid_side_mass_transfer_coefficient_CO2(T):
    k_L_CO2 = 2*((liquid_diffusivity_CO2(T)*bubble_rising_velocity(T))/(np.pi*d_bubble_m))**0.5
    return k_L_CO2

def k_La_CO2_func(T):
    k_La_CO2 = a*liquid_side_mass_transfer_coefficient_CO2(T)
    return k_La_CO2
```

```python
def liquid_diffusivity_H2S(T):
    D_L_25_H2S = (1.61e-5)/(100**2)
    μ_L_25 = liquid_dynamic_viscosity(25)
    constant_25_H2S = D_L_25_H2S*μ_L_25/(25 + 273.15)
    D_L_H2S = constant_25_H2S*(T + 273.15)/liquid_dynamic_viscosity(T)
    return D_L_H2S

def liquid_side_mass_transfer_coefficient_H2S(T):
    k_L_H2S = 2*((liquid_diffusivity_H2S(T)*bubble_rising_velocity(T))/(np.pi*d_bubble_m))**0.5
    return k_L_H2S

def k_La_H2S_func(T):
    k_La_H2S = a*liquid_side_mass_transfer_coefficient_H2S(T)
    return k_La_H2S

def P_vap_H2O(T):
    A_H2O = 4.6543
    B_H2O = 1435.264
    C_H2O = -64.848
    P_vap_H2O = 10**(A_H2O - (B_H2O / ((T + 273.15) + C_H2O)))
    return P_vap_H2O * 0.986923

def k_H_CO2(T):
    k_H_CO2_start = 0.034*np.e**(2600*((1/(T + 273.15)) - (1/298.15)))
    k_H_CO2 = k_H_CO2_start*liquid_density(T)*(1.01325/1000)
    return k_H_CO2

def k_H_H2S(T):
    k_H_H2S_start = 0.1*np.e**(2300*((1/(T + 273.15)) - (1/298.15)))
    k_H_H2S = k_H_H2S_start*liquid_density(T)*(1.01325/1000)
    return k_H_H2S
```

```
# Calculate constants at operating temperature
K_eq_val = K_eq(T_operating)
K_1c_val = K_1c(T_operating)
K_2c_val = K_2c(T_operating)
K_1s_val = K_1s(T_operating)
K_2s_val = K_2s(T_operating)
K_w_val = K_w(T_operating)
k_f_val = k_f(T_operating)
k_La_CO2 = k_La_CO2_func(T_operating)
k_La_H2S = k_La_H2S_func(T_operating)
P_vap_H2O_val = P_vap_H2O(T_operating)
k_H_CO2_val = k_H_CO2(T_operating)
k_H_H2S_val = k_H_H2S(T_operating)

def solve_stage(liquid_in, gas_out, stage_num):
    """
    Solve a single stage given liquid inlet and gas outlet conditions

    liquid_in: dict with liquid concentrations entering the stage
    gas_out: dict with gas flow rate and compositions leaving the stage
    stage_num: stage number (1 is top stage)

    Returns: liquid_out, gas_in dictionaries
    """

    # Extract input conditions
    c_Na_in = liquid_in['Na']
    c_H2S_in = liquid_in['H2S']
    c_HS_in = liquid_in['HS']
    c_S_in = liquid_in['S']
    c_CO2_in = liquid_in['CO2']
    c_H2CO3_in = liquid_in['H2CO3']
    c_HCO3_in = liquid_in['HCO3']
    c_CO3_in = liquid_in['CO3']

    G_flow_out = gas_out['flow']
    y_H2O_out = gas_out['H2O']
    y_H2S_out = gas_out['H2S']
    y_CO2_out = gas_out['CO2']
```

```
    # Calculate interfacial concentrations
    c_H2S_i_stage = k_H_H2S_val * y_H2S_out * P_operating
    c_CO2_i_stage = k_H_CO2_val * y_CO2_out * P_operating

    # Sodium is conserved
    c_Na_out = c_Na_in

    # Define equations for this stage with transformed variables
    def equations(vars):
        pH, y_scaled, z_scaled = vars  # pH, 1000*c_H2CO3_out, 1000*c_H2S_out

        # Convert back to actual concentrations
        c_H_out = 10**(-pH)
        c_H2CO3_out = y_scaled / 1000.0
        c_H2S_out = z_scaled / 1000.0

        # Calculate other concentrations
        c_OH_out = K_w_val / c_H_out
        c_HCO3_out = c_H2CO3_out * K_1c_val / c_H_out
        c_CO3_out = c_HCO3_out * K_2c_val / c_H_out
        c_HS_out = c_H2S_out * K_1s_val / c_H_out
        c_S_out = c_HS_out * K_2s_val / c_H_out

        c_CO2_out = ((L_flow*(c_H2CO3_out + c_HCO3_out + c_CO3_out) - L_flow*(c_H2CO3_in + c_HCO3_in + c_CO3_in)) /
                     (k_f_val*(1 - ε_gas_fraction/100)*V_stage)) + (c_H2CO3_out/K_eq_val)

        term2 = k_La_CO2*V_stage*(c_CO2_i_stage - c_CO2_out) + k_La_H2S*V_stage*(c_H2S_i_stage - c_H2S_out)
        G_flow_in = G_flow_out + (term2/(1 - y_H2O_out))

        y_H2S_in = ((G_flow_out*y_H2S_out) + (k_La_H2S*V_stage*(c_H2S_i_stage - c_H2S_out))) / G_flow_in
        y_CO2_in = ((G_flow_out*y_CO2_out) + (k_La_CO2*V_stage*(c_CO2_i_stage - c_CO2_out))) / G_flow_in

        eq1 = c_Na_out + c_H_out - c_HCO3_out - 2*c_CO3_out - c_OH_out - c_HS_out - 2*c_S_out
        eq2 = (G_flow_out*y_CO2_out + L_flow*(c_CO2_out + c_H2CO3_out + c_HCO3_out + c_CO3_out) -
               G_flow_in*y_CO2_in - L_flow*(c_CO2_in + c_H2CO3_in + c_HCO3_in + c_CO3_in))
        eq3 = (G_flow_out*y_H2S_out + L_flow*(c_H2S_out + c_HS_out + c_S_out) -
               G_flow_in*y_H2S_in - L_flow*(c_H2S_in + c_HS_in + c_S_in))

        return [eq1, eq2, eq3]
```

```
# Try multiple initial guesses with transformed variables
initial_guesses = [
    [7.0, 1.0, 1.0],      # pH=7, 1000*[H2CO3]=1, 1000*[H2S]=1
    [6.8, 1.2, 1.0],      # pH=6.8, 1000*[H2CO3]=1.2, 1000*[H2S]=1
    [7.2, 0.8, 1.0],      # pH=7.2, 1000*[H2CO3]=0.8, 1000*[H2S]=1
    [7.0, 1.0, 1.2],      # pH=7, 1000*[H2CO3]=1, 1000*[H2S]=1.2
    [7.0, 1.0, 0.8],      # pH=7, 1000*[H2CO3]=1, 1000*[H2S]=0.8
    [6.5, 1.5, 1.5],      # pH=6.5, 1000*[H2CO3]=1.5, 1000*[H2S]=1.5
    [7.5, 0.5, 0.5],      # pH=7.5, 1000*[H2CO3]=0.5, 1000*[H2S]=0.5
]

best_solution = None
best_residual = 1e10
best_equations_residual = None

for guess in initial_guesses:
    try:
        solution = fsolve(equations, guess, xtol=1e-12)
        residual_vals = equations(solution)
        residual = sum([r**2 for r in residual_vals])

        # Check if solution is physically reasonable
        pH_val, y_scaled_val, z_scaled_val = solution
        c_H_val = 10**(-pH_val)
        c_H2CO3_val = y_scaled_val / 1000.0
        c_H2S_val = z_scaled_val / 1000.0

        # Physical constraints
        if (4 <= pH_val <= 14 and c_H2CO3_val > 0 and c_H2S_val > 0 and
            y_scaled_val > 0 and z_scaled_val > 0 and residual < best_residual):
            best_residual = residual
            best_solution = solution
            best_equations_residual = residual_vals
    except:
        continue

if best_solution is None:
    raise ValueError(f"Could not converge stage {stage_num}")
```

```
# Extract solution and convert back to actual concentrations
pH_out, y_scaled_out, z_scaled_out = best_solution
c_H_out = 10**(-pH_out)
c_H2CO3_out = y_scaled_out / 1000.0
c_H2S_out = z_scaled_out / 1000.0

# Calculate derived concentrations
c_OH_out = K_w_val / c_H_out
c_HCO3_out = c_H2CO3_out * K_1c_val / c_H_out
c_CO3_out = c_HCO3_out * K_2c_val / c_H_out
c_HS_out = c_H2S_out * K_1s_val / c_H_out
c_S_out = c_HS_out * K_2s_val / c_H_out

c_CO2_out = ((L_flow*(c_H2CO3_out + c_HCO3_out + c_CO3_out) - L_flow*(c_H2CO3_in + c_HCO3_in + c_CO3_in)) /
             (k_f_val*(1 - ε_gas_fraction/100)*V_stage)) + (c_H2CO3_out/K_eq_val)

term2 = k_La_CO2*V_stage*(c_CO2_i_stage - c_CO2_out) + k_La_H2S*V_stage*(c_H2S_i_stage - c_H2S_out)
G_flow_in = G_flow_out + (term2/(1 - y_H2O_out))

y_H2S_in = ((G_flow_out*y_H2S_out) + (k_La_H2S*V_stage*(c_H2S_i_stage - c_H2S_out))) / G_flow_in
y_CO2_in = ((G_flow_out*y_CO2_out) + (k_La_CO2*V_stage*(c_CO2_i_stage - c_CO2_out))) / G_flow_in
y_H2O_in = y_H2O_out

# Package outputs
liquid_out = {
    'Na': c_Na_out,
    'H': c_H_out,
    'OH': c_OH_out,
    'CO2': c_CO2_out,
    'H2CO3': c_H2CO3_out,
    'HCO3': c_HCO3_out,
    'CO3': c_CO3_out,
    'H2S': c_H2S_out,
    'HS': c_HS_out,
    'S': c_S_out,
    'pH': pH_out
}
```

```
    gas_in = {
        'flow': G_flow_in,
        'H2O': y_H2O_in,
        'CO2': y_CO2_in,
        'H2S': y_H2S_in
    }

    # Return individual residuals for plotting
    residuals = {
        'cb': best_equations_residual[0],   # Charge balance
        'cdb': best_equations_residual[1],  # Carbon dioxide balance
        'sb': best_equations_residual[2]    # Sulphur balance
    }

    return liquid_out, gas_in, best_residual, residuals

# Initialize column simulation
print("="*120)
print("Solving stripping column with automatic stage determination...")
print("="*120)

# Initialize liquid feed at top (stage 1)
liquid_feed = {
    'Na': c_NaHS_feed,
    'H2S': c_H2S_feed,
    'HS': c_NaHS_feed,  # Initial HS- from NaHS feed
    'S': c_S_feed,
    'CO2': c_CO2_feed,
    'H2CO3': c_H2CO3_feed,
    'HCO3': c_HCO3_feed,
    'CO3': c_CO3_feed
}

# Initialize gas outlet from top (stage 1)
y_H2O_top = P_vap_H2O_val/P_operating
y_H2S_top = ((x_H2S_recovery/100)*c_NaHS_feed*L_flow)/G_flow_out_top
y_CO2_top = 1 - y_H2O_top - y_H2S_top
```

```
gas_top = {
    'flow': G_flow_out_top,
    'H2O': y_H2O_top,
    'CO2': y_CO2_top,
    'H2S': y_H2S_top
}

# Calculate target sulphur remaining in liquid (what should NOT be stripped)
total_sulphur_feed = c_NaHS_feed  # mol/L of sulphur from NaHS
target_sulphur_remaining = total_sulphur_feed * (1 - x_H2S_recovery/100)

# Storage for results
stage_results = {}
liquid_profiles = []
gas_profiles = []
residual_profiles = []

# Solve column from top to bottom
current_liquid = liquid_feed.copy()
current_gas = gas_top.copy()

# Convergence criteria
converged = False
final_stage = 0
previous_total_sulphur = None  # Track previous stage sulphur content

for stage in range(1, max_stages + 1):
    print(f"Solving stage {stage}...")

    try:
        liquid_out, gas_in, residual, residuals = solve_stage(current_liquid, current_gas, stage)

        # Calculate sulphur mass balance for this stage
        total_sulphur_out = liquid_out['H2S'] + liquid_out['HS'] + liquid_out['S']
```

```
        # Check if we should stop BEFORE reaching the target
        if previous_total_sulphur is not None:
            # If current stage would go below target, stop at previous stage
            if total_sulphur_out <= target_sulphur_remaining:
                print(f"\n*** CONVERGED at stage {stage-1} (stopping just before target) ***")
                print(f"Target sulphur remaining: {target_sulphur_remaining:.6e} mol/L")
                print(f"Stage {stage-1} sulphur remaining: {previous_total_sulphur:.6e} mol/L")
                print(f"Stage {stage} would have: {total_sulphur_out:.6e} mol/L")
                converged = True
                final_stage = stage - 1
                break

        # Store results for this stage
        stage_results[stage] = {
            'liquid_in': current_liquid.copy(),
            'liquid_out': liquid_out.copy(),
            'gas_in': gas_in.copy(),
            'gas_out': current_gas.copy(),
            'residual': residual,
            'residuals': residuals
        }

        liquid_profiles.append({
            'stage': stage,
            **liquid_out
        })

        gas_profiles.append({
            'stage': stage,
            'flow_in': gas_in['flow'],
            'flow_out': current_gas['flow'],
            'y_H2O_in': gas_in['H2O'],
            'y_CO2_in': gas_in['CO2'],
            'y_H2S_in': gas_in['H2S'],
            'y_H2O_out': current_gas['H2O'],
            'y_CO2_out': current_gas['CO2'],
            'y_H2S_out': current_gas['H2S']
        })
```

```
        residual_profiles.append({
            'stage': stage,
            'cb': residuals['cb'],
            'cdb': residuals['cdb'],
            'sb': residuals['sb']
        })

        # Store current sulphur content for next iteration
        previous_total_sulphur = total_sulphur_out

        # Update for next stage
        current_liquid = liquid_out.copy()
        current_gas = gas_in.copy()

    except Exception as e:
        print(f"Error solving stage {stage}: {e}")
        break

if not converged:
    print(f"\nWARNING: Did not converge within {max_stages} stages")
    final_stage = len(liquid_profiles)
else:
    print(f"\nColumn converged within {final_stage} stages")

print("="*120)
print("Column solution complete!")

# Convert to DataFrames for easier analysis
df_liquid = pd.DataFrame(liquid_profiles)
df_gas = pd.DataFrame(gas_profiles)
df_residuals = pd.DataFrame(residual_profiles)

# Calculate actual recovery achieved
if final_stage > 0:
    final_liquid = stage_results[final_stage]['liquid_out']
    final_sulphur_remaining = final_liquid['H2S'] + final_liquid['HS'] + final_liquid['S']
    actual_recovery = (1 - final_sulphur_remaining/total_sulphur_feed) * 100
```

```
    print(f"\nFINAL PERFORMANCE:")
    print(f"Operating temperature: {T_operating} °C")
    print(f"Operating pressure: {P_operating} atm")
    print(f"Number of stages required: {final_stage}")
    print(f"Actual H2S recovery: {actual_recovery:.5f} %")
    print(f"Target H2S recovery: {x_H2S_recovery:.5f} %")
    print(f"Final sulphur remaining: {final_sulphur_remaining:.6e} mol/L")
    print(f"Target sulphur remaining: {target_sulphur_remaining:.6e} mol/L")

# Display detailed results for key stages
if final_stage >= 1:
    print("\n" + "="*120)
    print("DETAILED RESULTS FOR STAGE 1 (TOP OF COLUMN - FEED STAGE)")
    print("="*120)

    stage_1 = stage_results[1]

    print("LIQUID INLET CONDITIONS (FEED):")
    for key, value in stage_1['liquid_in'].items():
        if isinstance(value, float):
            print(f"  {key:8s}: {value:.8e} mol/L")

    print("\nLIQUID OUTLET CONDITIONS:")
    for key, value in stage_1['liquid_out'].items():
        if key == 'pH':
            print(f"  {key:8s}: {value:.4f}")
        elif isinstance(value, float):
            print(f"  {key:8s}: {value:.8e} mol/L")

    print(f"\nSUM OF THE SQUARES OF THE BALANCE EQUATIONS: {stage_1['residual']:.2e}")

if final_stage >= 2:
    print("\n" + "="*120)
    print(f"DETAILED RESULTS FOR STAGE {final_stage} (BOTTOM OF COLUMN)")
    print("="*120)

    stage_last = stage_results[final_stage]
```

```
    print("LIQUID OUTLET CONDITIONS (FINAL):")
    for key, value in stage_last['liquid_out'].items():
        if key == 'pH':
            print(f"  {key:8s}: {value:.4f}")
        elif isinstance(value, float):
            print(f"  {key:8s}: {value:.8e} mol/L")

    print(f"\nSUM OF THE SQUARES OF THE BALANCE EQUATIONS: {stage_last['residual']:.2e}")

# ================================================================================
# SYSTEM PROPERTIES
# ================================================================================

print('')
print("="*120)
print("SYSTEM PROPERTIES")
print("="*120)
print('')
print('~~~~~ EQUILIBRIUM CONSTANTS ~~~~~')
print("K_eq (@", T_operating, "°C) =", K_eq_val)
print("K_1c (@", T_operating, "°C) =", K_1c_val)
print("K_2c (@", T_operating, "°C) =", K_2c_val)
print("K_1s (@", T_operating, "°C) =", K_1s_val)
print("K_2s (@", T_operating, "°C) =", K_2s_val)
print("K_w (@", T_operating, "°C)  =", K_w_val)
print('')
print('~~~~~ FORWARD RATE CONSTANT ~~~~~')
print("k_f (@", T_operating, "°C)  =", k_f_val, "s^-1")
print('')
print('~~~~~ FLUID PROPERTIES ~~~~~')
print("a =", a, "1/m")
print("σ_L (@", T_operating, "°C) =", liquid_surface_tension(T_operating), "N/m")
print("ρ_L (@", T_operating, "°C) =", liquid_density(T_operating), "kg/m^3")
print("μ_L (@", T_operating, "°C) =", liquid_dynamic_viscosity(T_operating), "Pa.s")
print("u_b (@", T_operating, "°C) =", bubble_rising_velocity(T_operating), "m/s")
print('')
```

```
print('~~~~~ MASS TRANSFER COEFFICIENT FOR CO2 ~~~~~')
print("k_La_CO2 (@", T_operating, "°C) =", k_La_CO2, "s^-1")
print('')
print('~~~~~ MASS TRANSFER COEFFICIENT FOR H2S ~~~~~')
print("k_La_H2S (@", T_operating, "°C) =", k_La_H2S, "s^-1")
print('')
print('~~~~~ HENRY COEFFICIENT FOR CO2 ~~~~~')
print("k_H_CO2 (@", T_operating, "°C) =", k_H_CO2_val, "mol/L.atm")
print('')
print('~~~~~ HENRY COEFFICIENT FOR H2S ~~~~~')
print("k_H_H2S (@", T_operating, "°C) =", k_H_H2S_val, "mol/L.atm")
print('')
print('~~~~~ VAPOR PRESSURE FOR H2O ~~~~~')
print("P_vap_H2O (@", T_operating, "°C) =", P_vap_H2O_val, "atm")
print('')

# ============================================================================
# PLOTTING SECTION
# ============================================================================

print("="*120)
print("PLOTTING SECTION")
print("="*120)

# Matplotlib parameters for plots
plt.style.use('default')
plt.rcParams['figure.figsize'] = [14, 10]
plt.rcParams['font.size'] = 12
plt.rcParams['axes.titlesize'] = 16
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['legend.fontsize'] = 12
plt.rcParams['xtick.labelsize'] = 11
plt.rcParams['ytick.labelsize'] = 11

# 1. INDIVIDUAL LIQUID CONCENTRATIONS PLOTS
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 12))
```

```
# Sulphur species
ax1.semilogy(df_liquid['stage'], df_liquid['HS'], 'o-', label='HS⁻', linewidth=3, markersize=6, color='green',
             markeredgecolor='black', markeredgewidth=0.8)
ax1.semilogy(df_liquid['stage'], df_liquid['H2S'], 'o-', label='H₂S', linewidth=3, markersize=6, color='limegreen',
             markeredgecolor='black', markeredgewidth=0.8)
ax1.semilogy(df_liquid['stage'], df_liquid['S'], 'o-', label='S²⁻', linewidth=3, markersize=6, color='lime',
             markeredgecolor='black', markeredgewidth=0.8)
ax1.set_xlabel('Stage Number', fontsize=14, fontweight='bold')
ax1.set_ylabel('Concentration (mol/L)', fontsize=14, fontweight='bold')
ax1.set_title('Individual Sulphur Species Concentrations in Liquid Phase', fontsize=16, fontweight='bold', pad=20)
ax1.legend(fontsize=12, frameon=True, fancybox=True, shadow=True)
ax1.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')

# Carbon species
ax2.semilogy(df_liquid['stage'], df_liquid['HCO3'], 'o-', label='HCO₃⁻', linewidth=3, markersize=6, color='purple',
             markeredgecolor='black', markeredgewidth=0.8)
ax2.semilogy(df_liquid['stage'], df_liquid['CO2'], 'o-', label='CO₂', linewidth=3, markersize=6, color='darkviolet',
             markeredgecolor='black', markeredgewidth=0.8)
ax2.semilogy(df_liquid['stage'], df_liquid['CO3'], 'o-', label='CO₃²⁻', linewidth=3, markersize=6, color='magenta',
             markeredgecolor='black', markeredgewidth=0.8)
ax2.semilogy(df_liquid['stage'], df_liquid['H2CO3'], 'o-', label='H₂CO₃', linewidth=3, markersize=6, color='deeppink',
             markeredgecolor='black', markeredgewidth=0.8)
ax2.set_xlabel('Stage Number', fontsize=14, fontweight='bold')
ax2.set_ylabel('Concentration (mol/L)', fontsize=14, fontweight='bold')
ax2.set_title('Individual Carbon Species Concentrations in Liquid Phase', fontsize=16, fontweight='bold', pad=20)
ax2.legend(fontsize=12, frameon=True, fancybox=True, shadow=True)
ax2.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')

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

```
# 2. TOTAL LIQUID CONCENTRATIONS PLOT
plt.figure(figsize=(12, 8))
total_sulphur = df_liquid['H2S'] + df_liquid['HS'] + df_liquid['S']
total_carbon = df_liquid['CO2'] + df_liquid['H2CO3'] + df_liquid['HCO3'] + df_liquid['CO3']
plt.semilogy(df_liquid['stage'], total_carbon, 'o-', label='Total Carbon', linewidth=3, markersize=6,
             color='hotpink', markeredgecolor='black', markeredgewidth=0.8)
plt.semilogy(df_liquid['stage'], total_sulphur, 'o-', label='Total Sulphur', linewidth=3, markersize=6,
             color='springgreen', markeredgecolor='black', markeredgewidth=0.8)
plt.axhline(y=target_sulphur_remaining, linestyle='--', linewidth=2, label=f'Target ({target_sulphur_remaining:.5e} mol/L)',
            color='black')
plt.xlabel('Stage Number', fontsize=14, fontweight='bold')
plt.ylabel('Total Concentration (mol/L)', fontsize=14, fontweight='bold')
plt.title('Total Carbon and Sulphur Species Concentrations in Liquid Phase', fontsize=16, fontweight='bold', pad=20)
plt.legend(fontsize=12, frameon=True, fancybox=True, shadow=True)
plt.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')
plt.tight_layout()
plt.show()

# 3. GAS PHASE PROPERTIES PLOT
plt.figure(figsize=(12, 8))
plt.plot(df_gas['stage'], df_gas['y_H2O_out'], 'o-', label='H₂O Vapor Fraction', linewidth=3, markersize=6,
         color='cyan', markeredgecolor='black', markeredgewidth=0.8)
plt.plot(df_gas['stage'], df_gas['y_CO2_out'], 'o-', label='CO₂ Vapor Fraction', linewidth=3, markersize=6,
         color='violet', markeredgecolor='black', markeredgewidth=0.8)
plt.plot(df_gas['stage'], df_gas['y_H2S_out'], 'o-', label='H₂S Vapor Fraction', linewidth=3, markersize=6,
         color='greenyellow', markeredgecolor='black', markeredgewidth=0.8)
plt.plot(df_gas['stage'], df_gas['flow_out'], 'o-', label='Vapor Flowrate (mol/s)', linewidth=3, markersize=6,
         color='orange', markeredgecolor='black', markeredgewidth=0.8)
plt.xlabel('Stage Number', fontsize=14, fontweight='bold')
plt.ylabel('Vapor Fractions and Flowrate (mol/s)', fontsize=14, fontweight='bold')
plt.title('Gas Phase Properties: Vapor Fractions and Flowrate', fontsize=16, fontweight='bold', pad=20)
plt.legend(fontsize=12, frameon=True, fancybox=True, shadow=True)
plt.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')
plt.tight_layout()
plt.show()
```

```
# 4. pH PROFILE PLOT
plt.figure(figsize=(12, 8))
plt.plot(df_liquid['stage'], df_liquid['pH'], 'o-', label='pH', linewidth=3, markersize=6, color='firebrick',
             markeredgecolor='black', markeredgewidth=0.8)
plt.xlabel('Stage Number', fontsize=14, fontweight='bold')
plt.ylabel('pH Value', fontsize=14, fontweight='bold')
plt.title('Liquid Phase pH Profile Throughout the Column', fontsize=16, fontweight='bold', pad=20)
plt.legend(fontsize=12, frameon=True, fancybox=True, shadow=True)
plt.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')
plt.tight_layout()
plt.show()

# 5. RESIDUALS PLOT
plt.figure(figsize=(12, 8))
sum_of_squares = (df_residuals['cb']**2) + (df_residuals['cdb']**2) + (df_residuals['sb']**2)
plt.plot(df_gas['stage'], df_residuals['cb'], 'o-', label='Charge Balance', linewidth=3, markersize=6,
         color='dodgerblue', markeredgecolor='black', markeredgewidth=0.8)
plt.plot(df_gas['stage'], df_residuals['cdb'], 'o-', label='Carbon Dioxide Balance', linewidth=3, markersize=6,
         color='blue', markeredgecolor='black', markeredgewidth=0.8)
plt.plot(df_gas['stage'], df_residuals['sb'], 'o-', label='Sulphur Balance', linewidth=3, markersize=6,
         color='cornflowerblue', markeredgecolor='black', markeredgewidth=0.8)
plt.plot(df_gas['stage'], sum_of_squares, 'o-', label='Sum of Squares', linewidth=3, markersize=6,
         color='red', markeredgecolor='black', markeredgewidth=0.8)
plt.xlabel('Stage Number', fontsize=14, fontweight='bold')
plt.ylabel('Residual Values', fontsize=14, fontweight='bold')
plt.title('Model Convergence: Residual Values for Mass and Charge Balances', fontsize=16, fontweight='bold', pad=20)
plt.legend(fontsize=12, frameon=True, fancybox=True, shadow=True)
plt.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')
plt.tight_layout()
plt.show()

# 6. EQUILIBRIUM CONSTANT VALIDATION (SANITY CHECKS)
fig, ((ax1, ax2), (ax3, ax4), (ax5, ax6)) = plt.subplots(3, 2, figsize=(16, 18))
```

```
# CHECK 1: Carbonic acid first dissociation
ax1.plot(df_liquid['stage'], K_1c_val/df_liquid['H'], 'o-', label='K₁c/[H⁺]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax1.plot(df_liquid['stage'], df_liquid['HCO3']/df_liquid['H2CO3'], 'o-', label='[HCO₃⁻]/[H₂CO₃]', linewidth=3,
         markersize=6, markeredgecolor='black', markeredgewidth=0.8)
ax1.set_xlabel('Stage Number', fontsize=12, fontweight='bold')
ax1.set_ylabel('Ratio Value', fontsize=12, fontweight='bold')
ax1.set_title('Carbonic Acid 1st Dissociation:\nK₁c/[H⁺] = [HCO₃⁻]/[H₂CO₃]', fontsize=14, fontweight='bold')
ax1.legend(fontsize=10, frameon=True)
ax1.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')

# CHECK 2: Carbonic acid second dissociation
ax2.plot(df_liquid['stage'], K_2c_val/df_liquid['H'], 'o-', label='K₂c/[H⁺]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax2.plot(df_liquid['stage'], df_liquid['CO3']/df_liquid['HCO3'], 'o-', label='[CO₃²⁻]/[HCO₃⁻]', linewidth=3,
         markersize=6, markeredgecolor='black', markeredgewidth=0.8)
ax2.set_xlabel('Stage Number', fontsize=12, fontweight='bold')
ax2.set_ylabel('Ratio Value', fontsize=12, fontweight='bold')
ax2.set_title('Carbonic Acid 2nd Dissociation:\nK₂c/[H⁺] = [CO₃²⁻]/[HCO₃⁻]', fontsize=14, fontweight='bold')
ax2.legend(fontsize=10, frameon=True)
ax2.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')

# CHECK 3: Hydrogen sulfide first dissociation
ax3.plot(df_liquid['stage'], K_1s_val/df_liquid['H'], 'o-', label='K₁s/[H⁺]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax3.plot(df_liquid['stage'], df_liquid['HS']/df_liquid['H2S'], 'o-', label='[HS⁻]/[H₂S]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax3.set_xlabel('Stage Number', fontsize=12, fontweight='bold')
ax3.set_ylabel('Ratio Value', fontsize=12, fontweight='bold')
ax3.set_title('Hydrogen Sulfide 1st Dissociation:\nK₁s/[H⁺] = [HS⁻]/[H₂S]', fontsize=14, fontweight='bold')
ax3.legend(fontsize=10, frameon=True)
ax3.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')
```

```
# CHECK 4: Hydrogen sulfide second dissociation
ax4.plot(df_liquid['stage'], K_2s_val/df_liquid['H'], 'o-', label='K₂s/[H⁺]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax4.plot(df_liquid['stage'], df_liquid['S']/df_liquid['HS'], 'o-', label='[S²⁻]/[HS⁻]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax4.set_xlabel('Stage Number', fontsize=12, fontweight='bold')
ax4.set_ylabel('Ratio Value', fontsize=12, fontweight='bold')
ax4.set_title('Hydrogen Sulfide 2nd Dissociation:\nK₂s/[H⁺] = [S²⁻]/[HS⁻]', fontsize=14, fontweight='bold')
ax4.legend(fontsize=10, frameon=True)
ax4.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')

# CHECK 5: Water autoionization
ax5.plot(df_liquid['stage'], len(df_liquid['stage'])*[K_w_val], 'o-', label='Kw', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax5.plot(df_liquid['stage'], df_liquid['H']*df_liquid['OH'], 'o-', label='[H⁺] × [OH⁻]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax5.set_xlabel('Stage Number', fontsize=12, fontweight='bold')
ax5.set_ylabel('Product Value', fontsize=12, fontweight='bold')
ax5.set_title('Water Autoionization:\nKw = [H⁺] × [OH⁻]', fontsize=14, fontweight='bold')
ax5.legend(fontsize=10, frameon=True)
ax5.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')

# CHECK 6: CO2 hydration equilibrium
ax6.plot(df_liquid['stage'], len(df_liquid['stage'])*[K_eq_val], 'o-', label='Keq', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax6.plot(df_liquid['stage'], df_liquid['H2CO3']/df_liquid['CO2'], 'o-', label='[H₂CO₃]/[CO₂]', linewidth=3, markersize=6,
            markeredgecolor='black', markeredgewidth=0.8)
ax6.set_xlabel('Stage Number', fontsize=12, fontweight='bold')
ax6.set_ylabel('Ratio Value', fontsize=12, fontweight='bold')
ax6.set_title('CO₂ Hydration Equilibrium:\nKeq = [H₂CO₃]/[CO₂]', fontsize=14, fontweight='bold')
ax6.legend(fontsize=10, frameon=True)
ax6.grid(True, alpha=0.4, linestyle='--', linewidth=0.8, color='grey')

plt.tight_layout()
plt.subplots_adjust(top=0.94)
plt.show()
```

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--FINAL REACTIVE STRIPPING MODEL.htr

A5. Python Program for Operating Temperature Sensitivity Analysis

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--SENSITIVITY ANALYSIS - OPERATIN

A6. Python Program for Operating Pressure Sensitivity Analysis

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--SENSITIVITY ANALYSIS - OPERATIN

A7. Python Program for Stage Volume Sensitivity Analysis

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--SENSITIVITY ANALYSIS - STAGE VC

A8. Python Program for Outlet Gas Flowrate Sensitivity Analysis

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--SENSITIVITY ANALYSIS - OUTLET G

A9. Python Program for $NaHS$ Feed Concentration Sensitivity Analysis

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--SENSITIVITY ANALYSIS - NaHS FEE

A10. Python Program for H_2S Recovery Sensitivity Analysis

Viewable HTML file for this code (the HTML file for this code will also be submitted in a ZIP file along with this report for easy viewing):

--SENSITIVITY ANALYSIS - H2S RECC