Politecnico di Torino

Master's Degree Thesis

Master's Degree course in Energy Engineering

Modelling of photosynthetic membranes for solar fuel production

Supervisor
Prof. Eliodoro Chiavazzo
Co-supervisor
Gabriele Falciani

Candidate
Riccardo Ronco

A.A. 2021

Abstract

Photoelectrochemical reactions have been studied since the beginning of the 1970 for their promising ability of converting CO_2 to fuels and chemicals. The state of the art is focused on developing these reactions on solid electrodes which bring with them the issues linked to scarcity, instability and high cost. The European SoFiA project aims at using the unique self-assembling property of surfactants, and proton transport properties in soap films. The main goal is realizing an economical artificial photosynthetic membrane in form of stable soap film with engineered photocatalytic surfaces in order to produce a syngas, namely a mixture of carbon monoxide and oxygen. In this thesis a 1D COMSOL simulation of a surfactant monolayer, where the carbon dioxide half reaction occurs, has been carried out with a view on showing the feasibility of the project by simulating the behaviour of the main parameters over an 8-hour period. Furthermore a second 0D model was developed to study the oxygen evolution half reaction and validate against experimental data from literature.

Acknowledgements

A mio padre Sergio,
mia madre Cinzia e mia sorella Chiara.

Contents

List of Figures

List of Tables

List of symbols

IEA: Internation Energy Agency

PV: Photovoltaic

NZE: Net Zero emissions

APS: Announced Pledges Scenario

$STEPS$: Stated Policies Scenario

SDS: Sustainable Development Scenario

OEC: Oxygen evolving complex

$NADP$: Nicotinamide adenine dinucleotide phosphate

$NADPH$: Dihydronicotinamide-adenine dinucleotide phosphate

ATP: Adenosine Triphosphate

ADP:Adenosine diphosphate

$HOMO$: Highest occupied molecular orbital

$LUMO$: Lowest unoccupied molecular orbital

HER: Hydrogen evolution reaction

OER: Oxygen evolution reaction

NHE: Normal hydrogen electrode

PEC: Photoelectrochemical cells

ED: Electron donor

PS: Photosensitizer

TON: Turnover number

TOF: Turnover frequency

STF: Solar-to-fuel efficiency

FEM: Finite element method

$SoFiA$: Soap Film based Artificial Photosynthesis

CMC: Critical micelle concentration

GAC: Granular activated carbon

CAO: Oxidized activated carbon

CAR: Reduced activated carbon

CNT: Carbon nanotube

$SWCNT$: Single-walled carbon nanotube

$MWCNT$: Multi-walled carbon nanotube

CAT: Catechol

RES: Resorcinol

HYD: Hydroquinone

BFA: Bagasse fly ash

ACC: Commercial grade activated carbon

ACL: Laboratory grade activated carbon

BET: Brunauer-Emmett-Teller

NRW: Null reflecting water

GC: Gas chromatography

Chapter 1

Introduction

In this introductory chapter it is stated the importance of finding alternatives to fossil fuels furthermore some theoretical background for understanding the next chapters. Fossil fuels are the main energy source for both developed and developing countries. Now the risk related to the over-use of fuels with a strong environmental impact is well known and have been addressed by many studies under environmental, social and health points of view, but it is hard to shift from a fossil fuel based economy to a new one[1]. As stated in IEA (International Energy Agency) World Energy Outlook the world is trying to react to the problem, there are proofs that the future will be more electrified, interconnected and clean. Solar PV or wind now represents the cheapest available source of new electricity generation in most international markets, because of the increase of new investment in clean energy technologies. The cited report is divided into four different scenarios: the Net Zero Emissions by 2050 Scenario (NZE), the Announced Pledges Scenario (APS), the Stated Policies Scenario (STEPS), and the Sustainable Development Scenario (SDS), In Figure 1.1 the total global primary energy supply is shown by scenarios. Still in 2050 the world energy demand is strongly dependent on fossil fuels, even the APS Scenario foresees that more of the half of the energy needs will be covered by coil, oil and natural gas [2].

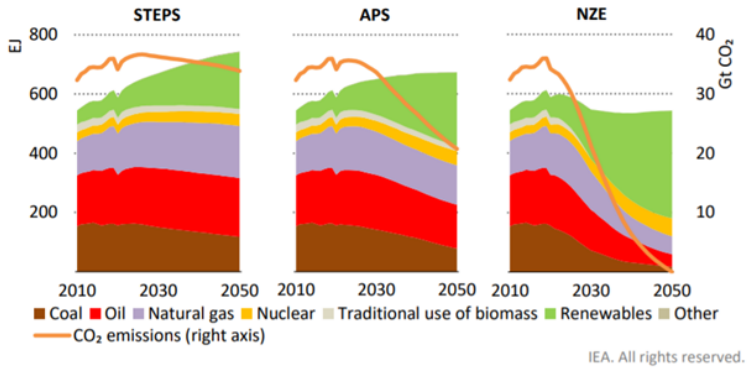

Figure 1.1: Total primary energy supply in different scenarios [2].

1.1 Solar fuels

Since there are many difficulties to react quickly and effectively to the environmental issues caused by the excessive use of energy sources of fossil origin and the change needs to take action now, an identified solution could be a wider usage of solar fuels which may reduce the high response times required to switch from a society and a global economy based on carbon fuels. A solar fuel is essentially the same of a carbon fossil which has not been mined from the ground but it is produced by chemical or physical process mainly starting from water or carbon dioxide. In Figure 1.2 it is

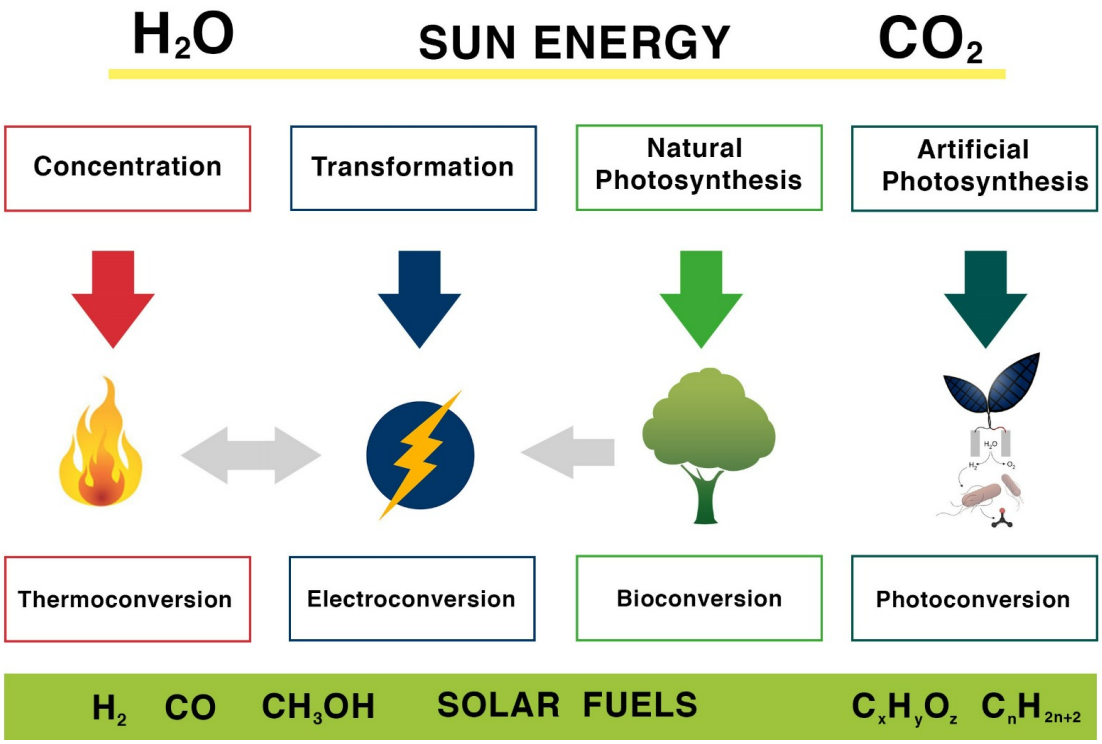

Figure 1.2: Scheme of possible pathways for solar fuels production.

possible to see the possible pathways to produce solar fuels. The first category deals with thermoconversion, solar light can be concentrated to generate high temperature heat in order to producuce solar fuels through thermal processes , it is reported that after a gas-water shift CO_2 can be reduced to produce a syngas which is a mixture of CO and H_2, subsequently the syngas can be utilized in a reactor to produce methanol [3]. Moreover the high temperature can be utilized to produce electricity, as it can be seen from the arrow in the scheme, then the electroconversion process can be utilized. This second category contains a wide range of applications, not only solar photovoltaics can be used but also technologies such as geothermal, wind and hydro power. The most famous process is electrolysis, basically it consists of a process where chemical reactions take place through the use of electricity. Water

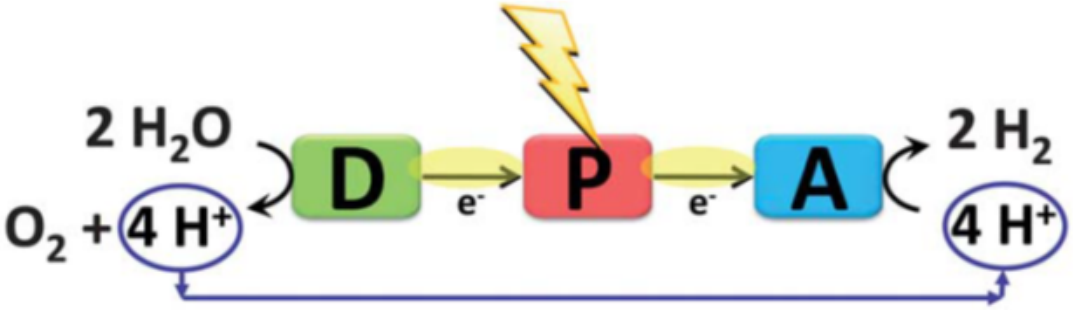

Figure 1.3: Principle reaction in artificial photosynthesis to make a solar fuel [8].

can be split via electrolysis to produce hydrogen which eventually can fed a fuel cell or it can be utilized as a energy storage of solar energy [4]. Moreover it is reported the possibility of realizing co-electrolys where H_2 and CO_2 are converted together into a syngas using reversible solid oxide cells [5]. The third category gathers all the natural photosynthesis pathways which includes algae, plants and trees that harvest solar energy to produce biomasss or biofuels, furthermore biological material may be utilized in a more direct way, indeed some micro-organisms can be genetically modified in such a way that they produce desired compounds by means of natural photosynthesis [6]. Eventually the last category that it is possible to see in Figure 1.2 deal with artificial photosynthesis which is a direct process that aims to mimic the natural photosynthesis to produce chemical compounds, namely solar fuels [7]. In Figure 1.3 a principle reaction in artificial photosynthesis to make a solar fuel (hydrogen) is depicted, P represents the molecule or the object which is in charge of absorbing the energy of the photon, A is the proton reducing catalyst while D is water oxidizing catalyst. These three steps: light absorption, molecules splitting (or reduction) and fuel generation are the basis of the photosynthesis, both natural and artificial [8].

1.1.1 Natural photosynthesis

Natural photosynthesis is a biological process where energy from sunlight is absorbed and stored to convert the pure energy of light into the free energy needed to power life. This process represents the basis of life on planet Earth, and it is responsible for most of our energy resources. The natural photosynthesis takes place in pigment-containing cells, in eukaryotic photosynthetic cells the process is located in subcellular structures known as chloroplasts, it contains chlorophyll pigments and in most of the organisms is the place where occurs the main phases of photosynthesis [9]. The process can be divided into four main stages as it is represented in Figure 1.4:

- **Light harvesting:** absorption of light by the antenna molecules (chlorophylls, Chl), that lead to electron-hole pairs(excitons) generations.

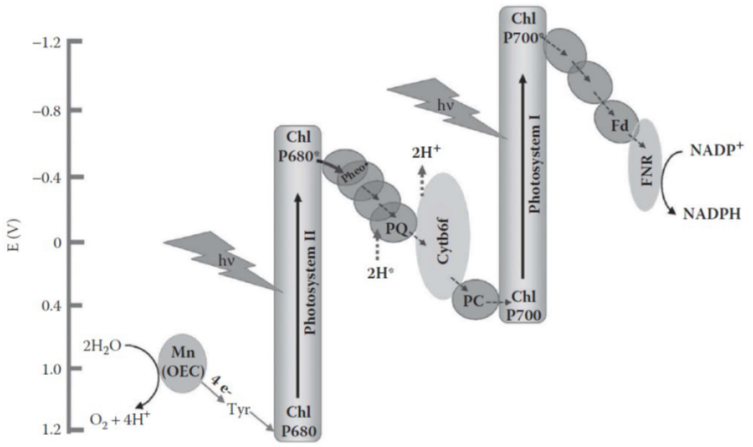

Figure 1.4: A simplified depiction of the molecular processes (light reactions) takes place in the biological photosynthesis [10].

- **Charge separation:** the charge separation of excitons and subsequent migration to appropriate reaction sites via various redox-active cofactors located in the photosystems II and I.
- **Water splitting:** the holes are used to split water in the OEC (Oxygen-evolving complex). Thanks to holes coming from P680, an enzyme with manganese-calcium core oxidizes two molecules of water releasing oxygen. Negative charges are passed to photosystem I, where sunlight is used to increase the energy of the electrons in order to drive the CO_2 fixation reactions.
- **Fuel Production:** electrons are then used in a chemical reaction to reduce NADP to NADPH that ultimately, with ATP and ADP produces carbohydrates such as glucose [10].

Natural photosynthesis process is particularly complex. Under ideal conditions: considering the absorption of all solar photons with $\lambda < 700$ nm, 8 protons for each CO_2 molecule fixed and assuming as the principal product the d-glucose it has been calculated a theoretical efficiency of 13.0% [11]. In practical for crop plants in both temperate and tropical zones efficiency typically does not exceed 1%. Higher 3% annual yields are reported for micro-algae grown in bioreactors [12].

1.1.2 Basic concepts

1.1.2.1 Photon absorption

Photon absorption is a process in which the photon loses its entire energy to an atomic electron which is in turn liberated from the atom or get promoted [14]. This process requires the incident photon to have an energy greater than the binding energy of an orbital electron, the energy of a photon is strongly dependent on its

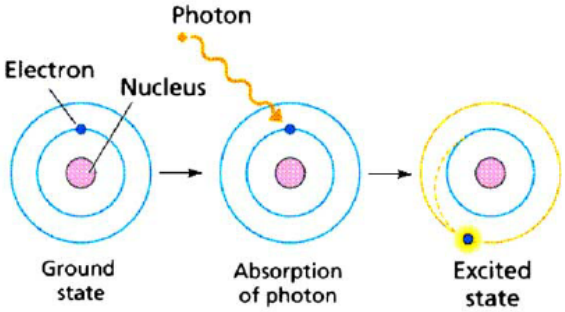

Figure 1.5: Excitation of electrons by photons [13].

wave length and directly proportional to the Planck's constant h which is equal to 6.626 x $10^{-34} Js$:

$$E_{ph} = h\nu \tag{1.1}$$

As a result of this phenomenon the electron may both leave the atom or it can be promoted to a more energetic orbital, in the second case after a certain period of time, determined by the relaxation process, the atom comes back to its initial energy level emitting energy (another photon). When atom have all its electrons in the lowest possible energy level, closest to the nucleus it is referred to be in the ground state while all the others are excited states. In Figure 1.5 it is possible

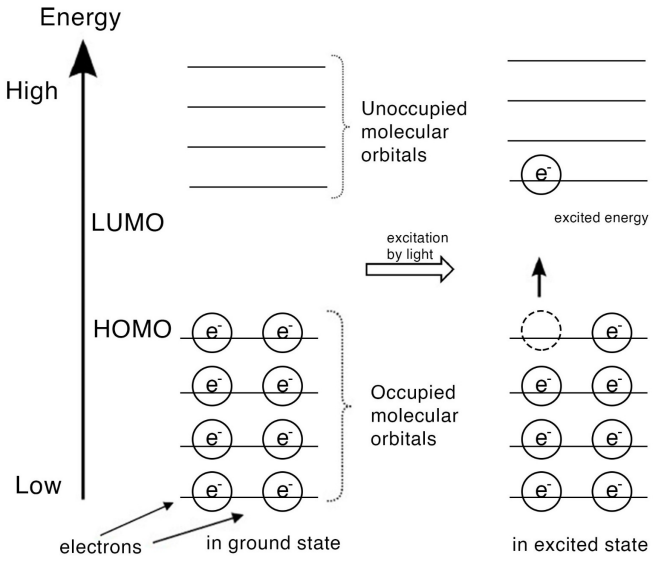

Figure 1.6: Schematic representation of HOMO and LUMO.

to see a representation of the process of excitation of an atom by a photon. As for the atom the same happen for the molecular structures, in fact they can not change their energy levels in a continuous way but only according to discrete energy levels called molecular orbitals. The highest occupied molecular orbital is defined

as HOMO,while lowest unoccupied molecular orbital as LUMO. It is possible for one or more electrons to pass from HOMO to LUMO thanks to photons which they have to hold enough energy to overcome the orbitals energy barriers. In Figure 1.6 it is possible to observe a schematic representation of the passage from less energetic orbital to another [13, 15].

1.1.2.2 Water splitting

The concept of water splitting together with carbon dioxide reduction, is the basis for solar fuel processes. It is a reaction which requires energy to split water in its constituents. There are several ways to obtain the water splitting, the conventional method is a water electrolyzer where the following reactions take place, the first one at the oxygen evolution catalysts with a redox potential of 0V vs NHE.

$$2\,H_2O \longrightarrow O_2 + 4\,H^+ + 4\,e^- \tag{1.2}$$

and the hydrogen evolution reaction:

$$4\,H^+ + 4\,e^- \longrightarrow 2\,H_2 \tag{1.3}$$

at its catalyst with a potential of 1.23V vs NHE; where NHE stands for "normal reference eletcrode" which is basically an electrode made of platinum in 1 M acid solution. In Figure 1.7 a scheme of a traditional water electrolyzer is depicted together with the reaction in acidic and alkaline solution, the two electrodes are divided by a proton exchange membrane in acidic condition or for the alkaline case by a diaphragm which are avoiding the contact between the two products in order to avoid an explosions [16].

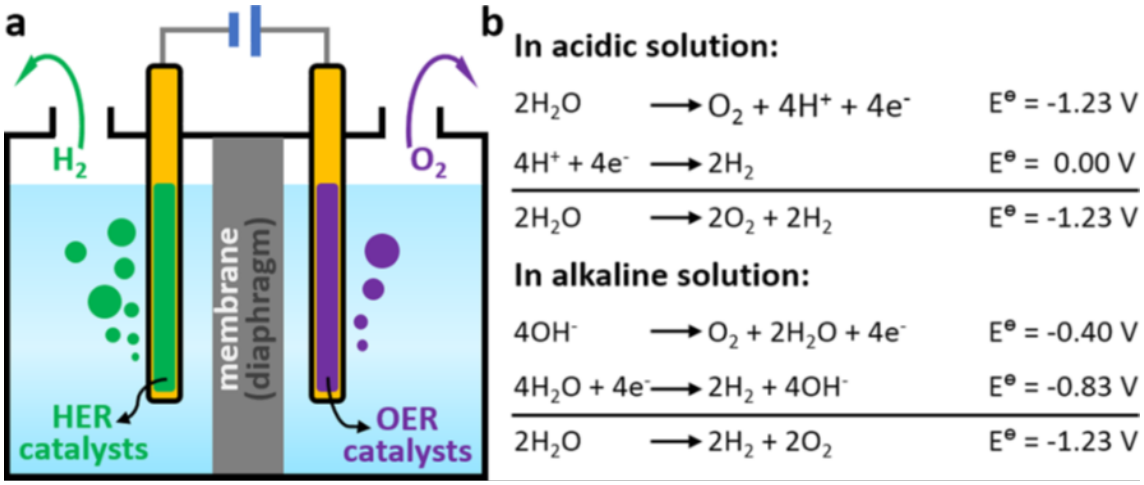

Figure 1.7: (a) Scheme of conventional water electrolyzers. (b) Water splitting reactions under acidic and alkaline conditions [16].

1.1.2.3 Carbon dioxide reduction

Carbon dioxide is a really stable molecule, this is why it requires high energy for bond breaking. Indeed the monoelectronic reduction of CO_2 occurs at a potential of about 2V, for this reason it would be advantageous for artificial photosynthesis to perform directly polyelectronic transformations delineated in the following equations:

$$\mathrm{CO_2 + e^- \longleftrightarrow CO_2^-} \qquad [\mathrm{V = 1.9\,V}] \tag{1.4}$$

$$\mathrm{CO_2 + 2\,H^+ + 2\,e^- \longleftrightarrow HCOOH} \qquad [\mathrm{V = 0.61\,V}] \tag{1.5}$$

$$\mathrm{CO_2 + 2\,H^+ + 2\,e^- \longleftrightarrow CO + H_2O} \qquad [\mathrm{V = 0.51\,V}] \tag{1.6}$$

$$\mathrm{CO_2 + 4\,H^+ + 4\,e^- \longleftrightarrow HCHO + H_2O} \qquad [\mathrm{V = 0.48\,V}] \tag{1.7}$$

$$\mathrm{CO_2 + 6\,H^+ + 6\,e^- \longleftrightarrow CH_3OH + H_2O} \qquad [\mathrm{V = 0.38\,V}] \tag{1.8}$$

$$\mathrm{CO_2 + 8\,H^+ + 8\,e^- \longleftrightarrow CH_4 + H_2O} \qquad [\mathrm{V = 0.24\,V}] \tag{1.9}$$

Recent studies are striving to reproduce a multiple electron transfer in one step while according to the experimental results it is reported that the process on semiconductors start with one electron only which excites the CO_2 making the first step the bottleneck of the whole reaction [17, 10].

1.1.2.4 Semiconductor materials

All the solid state materials can be grouped according to their conductive properties in three big categories: insulators, semiconductors and conductors. These subdivision comes from their difference in two energy bands called valence band and conduction band. The valence band is the highest range of electron energies in which electrons are normally present, while the conduction band is the lowest range of vacant electronic states [18], in Figure 1.8 it is possible to see the differences between these bands together with the concept of LUMO and HOMO. Insulators show a big band gap this is why they are not good materials to conduct electricity, in conductors this energy gap can be mixed and sometimes there is not a clear division between the two, their values are overlapping, while semiconductors display a band difference but it is not as much as big like insulating materials, the band gap of silicon which is a "famous" semiconductor is 1.12 eV while the diamond, that is an insulator, reports a value of 5.5 eV. In particular semiconductors are of interest because they have a suitable band gap for the solar fuel applications, indeed photons can effectively move electrons from the valence band to the conduction one leaving holes. This duet called electron-hole pair is the basis of the working principle of semiconductive materials and it is a reversible process, after an electron is excited across the band gap (carrier generation) eventually the conduction band electron goes back to occupy the energy state of an electron hole in the valence bad, this process is known as charge recombination [19]. Moreover the conduction can be subdivided depending on the presence of impurities: the p-type materials and the n-type materials. Indeed the semiconductors valence electrons can be doped

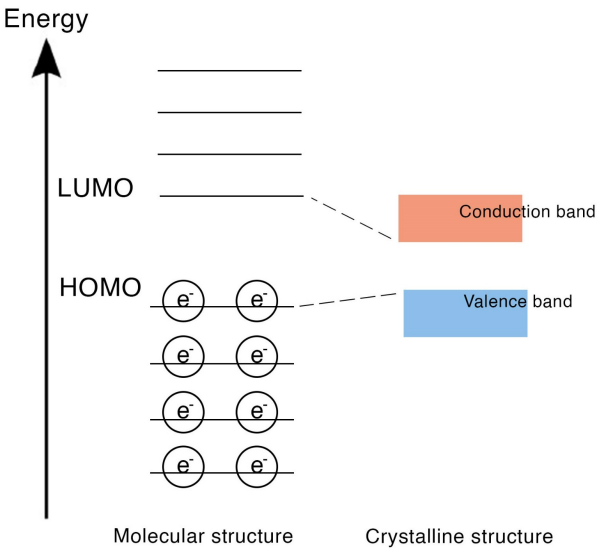

Figure 1.8: Valence and conduction band of semiconductors.

to produce an imbalanced number electrons or holes, in the case a semiconductor presents an excess number of hole it is called a p-type, on the other hand a n-type is characterised by a larger number or free electrons [20]. Eventually in Figure 1.9 it is possible to see the band structure of the most employed semiconductors for water splitting and carbon dioxide reduction regarding solar fuels reactions.

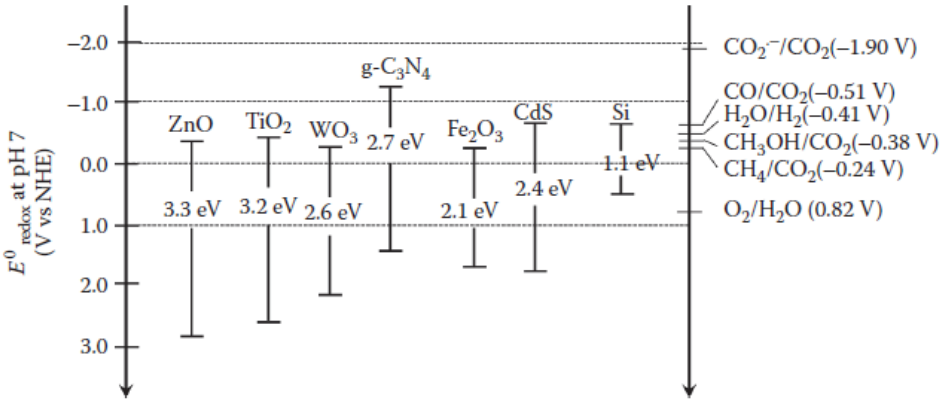

Figure 1.9: Band gaps, positions of Econduction and Evalence of various popular semiconductor photocatalysts, and the potentials of the redox couples participating in fuel generation reactions [10].

1.2 Artificial photosynthesis

Photosynthesis is the formation of new compounds thanks to photon driven reactions, the basis of photosynthesis is photoinduced electron transfer, in this process light is absorbed by a chromophore to produce an excited state. Artificial photosynthesis systems, mimicking the natural one, use organic or metal-organic materials to harvest sunlight and convert it to electrochemical potential with a view to fuel production in a more efficient and comparted way [21]. In the field of artificial photosynthesis it may be useful to distinguish the different technologies between molecular and non-molecular processes.

1.2.1 Non-molecular processes

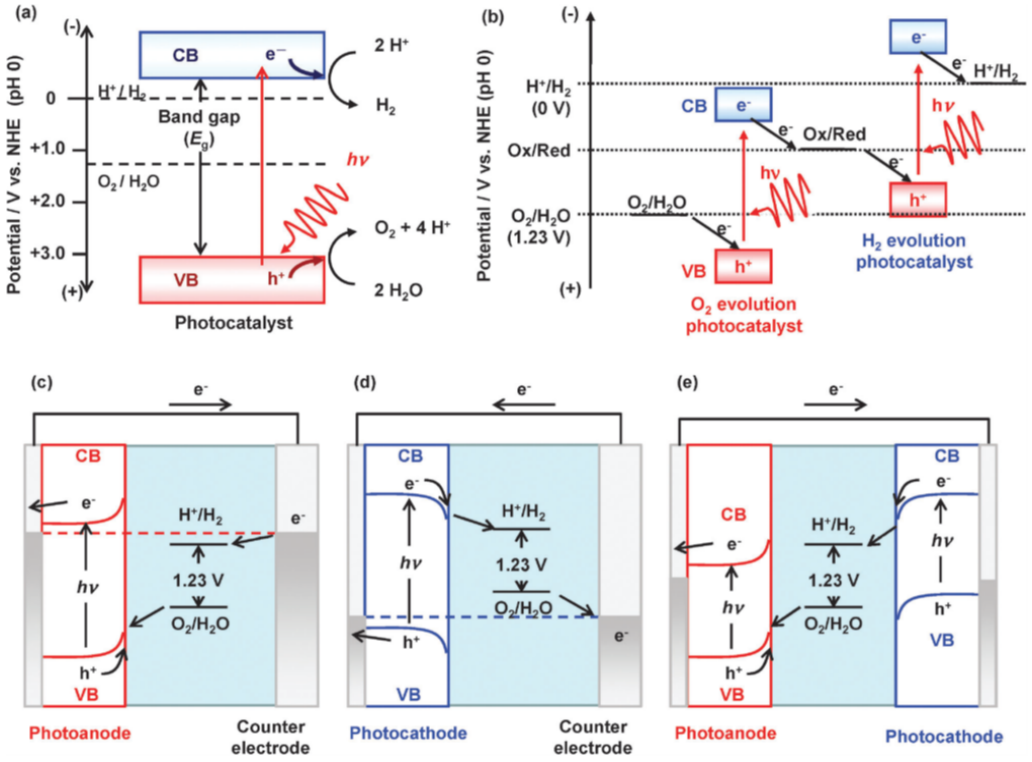

Figure 1.10: Energy diagrams of photocatalytic water splitting based on (a) one-step excitation and (b) two-step excitation; and PEC water splitting using (c) a photoanode, (d) photocathode, and (e) photoanode and photocathode in tandem configuration. The band gaps are depicted smaller in (b) and (e) to emphasize that semiconductors with a narrow band gap can be employed. [22].

Non-molecular systems are based on light driven catalyst and materials for photon capture which are not molecules, the photocatalytic processes take place on metal surface, matal-oxide, semiconductors, nano-structured or carbon based materials. The photoabsorber should have a fitting band-gap in order to be able to utilize

the widest possible amount of solar flux, moreover it has to be made from non-toxic and abundant materials [8, 23]. One of the most known technological solutions between non-molecules processes are photoelectrochemical cells (PEC), in Figure 1.10 an example for PEC photocatalytic water splitting is given for semiconductors material. The reaction can take place through a one step excitation process as shown or in two steps as shown in the energy diagrams in fig 1.10a, fig 1.10b, respectively. In the first case a n-type semiconductors is employed, it is important to notice that the top level of the valence band has to be larger than the than the oxygen evolution potential to allow a photoanode to generate oxygen Fig 1.10c. On the other hand, a p-type semiconductor works as a photocathode for hydrogen evolution when the conduction band edge is more negative than the hydrogen evolution potential as shown in Fig 1.10d. Eventually another possible option is to connect in tandem a photoanode and a photocathode with a two-step excitation [22, 24].

Another example involving semiconductors is the photocatalytic reduction of carbon dioxide on a semiconductor. The process is shown in Figure 1.11 in this case reduction and oxidation cocatalyst are utilized, using additional catalyst help the system which are useful because they are able to lower the activation potential for CO_2 reduction which is as said relatively high, furthermore they can improve the stability of semiconductor photocatalyst by consuming the photoexcited electrons and the holes and they can increase the selectivity of CO_2 reduction toward specific molecules [25].

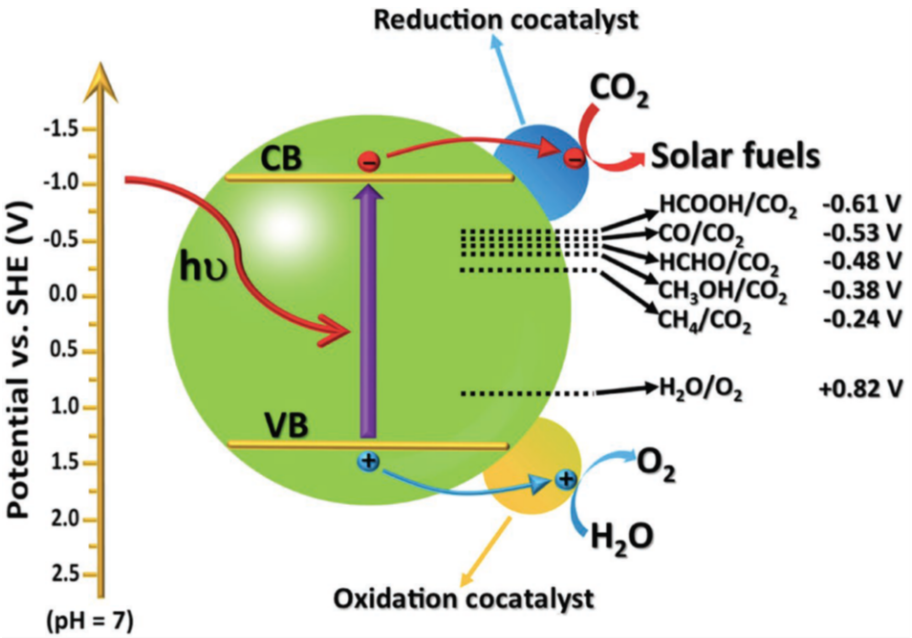

Figure 1.11: Schematic illustration of photocatalytic CO_2 reduction on a semiconductor photocatalyst coloaded with reduction and oxidation cocatalysts for solar fuel production [25].

1.2.2 Molecular processes

On the other hand in molecular processed the essential element is that the catalysts are molecules, this is why these systems are often referred as homogeneous. In Figure 1.12 a schematic representation of light-driven hydrogen production mechanism thanks to a molecular process, the same principles described in the non-molecular processes are the basis of this mechanism: the photon absorption leads to a separation of electron-hole pairs which initiate the photocatalytic process. The photons excites a molecule ([PS] in the figure) which provides electrons to the [2Fe2S] catalyst that is responsible for driving proton reduction. An electron donor ED is present, it is important that these compounds are easily oxidizable in order to ensure rapid recovery of the PS before charge recombination takes place. In particular in this case the process should be repeated twice in order to accumulate two electrons in the catalyst and effectively reduce two proton to produce a H_2 molecule. The light absorber element referred as [PS] in the described example of a molecular photocatalytic hydrogen production is a photosensitizer the catalyst were made of Fe but they can be produced using many different materials such as cobalt, iron, manganese or nickel for molecular applications [26]. A photosensitizer absorb elec-

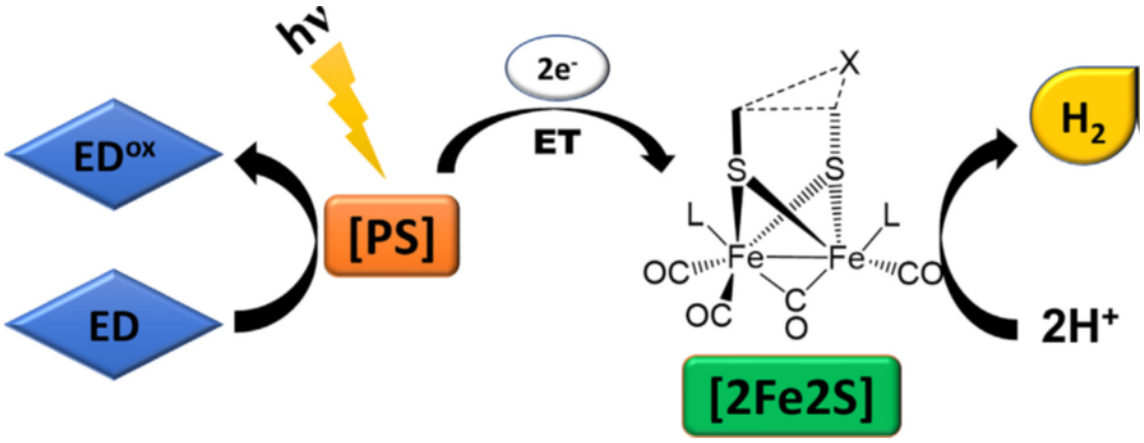

Figure 1.12: Schematic representation of a molecular light-driven hydrogen production mechanism. ED: electron donor, PS: photosensitizer and [2Fe2S] [26].

tromagnetic radiation consisting of infrared radiation, visible light radiation, and ultraviolet radiation and transfer absorbed energy into neighboring molecules. It is a material which has to be able to absorb light to give an excited state, this state must be able to oxidize or reduce the neighbouring molecules. Practically a light absorption sensitizer is a chemical species that ensure the accomplishment of photochemical reactions which can not take place if the reactants are not able to absorb light [27]. Many photosensitizers are organic or organo-metallic compounds, one the most known and studied is the metal-based $[Ru(bpy)_3]^{2+}$- type complexes (Figure 1.13). They are appreciated because their excited state is relatively long, allowing them to participate both in a single-electron reduction or oxidation avoiding charge recombination, they show a good stability respect to another sensitizers

and they are compatible with a wide range of pH. Moreover they display a broad absorption of visible light and their photophysical properties can be modified in order to extend their absorption capabilities from the infrared to the UV region [28].

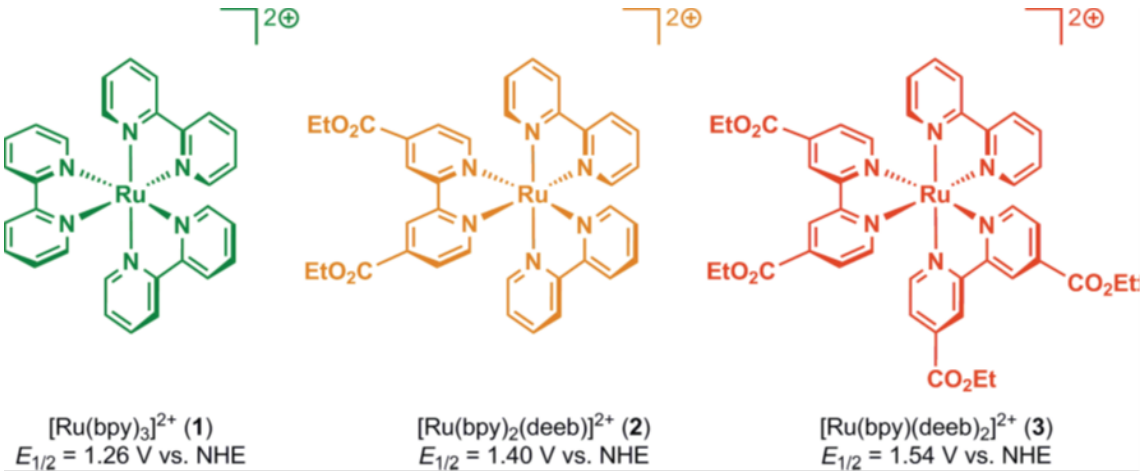

Figure 1.13: Examples of $[Ru(bpy)_3]^{2+}$-type photosensitizers [28].

The advantages of molecular systems compared to non-molecular ones is that it is easier to study the mechanism of the reaction involved and the catalyst can be made from inexpensive and abundant materials. On the other hand there are more studies available on non-molecular processes and they are more stable and more robust against degradation. A parameter which may be useful to compare different photocatalytic system is the Turnover number (TON) which can be calculated is the molecules produced per molecule of catalys divided by the catalyst's lifetime. Another parameter is the turnover frequency (TOF) which is a measure of the efficiency of a catalyst, calculated as the derivative of the number of turnovers of the catalytic cycle with respect of the time per active site [8, 10, 29].

1.2.3 Lab-scale devices

The typical lab-scale devices for artificial photosynthesis are:

- Photoelectrochemical cells (PEC);
- Particle-based devices;
- Homogeneous catalysis;
- Self-assembled membranes.

The first example of PEC which was producing a fuel thanks to photochemical splitting of water was reported by Fujishima and Honda who built the first artificial leaf in 1972. As shown in Figure 1.14 the system is composed of two electrodes, one

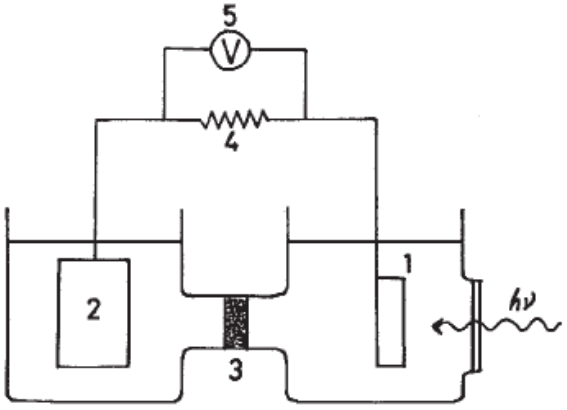

Figure 1.14: Electrochemical cell in which the TiO_2 electrode is connected with a platinum electrode. The surface area of the platinum black electrone used was approximately $30cm^2$ [30].

made of titanium (1), the other of platinum(2), the leaf is separated by a proton conducting membrane(3), in aqueous electrolyte, and connected by an external circuit(4). When the TiO_2 electrode is irradiated the electron flows to the Pt electrode, the negative charge movement defines the TiO_2 as anode where oxygen evolution occurs while at the cathode the reduction (hydrogen evolution). Even though the STF (solar-to-fuel efficiency) is very low since only few photons have enough energy to move the electrons [30]. The problem is that materials such as platinum which shows a good band gap,generally are not cheap elements so there are difficulties to make the project commercially feasible. Over the years the research tried to develop PEC using available and inexpensive materials. In 2011, Reece and coworkers developed a solar water splitting system made of a commercial triple junction amorphous silicon while the catalysts were made from an alloy of earth-abundant metals and a cobalt catalyst. The PEC was configured in two ways how it is displayed in Figure 1.15, wired and wireless, in the wired cell the alloy which is the H_2 catalyst, was deposited on an almost transparent Ni mesh substrate which was wired to triple junction while in the wireless one the alloy was deposited directly on the opposing of the junction. The wired configuration displayed an higher solar-to-fuel efficiency (4.7%) but after 1 hour it showed a rapid decline in activity while the wireless configuration demonstrated a fewer efficiency (2.5%) but the cell remained stable for 10 hours and its performance gradually declined to 80% of its initial value after 24 hours [31].

More recently in 2018 Cheng and coworkers reached an efficiency of 19.3% and 18.5% in acidic and neutral electrolytes, respectively. In this case they used a sandwich of different materials with a tailored multifunctional crystalline titania interphase layer who was preventing from corrosion and facilitating the electron transfer [32].

For what concern particle-based artificial photosynthesis, as the others photo-

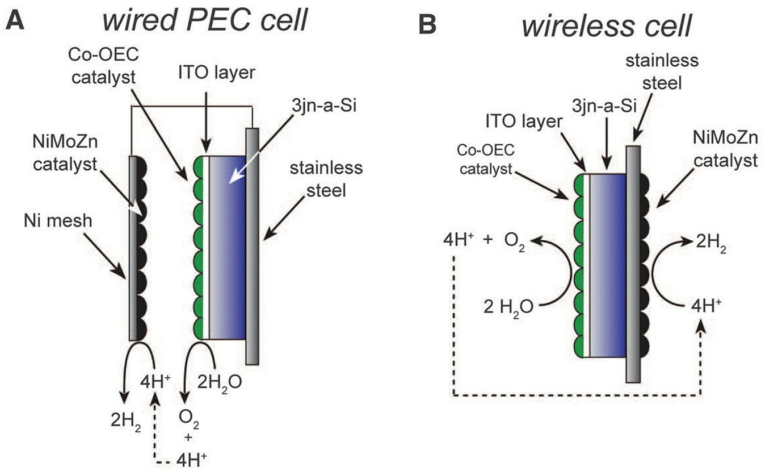

Figure 1.15: PEC wired and wireless systems [31].

catalytic processes, also particle-based devices one can ben subdivided in four parts with the difference that it takes places on particles level. Some kinds of reactors utilize particle suspensions where the particles are free to move, and they are not incorporated as part of membrane to separate the redox reaction. In this way both oxidation and reduction occur on each light absorber particle. Fabian and coworkers described two kinds of reactors, the type 1 evolves H_2 and O_2 in the same compartment while the type 2 in separate vessels as depicted in Figure 1.16 [33]. The next

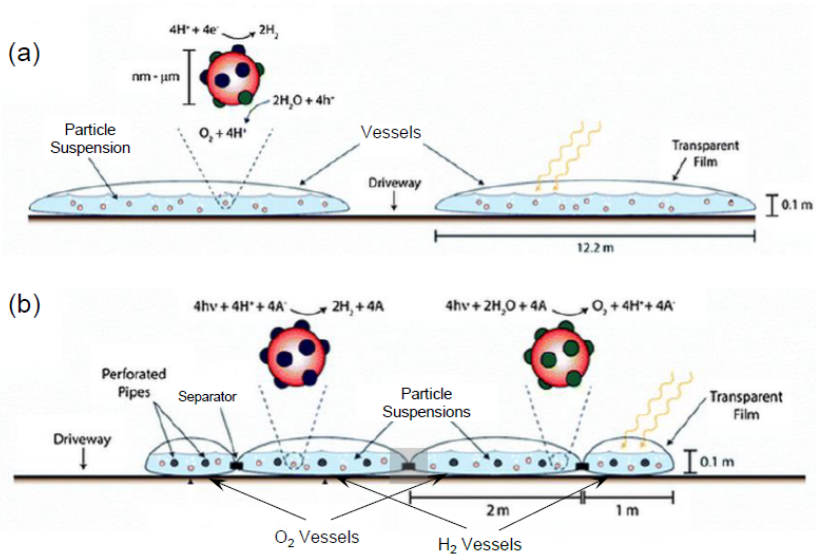

Figure 1.16: Two general designs for (a) a Type 1 reactor and (b) a Type 2 reactor [33].

on the list is homogeneous photosynthesis which is typically an aqueous solution containing a electron donor, a photosensitizer or a dye to collect photons, the cat-

alysts to lower the activation energy and a buffer. The research is at the moment focused on developing stable materials capable to last in time and lowering the effect of decompositions in the materials, this is why it not given a specific solar-to-fuel effiency. Recently it is reported that in 2019 Zhang and coworkers demonstrated that heteroleptic copper(I) displayed a promising performance as a water-soluble and earth-abundant photosensitizer for the CO_2 to CO photoconversion [34].

In the end self-assembled membranes are the one who recalls more the natural photosynthesis, Stikacane and coworkers developed a biomimicking system because the organic membranes result really efficient in the charge separation process, in this study electron transfer across the lipid bilayer is ensured via the transmembrane protein complex MtrCAB but there is still no fuel production [35].

1.3 Continuum modeling

This thesis work is linked to the EU funded FET OPEN H2020 SoFiA project "Soap Film Based Artificial Photosynthesis" which will be better described in the next section. The aim of this thesis work is to develop a computer simulation which is able to reproduce real phenomena using a commercial software called COMSOL Multiphysics . The cited software exploits the theory of continuum modeling and, in particular, the finite element method to give solutions to those problems which can not be solved analytically. Generally speaking all the finite element methods can be subdivided in steps, the first one is the finite element discretization where the considered domain is divided in a finite number n of subdomains (or elements). The elements are connected to each other at points called nodes. The second step is to define the set of equations that rule the elements, afterwards is to find a way to assembly the element equations, this part is often taken care of by numerical models. Eventually it is important to estimate the errors associated to the method checking if the final solution tend to converge to the problem solution [36, 37]. In Figure 1.17 a representation of the nodes and elements for a biological tissue is given.

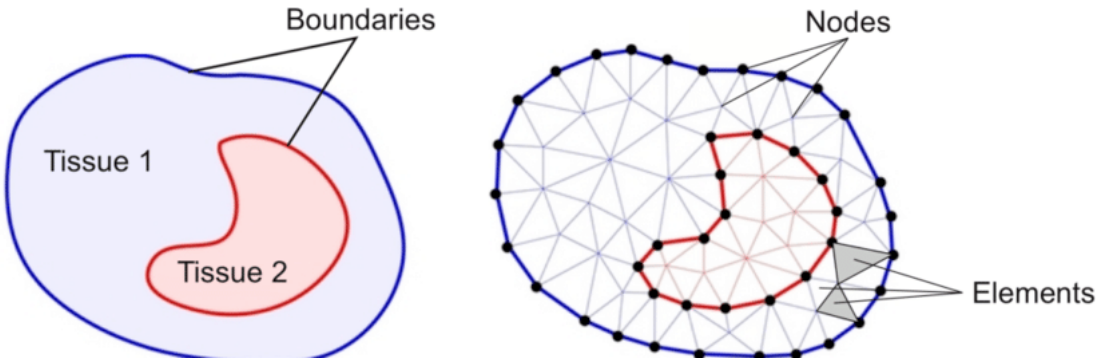

Figure 1.17: Schematic representation of a finite element method (FEM) model [38].

1.4 SoFiA project

Since all solid state devices suffer from wear and decomposition issues, moreover the bubbles on metal electrodes can slow the reactions and molecular processes bring the problem of instability, the idea is to perform artificial photosynthesis on liquid membranes. The project proposes an innovative approach to the solar fuel production thorough the utilization of surfactants and proton transport properties in soap films like the one depicted in Figure 1.18. SoFiA aims to reproduce an artificial photosynthesis device in form of stable soap film which is affordable and can incorporate state of the art catalysts. The project starts with the goal of CO_2 reduction thanks to sunlight, but it can be modified to produce H_2 in the future [39].

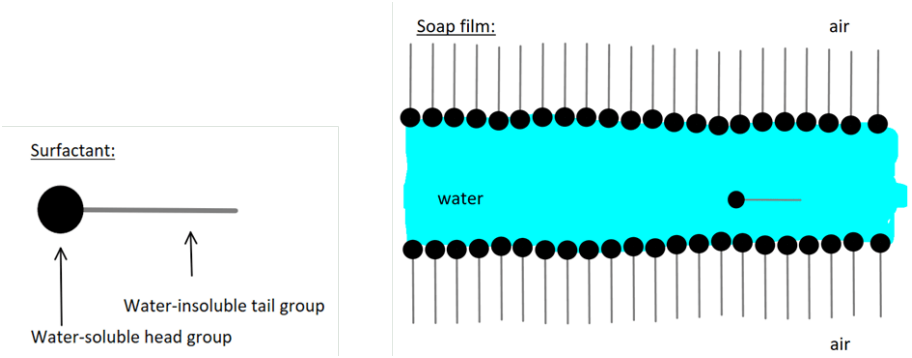

Figure 1.18: Soap film scheme and surfactant representation [39].

1.4.1 Soap films and surfactant monolayers

As hinted the main goal is to realize a photocatalytic system in a soap film which is the basic structural unit of foams and act as cell walls encapsulating the gas with a thickness of around 1 μm. Before describing what soap films and surfactant monolayers are same concepts are needed. Dealing with liquid surfaces one of the most relevant parameter is the surface tension, the concept is shown in Figure 1.19, the amount of work needed to increase the surface is proportional to the increase itself, the proportional constant is equal to the surface tension which is calculated in N/m. From a molecular point of view is energetically favourable to be surrounded by other molecules thus the molecules attract each other making an opposition respect to the surface modification with a result that the surface tension tends to minimize the area. The surface tension is an interfacial property and it can be reduced by surfactants, molecules which present an amphiphilic behavior, in fact they consist of two parts an hydrophilic hydroxyl group (the head) and an hydrophobic hydrocarbon assembly. It is common to classify surfactants in aqueous media according to the

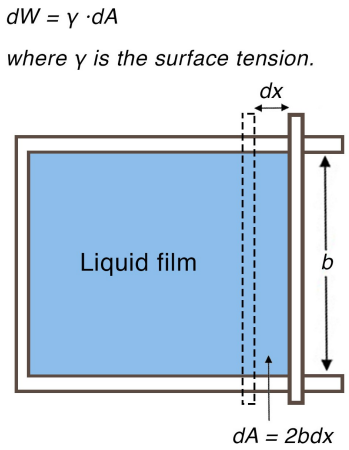

Figure 1.19: Schematic set-up to define the surface tension.

nature of their hydrophilic functional groups:

- Anionic surfactants;
- Cationic surfactants;
- Nonionic surfactants;
- Amphoteric and Zwitterionic surfactants.

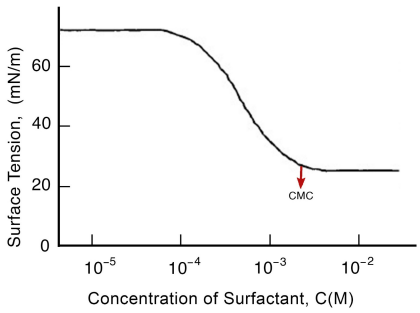

Figure 1.20: Surface tension variation of a typical aqueous surfactant solution.

Anionic ones have a negative charge while cationic surfactants a positive one, on the other hand nonionic are not charged and amphoteric and zwitterionic carry both a positive and a negative charge with the result that the net charge is zero. In aqueous solutions surfactants aggregate forming structures called micelles, in Figure 1.20 it is possible to see the surface tension variation of a typical aqueous surfactant solution,

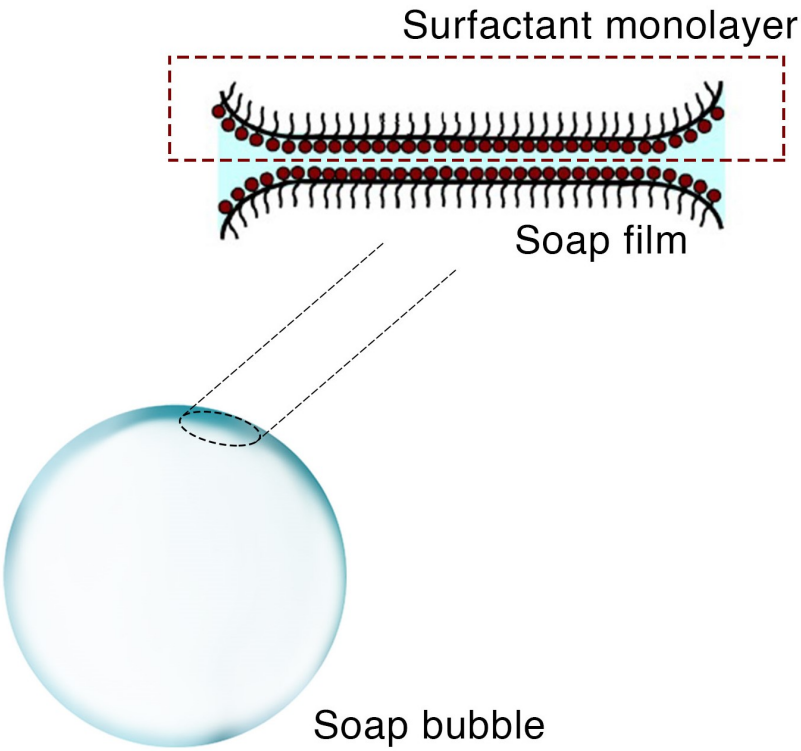

Figure 1.21: Graphical presentation of a bubble, a soap film with the surfactant monolayer.

at a certain point it is possible to spot a plateau corresponding to a concentration value called critical micelle concentration (CMC).

In chemistry a dispersion of particles in a continuous medium where the particles are gas bubbles and the medium is a liquid, which forms a collection of thin liquid films is called foam. Liquid foam are instable but they may last longer thanks to stabilizers such as soaps, detergents and proteins. In the case it is stabilized by surfactants it may be referred as soap film, indeed the surfactant not only reduces the surface tension but also causes a repulsive force between two parallel gas-liquid interfaces which is an interaction that stabilizes the film reducing the disjoining effect [40]. In the case the amphiphiles molecules does not dissolve in the liquid the formation of a compact monolayer film occur known as surfactant monolayer. In Figure 1.21 the different definitions of soap film and surfactant monolayer is depicted, simplifying it may be said that a surfactant monolayer is basically half of a soap film [41, 42, 43].

The process of monolayer film formation at the air-water interface has been studied for a long time. Langmuir in the early 1917 found that the films can be

spread on solid surfaces. The monolayer films recall the assemblies of biological membranes and many reactions may be realized within. Yet in 1979 there were researches reporting electron and donors acceptors embedding redox processes in the presence of $[Ru(bpy)_3]^{2+}$ showing that thanks to the special properties of molecular organization and packing of soap films the reactivity was way higher compared to micelles and microemulsions [44]. The perk of deploying soap films lays on the fact that they can be continuously regenerated and there are proof that they are easily tunable, indeed it is not complicated to realize thin layers of water in the form of soap film, furthermore their nature and their thickness can be varied [45]. For example in the research of Mamane and coworkers a liquid thin film is stabilized by a cation photosurfactant which is able to switch the hydrophobic tail from a trans to a cis conformation through different value of illumination causing a modification in the soap film [46].

1.5 Theory

In this part an overview over physics and solved equations by COMSOL is reported. The topic covered are diffusion according to the Fick's law, the thermodynamics of interfaces, looking for appropriate formulas for gas-liquid surfaces and their adsorption properties. Moreover it is useful to understand how the software deals with the chemical equilibrium and eventually the Henry's law is outlined to understand which input data is given to the simulations

1.5.1 Fick's laws of diffusion

The theoretical model to describe diffusion is the Fick's one, the diffusion can both take place across the surfactant monolayer and in the atmosphere, at the same time both gas and liquid molecules are diffusing in the region individuated by the monolayer. Fick's law of diffusion describes how particles under random thermal motion tend to spread from a region of higher concentration to a region of lower concentration [47]. The first Fick's law for diffusion is:

$$J = -D\nabla c \tag{1.10}$$

where ∇c is the concentration spacial gradient, the above formula can be simplified for 1D geometries considering the x direction only:

$$J = -D\frac{dc}{dx} \tag{1.11}$$

Where J is the flux and represents the number of moles that are flowing through a unit area in the time unit, D is the diffusion coefficient or diffusivity, and c is the concentration of the species considered. The negative sign indicates that the concentration gradient is negative. Intuitively it means that the particles tend to move from a more concentrated domain to a less one. This first law can be utilized to study steady phenomena because it does not take into account the concept of time, for this purpose it can be useful to explicit the second Fick's law of diffusion that comes from the first one:

$$\frac{dc}{dt} = D\frac{d^2c}{dx^2} \tag{1.12}$$

where the derivative respect to time is the rate of change in a certain control area while the second degree space derivative represents the changes that the change in concentration can take [48].

1.5.2 Thermodynamics of interfaces

The surfactant modify the liquid surface positioning itself at the gas-liquid interface. One of the ruling parameters in the thermodynamics of interfaces is the surface

excess which defines something like a surface concentration as:

$$\Gamma_i = \frac{N_i}{A} \tag{1.13}$$

where A is the interfacial area and N the number of moles another important parameter is the Gibbs free energy of interface which is equal to the following equation assuming the the interface is flat (planar):

$$dG = -SdT + VdP + \sum_i \mu_i dN_i + \gamma dA \tag{1.14}$$

The given equation comes from the first and second principle of thermodynamic with the chemical potential energy of the species and the amount of energy linked with the surface tension. Its value is possible to be derived with Gibbs free energy as:

$$\frac{\delta G}{\delta A}|_{T,P,N} = \gamma \tag{1.15}$$

Indeed the surface tension is the increase in the Gibbs free energy per increase in surface area at constant pressure, temperature and amount of molecules involved. The concept of Gibbs free energy is not only useful when talking about the surface tension but it can also be employed to describe thorugh the Gibbs adsorption isotherm, the change of the interfacial properties when molecules like surfactant are present at the surface. The isotherm function is derived from the interfacial energy, for the purposes of this thesis it is enough to know that at constant it follows:

$$d\gamma = -\sum \Gamma d\mu \tag{1.16}$$

In particular for a system of a two components it gives:

$$d\gamma = -\Gamma_1 d\mu_1 - \Gamma_2 d\mu_2 \tag{1.17}$$

with an appropriate decision of the interface it comes that $\Gamma_1 = 0$, subsequently it is possible to develop the chemical potential as:

$$\mu_2 = RT \ln\left(\frac{a}{a_0}\right) \tag{1.18}$$

Where a is the activity and a_0 the standard activity ($1\ mol/L$) differentiating this last equation a substituting into the previous one it comes:

$$\Gamma_2 = -\frac{a}{RT}\frac{\delta\gamma}{\delta a} \tag{1.19}$$

[49]. This is an important result because it directly tells that the surface tension decreases when the solution concentration increases. In particular it is useful to derive it in the case where gases are adsorbing to liquid surfaces, in this situation

the activity is directly proportional to the partial pressure of the adsorbing gas thus the equation can be written as:

$$\Gamma = -\frac{1}{RT}\frac{d\gamma}{d\ln P} \tag{1.20}$$

1.5.3 Langmuir adsorption model

Gas molecules adsorb on the surfactant monolayer, after that the CO_2 reduction takes place thanks to the electron supplied by the electron-relays. Here it is given a brief description of the adsorption phenomena and the Langmuir model is outlined. The adsorption is the accumulation of a substance at an interface, in the studied

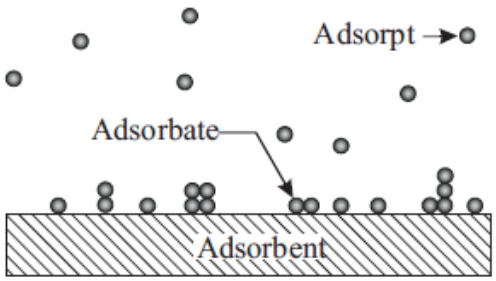

Figure 1.22: Definitions of adsorbent,adsorpt, and adsorbate [50].

case the adsorption occurs at the liquid-gas interface thanks to intermolecular forces. The molecule can desorb from the surface, the rates of adsorption and desorption define the amount on the surface at equilibrium. By definition what is found at the adsorbed state is called adsorbate, the material to be adsorbed is defined as adsorpt or adsorptive while the substance where adsorption takes place is called adsorbent. In figure 1.22 a graphical representation is given. The amount of adsorbate is generally depending on the pression and temperature [50].

Experimentally graphs called adsorption isotherms can be outlined, on the y axis there is the amount of material adsorbed while the abscissa is the partial pressure or, for adsorption from solution, the concentration is used. Langmuir model presents some assumptions, first of all adsorption does not takes place everywhere but only in localized sites, each site can accommodate only one atom or molecule, there are no phase transitions, the surface is energetically homogeneous without interactions between neighbouring adsorbed molecules and the process is reversible [51]. Furthermore, Langmuir assumes that at equilibrium adsorption and desorption are equal at equilibrium:

$$k_a C_e(1-\theta) = k_d\theta \tag{1.21}$$

where:

- k_a is the adsorption rate;
- k_d is the desorption rate;
- θ is the number of occupied sites divided the maximum number of sites, it represents the coverage percentage;
- C_e is the equilibrium concentration of adsorbate in the bulk region.

Making θ explicit, which is the most common form of this equation:

$$\theta = \frac{\frac{ka}{kd} C_e}{1 + \frac{ka}{kd} C_e} \tag{1.22}$$

Moreover θ can also be written in this form [52].

$$\theta = \frac{c_s}{\Gamma_s} \tag{1.23}$$

Where c_s is the species surface concentrations and Γ_s is the surface excess at saturated coverage. The surface excess may also be used in adsorption isotherms and it

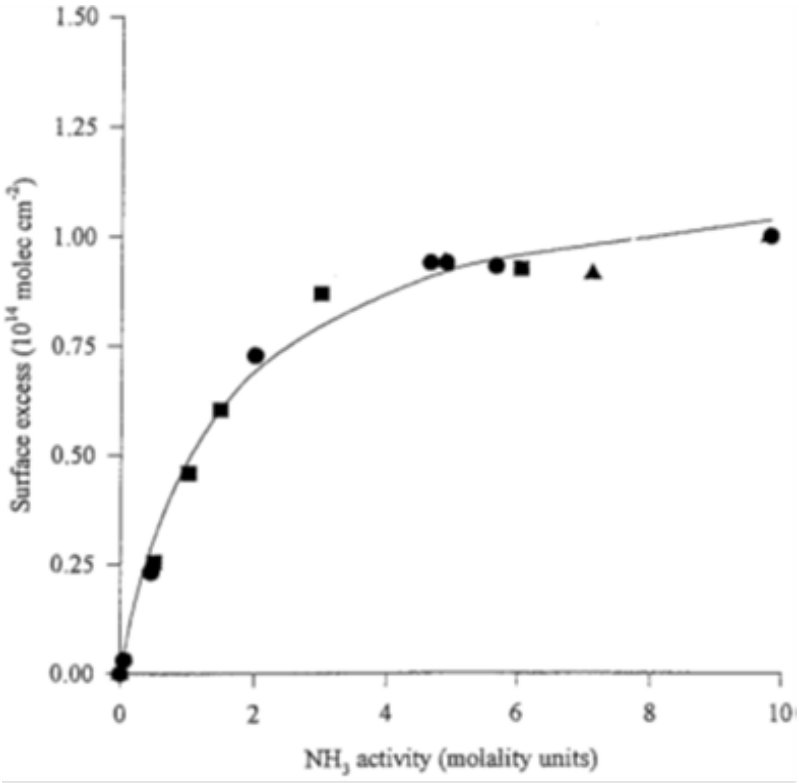

Figure 1.23: Ammonia surface excess at 298 K as a function of ammonia solution activity. The line shows a fit of the data to a Langmuir adsorption isotherm [53].

can be linked using Gibbs energy, in Figure 1.23 an example of Langmuir isotherm

fit to experimental data is given for a temperature of 298 K. In the Donaldson's work ammonia and other relevant atmospheric gases are studied when they adsorb at the air-water interface. The solid line represents the Langmuir isotherm while the points are experimental data, on the x axis there is the ammonia activity which can be exchanged with concentration for ideal solutions [53].

1.5.4 Chemical equilibrium

A chemical equilibrium is a state where reactants and products find their balance in a chemical reaction with the result that there is no observable change in the properties of the system. The equilibrium is stable, but the chemical species are in a continuum movement, the natural of chemical equilibriums is dynamic. This balance is not passive but active, in the presence of a perturbation the reactions system is able to react in a way to try to relieve the perturbation [54].

The modelling reactions of COMSOL is based on the mass action law. The law's concept is that two particles must collide to enter into the reaction, the probability of the collision is proportional to the product of their concentrations so the reaction rate must be proportional to the product of concentration of reacting substances [55]. Considering a general reaction:

$$\mathrm{aA} + \mathrm{bB} + ... \underset{\mathrm{k_j}^r}{\overset{\mathrm{k_j}^f}{\leftrightarrows}} \mathrm{xX} + \mathrm{yY} + ... \tag{1.24}$$

For this reaction the reaction rate can be described by the mass action law:

$$\mathrm{r}_j = \mathrm{k_j}^f \prod\nolimits_i \mathrm{c}_i{}^{-\mathrm{v_{ij}}} - \mathrm{k_j}^r \prod\nolimits_i \mathrm{c}_i{}^{\mathrm{v_{ij}}} \tag{1.25}$$

Here k_f and k_r denotes the forward and reverse rate constants. The concentration of species i is defined as c_i. v_{ij} indicates the stoichiometric coefficients. The quotient of the forward and the backward reaction rates is known as equilibrium constant and it is evaluated as follows:

$$K = \frac{[A]^a[B]^b}{[X]^x[Y]^y} \tag{1.26}$$

where the square brackets indicate the concentration values.

1.5.5 Henry's law

The Henry's law may be useful to define the concentration of gas species in the aqueous solution. It states that at constant temperature, the equilibrium concentration

Table 1.1: Equation 1.28 constants for pure water.

Gas	A	B	C	T min.	T max.
-	-	-	-	K	K
O_2	-161.6	8160	22.390	273	348
CO	-178	8750	24.875	278	323
CO_2	-145.1	8350	19.960	273	353

c_a of a gas species in a given volume of liquid depends on the partial pressure p of the gas itself as follows:

$$H = \frac{c_a}{p} \tag{1.27}$$

For CO_2, CO and O_2 the Henry's constant depends on temperature folowing the following Arrhenius type equation:

$$H = \frac{1}{101.325} exp\left(A + \frac{B}{T} + Clog(T)\right) \tag{1.28}$$

The values of A,B and C together with the temperature range of validity are reported in Table 1.1. The unit of measurement of H is $\frac{mol}{m^3 Pa}$. This value may be useful to be utilized with different unit measures, in particular in Table 1.2 the Henry's law costant are given for a temperature of 293.15 K for using the following conversion: The values are taken D2.1 Database of transport coefficients in Matlab environment.

Table 1.2: Henry's law constants at 293.15 [K] for pure water.

Gas	H	H	H
-	$mol/m^3 Pa$	M/atm	$-$
O_2	1.37 x 10^{-5}	1.40 x 10^{-3}	3.36 x 10^{-2}
CO	1.05 x 10^{-5}	1.07 x 10^{-3}	2.56 x 10^{-2}
CO_2	3.90 x 10^{-4}	3.96 x 10^{-2}	9.52 x 10^{-1}

Grant agreement ID: 828838 [56].

Chapter 2

Case study: description

The aim is to develop a multiscale and multiphysics model that will accurately describe and predict the performance of the new photosynthetic membranes to reduce carbon dioxide, as mentioned the model is built with a software called COMSOL Multiphysics which is a platform for finite element analysis that allows the user to solve various equations. The software packages include different physics phenomenon which can be applied on the model which can range from 0 dimension to a 3D geometry. In particular in this thesis work the Transport of diluted species and the chemistry modules are widely employed [57] In this case study only 0D and 1D models are implemented where the physics equation are applied in the finite elements of the geometry to connect each node thanks to a process called meshing which gives an idea on how the selected geometry has been subdivided, this is an important concept because the method is considered reliable when the error is linked to the element size, if element size is bigger the error should be bigger within the limits of calculations and calculator [58].

In Figure 2.1 a graphical representation of the element researched for the case study is given, most of the elements shown at the gas-liquid interface has already been hinted in the introductory part of this thesis work, the green elements characterized with a tail and a head are the surfactants which how has been reported they disposes at the surface according to their amphiphilic behavior In Figure 2.1 free electrons are represented meaning that at the interface there must elements which work as charge transfers ensuring the electrons needed for the carbon dioxide reduction, in the end the concept of photoactive molecules is depicted with those yellow shining drawings, for sure a photoactive molecule has to be present in order to produce an artificial photosynthesis process but in this work the decision of this element has not been faced because the level of the simulation did not reach an enough high level of accuracy, the inclusion of a photosensitizer in the simulations wil be something that must be studied once the other player of the simulation will be well-described. In the gas bulk there are moles of CO_2 which is the reagent together with moles of CO which is the product of carbon dioxide reduction, while in the liquid bulk is mainly a aqueous region where the gas phases are present according to the Henry's law together with the electron transfers inside and possibly

the photoactive molecules.

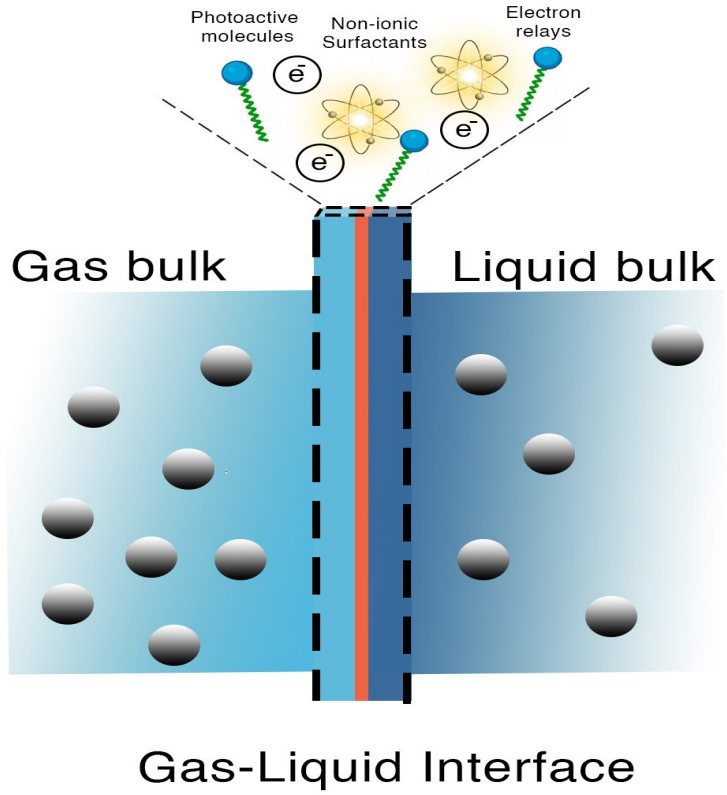

Figure 2.1: Graphical representations of the element researched for the case study.

In the next section the compounds utilized in this case study are analyzed.

2.1 Electron relays

Initially the compounds taken into account for the electron transfer mechanism were mainly two: the quinones, the sodium ascorbate. These chemicals share the characteristic of being water-soluble and also have suitable redox potentials for achieving carbon dioxide reduction, moreover the availability of experimental adsorption data is focal because the COMSOL models requires as an input the set of Langmuir parameters in order to reproduce the adsorption physics. Unluckily it is not common to find numerical results for the adsorption at the gas-liquid interface with the presence of surfactants monolayer. For what concern the sodium ascorbate is a salt produced from the ascorbic acdid ($AscH_2$), it is of interest because it is used as an anti-oxidant which means that it is a reducing agent The ascorbic acid half reaction,

written as a reduction, is shown below:

$$C_6H_6O_6 + 2\,H^+ + 2\,e^- \longleftrightarrow C_6H_8O_6 \tag{2.1}$$

And depicted in Figure 2.2

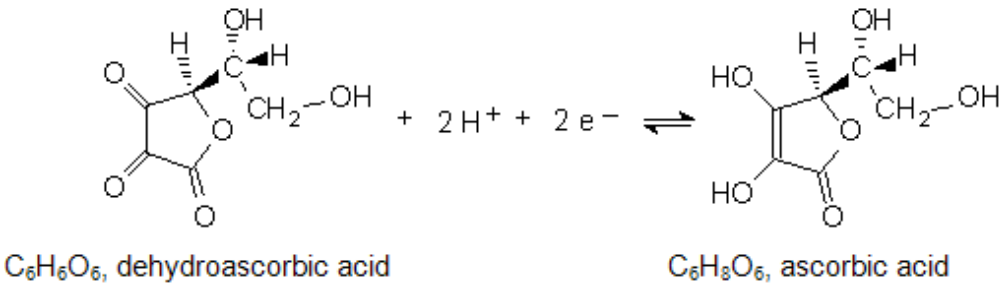

Figure 2.2: Ascorbic acid half reaction of reduction.

The redox potential for the sodium ascorbate is reported to be equal to 100mV with a concentration of 0.01M and a pH of 8 [59, 60]. the potential which is equal to 0.1V seems not high enough to realize carbon dioxide reduction, moreover it is hard to find examples of adsorption phenomana including surfactants, this is why the researched has been more focused on quinones which reports higher redox potentials with the same conditions and they show a larger number of studies in literature.

The quinones are a class of organic compounds which derive from aromatic compounds, in particular the redox reactions involving the hydroquinones, a type of quinones, are of interest. In Figure 2.3 it is possible to observe four different quinones; cathecol, resorcinol and hydroquinone are isomers. Hydroquinone is widely used as industrial solvent, it can often be a raw material from chemistry industry thus they are extensively found in its effluents [61]. Physically Hydro-

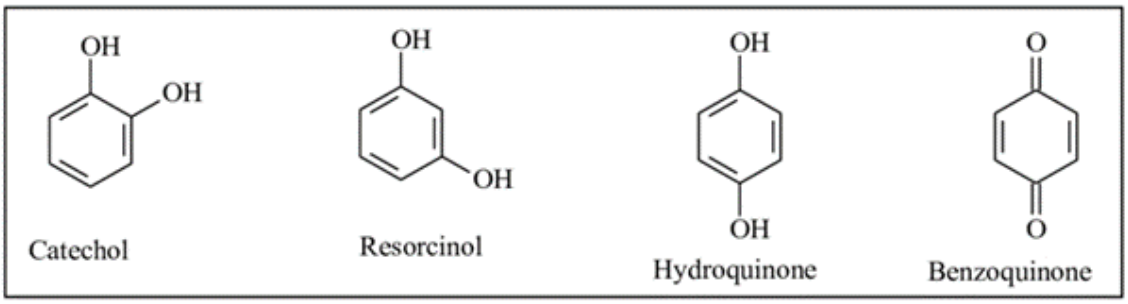

Figure 2.3: Structures of catechol, resorcinol, hydroquinone and benzoquinone [62].

quinone appears as light coloured crystals or solutions mildly toxic by ingestion or skin absorption, it is odorless and shows a slightly bitter taste in aqueous solutions with a solubility of 6.72 g/L at 20 C° [63]. Furthermore because of its simplicity and short response time, quinone/hydroquinone is one of the most studied organic redox

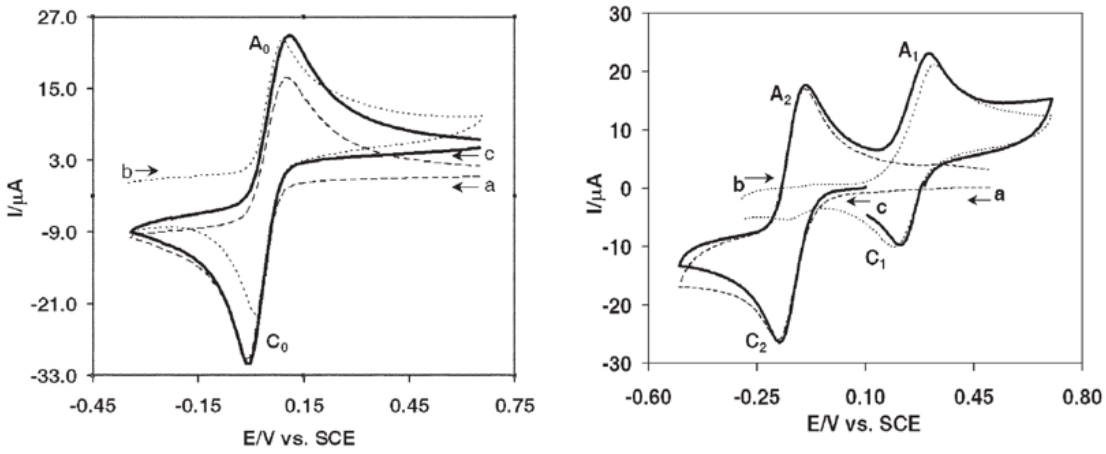

Figure 2.4: LEFT: Cyclic voltammograms of 1 mM (a) Q, (b) H_2Q and (c) QH in buffered solutions of pH 7.0 at a glassy carbon electrode. Scan rate = 100 mVs^{-1} and t=25 ±1°C. RIGHT: Cyclic voltammograms of 1 mM (a) Q, (b) $H2Q$ and (c) QH in aqueous unbuffered KCl 0.15 M solution at a glassy carbon electrode. Scan rate=100 mV s^{-1} and t=25 ±1°C [64].

couples, at constant temperature and well-buffered solution its potential is a linear function of pH with a slope of 59 mV, which means that one elctron is involved. In Figure 2.4 LEFT and Figure 2.4 RIGHT the work by Rafiee and Nematollahi is depicted [64], it shows the cyclic voltammetry, a potentiodynamic electrochemical measurement, recorded in buffered solutions and in unbuffered solutions. The two opposite peaks show that the process is reversible, after the first reduction (first peak) the reduced species start to be re-oxidized giving as a result a current with opposite polarity respect to the beginning. Hence, CV data can provide information about redox potentials and electrochemical reaction rates [65]. In Figure 2.4 LEFT the buffered case, the anodic peak is due to oxidation of H_2Q to Q and the cathodic peak is due to the reduction of the produced Q to H_2Q.The oxidation half reaction can be written as [66]:

$$\mathrm{H_2Q} \longrightarrow \mathrm{Q} + 2\,\mathrm{H^+} + 2\,\mathrm{e^-} \tag{2.2}$$

on the other hand in the unbuffered case in both negative and positive going shows two anodic and cathodic peaks. The behavior can be descrived by the following equations:

$$\mathrm{Q} + \mathrm{H_2O} \underset{-2\,\mathrm{e^-}\ \text{Generation of peak}\ \mathrm{A_2}}{\overset{+2\,\mathrm{e^-}\ \text{Generation of peak}\ \mathrm{C_2}}{\longleftrightarrow}} \mathrm{HQ^-} + \mathrm{OH^-} \tag{2.3}$$

$$\mathrm{Q} + \mathrm{H_3O^+} \underset{-2\,\mathrm{e^-}\ \text{Generation of peak}\ \mathrm{A_1}}{\overset{+2\,\mathrm{e^-}\ \text{Generation of peak}\ \mathrm{C_1}}{\longleftrightarrow}} \mathrm{HQ^-} + \mathrm{H_2O} \tag{2.4}$$

In the first reaction water acts as a proton donor in the reduction of Q but in the other case water acts as an acceptor of the proton, which is generated in the oxidation of HQ^-. In this second case the difference between oxidation potential peaks is about 0.41 V. The two graphics present a trend for QH too, its name is

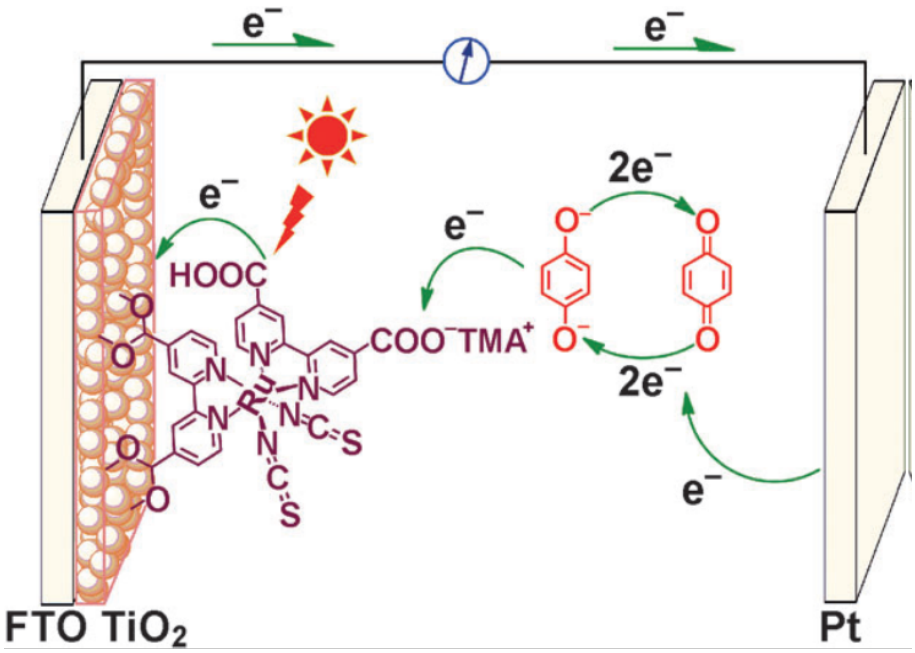

Figure 2.5: Dye-Sensitized Solar Cells Based on Hydroquinone/Benzoquinone as a Bioinspired Redox Couple using a hybrid electrolyte involving tetramethylammonium (TMA) hydroquinone/benzoquinone couple. FTO = fluorine-doped tin oxide [67].

quinhydrone, some studies support the idea of an intermediate step in hydroquinone oxidation through the formation of quinhydrone.

In this thesis hydroquinones are of interest because quinones are small, have a tuneable redox potential, they are readily prepared and in photosynthesis processes they can be used as electron acceptors or donors. The first examples in which quinones were used are reported in 1978 and 1979 by Kong and Tabushi, respectively [68, 69]. In the work of Tabushi and coworkers a donor-acceptor duet were reported using porphyrins, synthetic molecules closely related to natural chlorins and bacteriochlorins such as the chlorophylls in the sense that they have an absorption band in the visible spectrum hence they can absorb light. In the cited work the couple porphyrins/hydroquinone showed that the rate of electron transfer is proportional with hydroquinone concentration and foresaw that the donor-acceptor combination seems to be a mechanism that can be applied to a wider range of electron transfer systems. Porphoryn-quinone duets show charge recombination issues, it can be more rapid than charge separations, they suffer from a very low charge separation lifetimes, typically in the order of 100ps which is not enough for practical applications. Although this dyad has limitations as reaction centres, their study by a multitude of researchers has provided a great deal of useful information concerning the effects of structure and environment on electron transfer rates [21, 70].

Over the years studies has been developed for application involving solution-based Q/H_2Q couples in batteries, Dye Sensitized Photoelectrochemical Cells (DSPEC), or related applications [71]. Nawar and coworkers developed an organic molecule-

based flow battery using quinones which are a good alternative because they offer low toxicity, low cost and high reversibility in electrochemical reactions. In the work they developed a battery in low pH aqueous solutions using quinone/hydroquinone on the positive electrode and H_2/H^+ on the negative electrode, in the future the quinones solubility could be modified to increase the current density, furthermore it is possible to develop quinones with redox potential for use at the negative electrode in more to make an all Q/H_2Q flow battery [72].

In Figure 2.5 a dye-sensitized solar cells, which can be an economical method to convert solar light to electricity. In this case it is utilized a hybrid electrolyte involving the tetramethylammonium hydroquinone (HQ)/benzoquinone (BQ) redox couple . HQ and BQ form the charge transfer complex called quinhydrone. The results showed that when HQ/BQ is applied to a hybrid electrolyte using a N719 as photosensitizer the efficiency outperform the efficiency of 8.0 % a system employing the better known pure iodine-based electrolyte thanks to the higher gap differences between TiO_2 and the redox couple [67].

Because of its wide applications in charge transfer processes and water solubility the hydroquinone has been utilized in this thesis work as an electron donor to reduce carbon dioxide. Since it is not trivial to find researches in the bibliography

Table 2.1: Surface area from BET and parameter values of the Langmuir models for the monohidroxilated phenols adsorption on the CAG, CAR, CAO at pH 7.

Adsorbate	Simple	Surface area BET	Q_{max}	K	$R^2L.$	$R^2F.$
-	-	m^2/g	mg/g	L/mg	-	-
Catechol	CAG	1140	238.10	4.2 x 10^{-3}	0.90	0.96
/	CAR	1171	181.82	5.7 x 10^{-3}	0.97	0.87
/	CAO	1181	178.57	3.9 x 10^{-3}	0.97	0.96
Resorcinol	CAG	1140	178.57	6.0 x 10^{-3}	0.94	0.95
/	CAR	1171	232.56	5.5 x 10^{-3}	0.97	0.89
/	CAO	1181	163.93	4.8 x 10^{-3}	1.00	0.97
Hydroquinone	CAG	1140	169.49	1.3 x 10^{-2}	0.99	0.84
/	CAR	1171	232.56	8.3 x 10^{-3}	0.97	0.84
/	CAO	1181	232.56	3.2 x 10^{-3}	0.80	0.87

with experimental results linked to hydroquinones or generally phenolic compounds adsorption at the gas-liquid interface, it has been made the decision to go deeper into the analysis of the scientific papers where H_2Q or its analogues adsorb at a solid-liquid interface. According to the results found in literature granular activated carbon (GAC) shows the best adsorption capacity for removing phenolic compounds this is why it was often utilized in waste water treatments. Nevertheless GAC is

expensive and it loses adsorption efficiency after regeneration causing a limitation in its use, this has favoured the development of less costly adsorbents such as activated carbon cloth, waste Fe(III)/Cr(III) hydroxide, hypercrosslinked resin, TiO_2 surface, organoclays, bagasse fly ash and activated cashew nut shell [73]. Since its use against pollutants there are many studies for what concerns the activated carbon, its surface is typically divided in three main zones: the carbon basal planes, the oxygen-containing groups and inorganic ash, the studies on mechanism of adsorption of phenolic compounds report that the number of available sites increase with the amount of oxygen content of carbon. This is true until the adsorbate concentration does not get high enough so that the phenols molecules get packed more tightly on the surface, in this case it is reported that the influence of oxygen sites is lower. Moreover the aromatic rings can interact through a donor-acceptor interaction with the basal planes of AC in a not effective way [74]. An article which has been taken into consideration is the paper of Blanco-Martinez and coworkers [61] because of its availability of data on the different isotherms. The work studies the adsorption of different monohydroxylated phenols on granular activated carbon (GAC), oxidized activated carbon (CAO) and reduced activated carbon (CAR). CAO and CAR are obtained via chemical modifications, namely oxidation and reduction on the surface. The adsorption isotherms data are obtained by putting 0.500–0.250 g of the carbonaceous samples in contact with a 50mL volume of monohydroxylated phenols solutions, the effect of pH on the yield of adsorption is analyzed, showing that the amount of the phenols adsorbed diminishes as the pH solution increases. In Table 2.1 it is possible to see the Langmuir isotherm parameters for the phenols taken into account at a neutral pH. R^2 represents the coefficient of determination which is a statistical measurement that examines how differences in one variable can be explained by the difference in a second variable, when predicting the outcome of a given event, namely it shows how well a regression is fitting the data, in this case it is a measure of accuracy of the model compared to experimental outcomes both for a the Langmuir adsorption isotherm and the Freundlich adsorption model. Another material that possesses a large specific surface area are carbon nanotubes (CNT), they consist of sheets of carbon atoms covalently bonded in hexagonal arrays that are seamlessly rolled into a hollow cylindrical shape which can be arranged in a single rolled-up graphite sheet (SWCNT) structure or in several graphite layers called MWCNT. Of interest for this thesis work is a study conducted by Felipe Augusto Gorla and coworkers which develops an electrochemical study for the simultaneous determinantion of phenolic compounds like hydroquinone, catechol, 4-nitrophenol and acetaminophen, using an electroanalytical sensing system based on a multi-walled carbon nanotubes paste electrode in the presence of surfactant. The work shows that the surface concentrations for all phenolic compounds in the presence of the surfactant is higher compared to the a glassy carbon electrode, affirming that the presence of surfactant improves the detectability of phenolic compounds, the surface concentration value reported for the hydroquinone species is equal to 4.43 x $10^{-9} mol/cm^2$ with the presence of surfactant at neutral pH and 1.16 x $10^{-9} mol/cm^2$

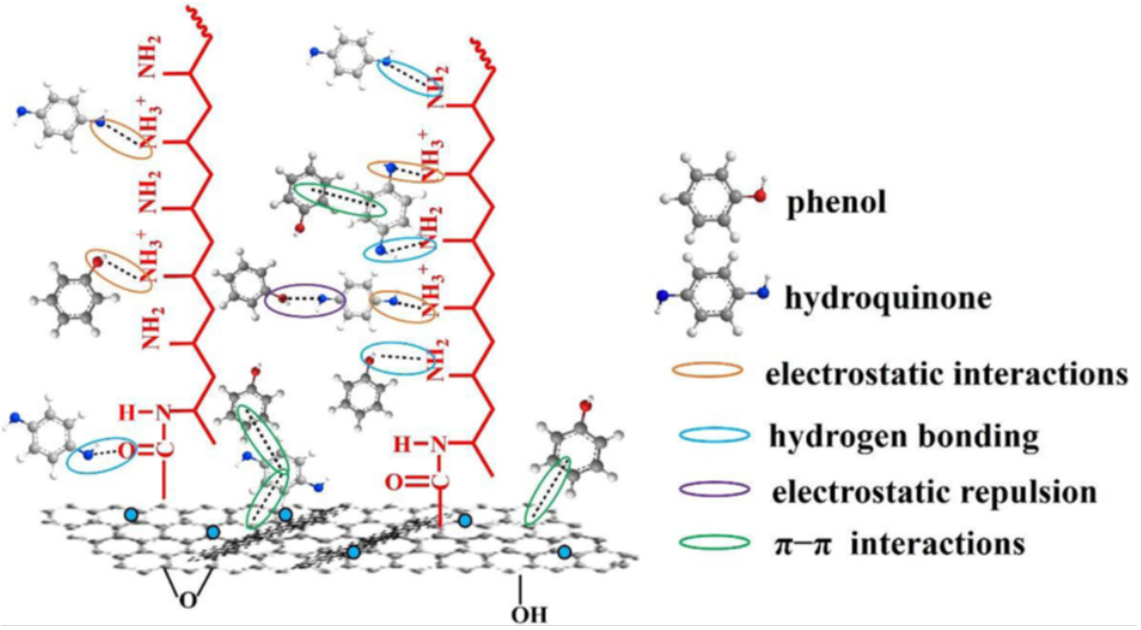

Figure 2.6: Schematic of the adsorption mechanism of phenol, hydroquinone and the phenol-hydroquinone mixture on the $PVAm-GO-(o-MWCNTs)-Fe_3O_4$ nanocomposite surface [75]

in its absence [76]. This work is interesting because it contains insights about the hydroquinones interaction with a surfactant but no information is provided about the adsorption isotherm model nor the equilibrium constants, this is why it has been analyzed a case where a nanohybrid multi-walled carbon nanotubes is employed in a batch experiment to study the removals property of the material toward the hydroquinones. This study is a good candidate because it describes a situation where 30 mg of magnetic nanocomposite powder is added to a 20 mL of various initial concentrations of phenol (100, 200, 300 mg/L), then the mixture is thermostatically shaked. In Figure 2.6 it can be seen the adsorption mechanism of phenols on the nanocomposite material together with the electric and bounds interaction that outline this interaction while in Table 2.2 the isotherm parameters are given. Once again, the table contains the value of the Freundlich isotherm coefficient of determination $R^2F.$, Both the models show that they can fit well the experimental results [75]. Batch experiments are of interest because the phenolic compounds and

Table 2.2: Surface area from BET and isotherm parameters for the adsorption of phenol and hydroquinone onto the nanocomposite.

Adsorbate -	Surface area BET m^2/g	Q_{max} mg/g	K_L L/mg	$R^2L.$ -	$R^2F.$ -
Phenol	341.8	224.21	0.0816	0.9989	0.9860
Hydroquinone	341.8	293.25	0.1916	0.9994	0.9846

the adsorbents are in contact inside an aqueous media which is the same situation that the soap film experience in the simulation that will be described in the next sections. The next work analyzed it deals again with batch experiments, in this instance the hydroquinones adsorb at the matal surface, in the study Abugazleh and coworkers the adsorption of catechol and hydroquinone on the surface of TiO_2 (anatase) and Fe_2O_3 (hematite) is investigated. The experiment is conducted in a neutral pH solution in the presence of a buffer where the oxides are added. It is reported that the maximum amount of adsorption of phenolic compounds is reached around a pH of 7 for the anatase and 8 for the hematite. According to this study the catechol is better fit by a Langmuir type isotherm while for the adsorption of the hydroquinone the Freundlich model suits better its behavior. This is the first case where the Langmuir theory is not the best method to describe the process of adsorption of hydroquinones at different surfaces. In Table the values of the maximum adsorption capacity of CAT and HYD for the different adsorbent/adsorbate couples are given. It is possible to notice that higher adsorption values of HYD are observed on TiO_2 than on Fe_2O_3, while for CAT, adsorption values are seen to be higher in Fe_2O_3 than in TiO_2 [77]. This article is on contradiction with what

Table 2.3: Surface area from BET and maximum adsorption capacity for the different adsorbent/adsorbate couples.

Adsorbent -	Adsorbate -	Surface area BET m^2/g	Q_{max} mg/g
TiO_2	Catechol	4.91	108.87
Fe_2O_3	Catechol	13.04	361.1
TiO_2	Hydroquinone	4.91	63.52
Fe_2O_3	Hydroquinone	13.04	58.49

encountered in previous ones, at the beginning hydroquinones and catechols were meant to be similar for what concern the adsorption behaviors while this article display a difference not only in numerical fitting but also in the general trend with respect to different surfaces. This means that a further study on different articles is needed to better understand the adsorption mechanism of phenolic compounds. The next papers analyzed deal with cheaper materials like ashes and resins still with an immersion methodology.

Phenolic compounds are reported to adsorb on carbon rich bagasse fly ash (BFA), a waste material from the sugarcane, in particular in the research of Srivastava and coworkers the results are given through a batch study where the influence of pH, concentration and temperature is studied, not only for the ashes but also on commercial and laboratory grade activated carbon, ACC and ACL, respectively. As it can be expected the adsorption of phenol increases with increase in adsorbent dosage, the effect of pH is the one already spotted for almost all the researches that were pursuing this result, namely as the pH increase the amount of adsorbed

Table 2.4: Surface area from BET and isotherm parameters for the adsorption of phenol onto the fly ash and activated carbon.

Adsorbate -	Surface area BET m^2/g	Q_{max} mg/g	K_L L/mg	$R^2L.$ -	$R^2F.$ -
BFA	168.39	23.832	0.0884	0.98970	0.99450
ACC	336.60	30.2187	0.0291	0.99160	0.97740
ACL	492.00	24.6458	0.2391	0.95430	0.99451

species decrease. While the effect of temperature is shown in Figure 2.7 in the different isotherms, with the increase in temperature the adsorptivity of phenol increases, since the adsorption is a exothermic process this is not expected as a result but it means merely that the adsorption process is controlled by diffusion which benefits from a temperature rise [78]. The research fits the model with six different isotherms, in Table 2.4 the Langmuir parameters together with the R^2 of the Freundlich isotherm which is the "competing" model with the case studied in this thesis work. The values are given for the three adsorbents at 30 °C.

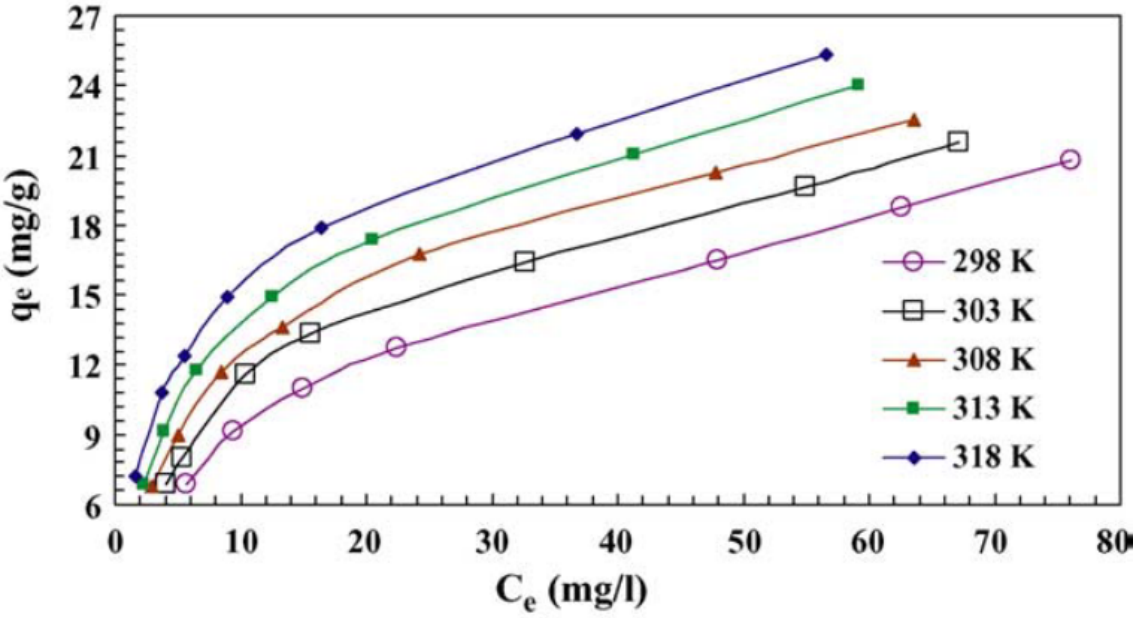

Figure 2.7: Equilibrium adsorption isotherms at different temperature for phenol–BFA system. initial pH: 6.5; BFA dosage: 10 g/l [78].

The model is able to fit well the adsorption behavior of phenols on the different adsorbents for both isotherms type [78]. For what concerns the resins an hypercrosslinked resin HJ-1 was developed for adsorbing catechol and resorcinol in aqueous solution by Jianhan Huang and coworkers. In this case it is clear that the adsorption capacity decreases with increase of the temperature. Unfortunately there

Table 2.5: Isotherm parameters for the adsorption of catechol and resorcinol onto the resin.

Adsorbate -	Temperature K	Q_{max} mg/g	K_L L/mg	$R^2L.$ -
Catechol	293	133.33	0.0039	0.9876
Catechol	303	126.58	0.0033	0.9892
Catechol	313	125.00	0.0026	0.9924
Resorcinol	293	128.21	0.0033	0.9884
Resorcinol	303	129.87	0.0026	0.9777
Resorcinol	313	128.21	0.0022	0.9902

are no available results for hydroquinones. In Table 2.5 the Langmuir parameters are available at different temperatures [79]. In this study the value of the surface area according to BET theory is not available, it is found in literature equal to around 730 m^2/g [80].

Eventually it matters to talk about the adsorption of phenolic compounds on organoclays. In the years they showed up as a cheap way to remove organic pollutants, the main material is bentonite thanks to its abundance and the low cost of the mineral, moreover their surface properties can be modified by exchange reactions. Again for these materials a scientific article which involve batch experiments is selected because the aqueous solution is considered a good environment to reproduce the soap film which is enclosed by surfactant molecules, this is why, when possible, researches which include surfactant molecules are analyzed. This is the case where a bentonite is modified on the surface with different organic cations. The work of Yıldız and coworkers showed the result of experiments carried out between hydroquinones and organically modified clays [81]. In Table 2.6 it is possible to see the experimental data at 298 K at neutral pH. After this review about the adsorption

Table 2.6: Langmuir isotherm model parameters of hydroquinone adsorption on HDTMA-B and ODTMA-B.

Adsorbent -	Surface area BET m^2/g	Q_{max} mg/g	K_L L/mg	R^2 -
HYD/HDTMA	35.68	16.6389	0.0073	0.9570
HYD/ODTMA	28.92	12.0482	0.0090	0.9885

on different adsorbents where the research has been focused on finding the one that can best fit the idea of the soap film the author decided to use the surface area of the pores according to the BET theory for the article selection. This decision comes from the assumption that there is less availability of adsorption sites at the gas-liquid interface compared to the solid-liquid surfaces. Briefly, the Brunauer-Emmett-Teller

Table 2.7: Summary of Langmuir isotherm model parameters of adsorption of phenolic compounds on different adsorbents.

Adsorbate -	Adsorbent -	Surface area BET m^2/g	Q_{max} mg/g	Reference -
Hydroquinone	CAG	1140	169.49	[61]
Hydroquinone	CAR	1171	232.56	[61]
Hydroquinone	CAR	1181	232.56	[61]
Hydroquinone	Nanocomposite	341.8	293.25	[75]
Hydroquinone	TiO_2	4.91	63.52	[77]
Hydroquinone	Fe_2O_3	13.04	58.49	[77]
Phenool	Bagasse fly ash	168.39	23.832	[78]
Catechol	Resin	730	133.33	[79]
Resorcinol	Resin	730	128.21	[79]
Hydroquinone	HDTMA	36.68	16.6389	[81]
Hydroquinone	ODTMA	28.92	12.0482	[81]

theory is used to explain the adsorption mechanisms between gas molecules and solid surfaces, moreover it is a basis for the measurement of the surface area of materials, it is focal to understand that it is not an absolute measure because the value of the specific areas of pores may depend on the adsorbate too [82, 83]. The specific surface area is an important measurement and almost all the articles taken into account contain this interaction information between adsorbent and adsorbate, to better define the problem in Table 2.7 the results from bibliography are grouped with their reference.

It is possible to notice that different materials (adsorbents) range from a wide numbers of surface area according to BET calculations. As stated the activated carbon and carbon nanotubes display a really good affinity for the phenolic compounds and this is why they are often employed in waste water treatments but from this thesis work point of view they have a too large availability of adsorption sites to reproduce the phenomena, this is reason why the author discarded them, even though there are reported results involving surfactants. The titanium and iron oxides have brought the issue that from an adsorption point of view, not necessary the isomers of hydroquinones have the same properties. Unfortunately no batch experiments or experiments related to solutions immersion have been analyzed with the hydroquinones as adsorbate but only with similar molecules and isomers, moreover given the large surface area of pores, the resins and the ashes have not been taken into account to integrate their data with the surfactant monolayer. The adsorbents which shows the smaller capacity of adsorption are the bentonites, while the BET surface area is bigger compared to the oxides. Eventually the bentonites have been

preferred respect to the titanium and iron surfaces because in the analyzed article there is a case where exactly hydroquinones are adsorbing on surfactant layers which is considered by the author the overall best fit, considered the environment of the experiment, the small uptake of hydroquinones, as it can be seen from Q_{max} value and the low surface area of pores.

2.2 Surfactant used in the SoFiA project

In the next chapter the simulation result are compared with experimental results which employ $C_{12}E_6$ as a surfactnat. $C_{12}E_6$ is a non-ionic surfactant having covalently bonded oxygen-containing hydrophilic groups, which are bonded to hydrophobic parent structures. In Figure 2.8 it is possible to see a schematic representation. It is formed by the ethoxylation of dodecanol (lauryl alcohol) to give a material with six repeat units of ethylene glycol. Thomas and coworkers described the behavior of this category of surfactants at the gas-liquid interface using neutral reflection, a technology which dissolves the surfactant in NRW (null reflecting water), a mixture of D_2O and H_2O in a composition in order to make its neutron scattering length equals to zero, to determine the surface coverage [84]. The results reported on surfactants with different numbers of ethylene oxide groups, showing that the extent of overlap between the alkyl chain and ethylene group increases with the number of n (C_nE_m). Studies of Chanda and Bandyopadhya who simulated complete monolayers of $C_{12}E6$ at the gas water interface to study their dynamical properties, reported that the long polar headgroups are more tilted toward the aqueous layer which is what this thesis wants to simulate [85, 86].

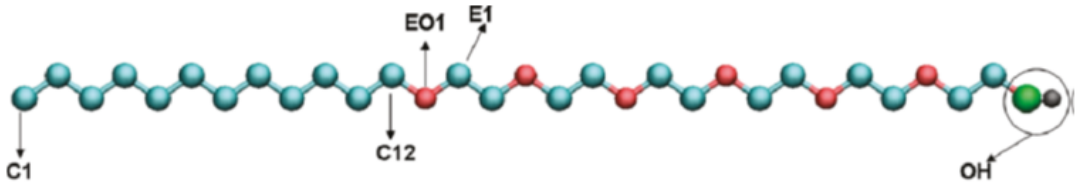

Figure 2.8: Schematic representation of $C_{12}E_6$. C1 represents the 1st alkyl group in the tail, C12 is the 12th alkyl group in tail,EO1 is the oxygen in the 1st ethylene glycol group,E1 is the ethylene in the 1st ethylene glycol group, and OH is the terminal OH group [86].

Chapter 3

Case study: simulations

The case study simulations are divided in three main sections:

- 1D model of a buffer with carbon dioxide dissociation;
- Validation of a 0D model;
- 1D COMSOL model of a surfactant monolayer with chemical reactions.

A validated buffer model is developed in the first section together with a model for CO_2 dissociation in aqueous solution. In the validation part the COMSOL solutions are compared to experimental results in order to prove that the models are capable of consistently reproduce physical phenomena to a specified confidence level. In the last part, half of a soap film is reproduced with a 1D model, in this section the monolayer is described and the parameters tuning is carried out to better understand how the different factors are influencing the global reaction.

3.1 Buffer model and carbon dioxide dissociation

3.1.1 Buffer model

It is useful to validate a buffer model in order to utilize it in the next simulation in those cases where there is a need to keep the process at neutral pH. Basically a buffer consists in a solution which can react to pH change upon the addition of an acidic or a base. In fact it is able to neutralize the pH modifications keeping the overall solution pH relatively stable [87]. The buffer implemented is the phosphate buffer which is composed of the four species shown in Figure 3.1. The equilibrium reactions which participates int the phosphate buffer are listed below [89]:

$$H_3PO_4 \overset{K_{ph1}}{\longleftrightarrow} H^+ + {H_2PO_4}^- \tag{3.1}$$

$${H_2PO_4}^- \overset{K_{ph2}}{\longleftrightarrow} H^+ + {HPO_4}^{2-} \tag{3.2}$$

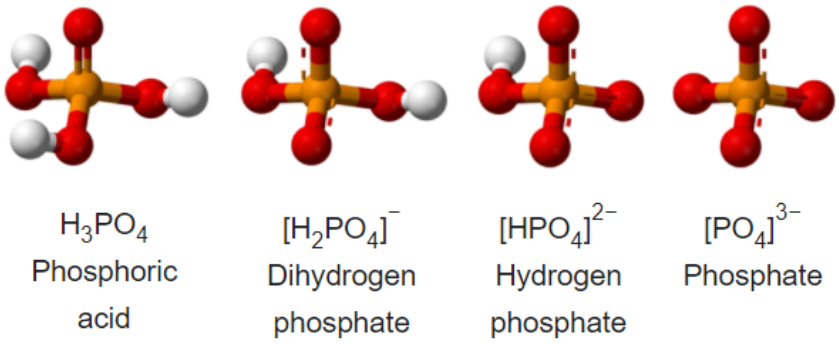

Figure 3.1: Phosphate buffer involved species [88].

$$HPO_4^{2-} \xleftrightarrow{K_{ph3}} H^+ + PO_4^{3-} \tag{3.3}$$

In Table 3.1 values of pk and equilibrium constants are given, K and pk are linked by the next relation:

$$K = 10^{-pK} \tag{3.4}$$

Table 3.1: Phosphate buffer equilibrium.

Equilibrium -	pK -	Equilibrium constant mol/L
$H_3PO_4 <-> H_2PO_4^- + H^+$	2.14	7.5 x 10^{-3}
$H_2PO_4^- <-> HPO_4^{2-} + H^+$	7.20	6.2 x 10^{-8}
$HPO_4^{2-} <-> PO_4^{3-} + H^+$	12.37	2.14 x 10^{-13}

Moreover the water dissociation is included in the model where the water is indicated as a solvent:

$$H_2O \xleftrightarrow{K_{H_2O}} OH^- + H^+ \tag{3.5}$$

With a initial concentration of H_2O equals to 1 mol/m^3 and an equilibrium constant 1 x 10^{-8}.

These equations are implemented in COMSOL in a 1D model, the simulation exploits a geometry consisting of a line 0.01 m long, the diffusion coefficients of the species are set at 1 x 10^{-9}. The model implemented is time dependent with the time ranging from 0 to 900 seconds with a time-step of 1s.

The result is depicted in Figure 3.2 where the phosphoric acid speciation is presented, the dotted lines represent the experimental data [90] for the four species while the points are the results from the COMSOL simulation, the dots are obtained by varying the initial concentrations. It is possible to notice that the model is able to reproduce experimental values with a difference for the most extreme values which are not realistic, indeed a pH of 15 is reported.

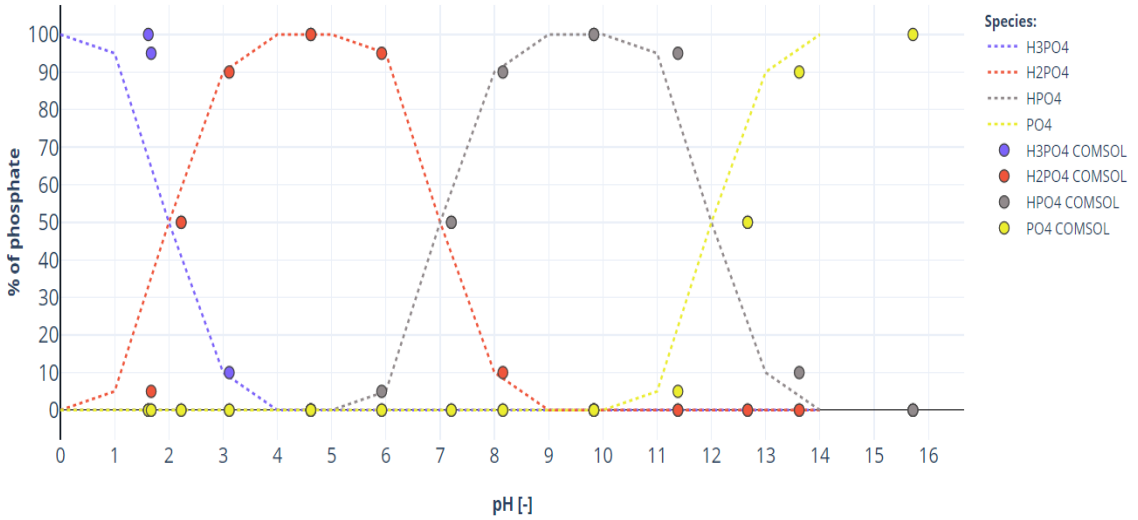

Figure 3.2: Phosphoric acid speciation.

3.1.2 Carbon dioxide dissociation model

The carbon dioxide dissociate in aqueous solutions giving a acidic pH which depends on the CO_2 aqueous concentration. The phenomenon is described by the following two equations:

$$CO_2{}^{aq} + H_2O \overset{K_{c1}}{\longleftrightarrow} H^+ + HCO_3^- \tag{3.6}$$

$$HCO_3{}^- \overset{K_{c2}}{\longleftrightarrow} H^+ + CO_3{}^{2-} \tag{3.7}$$

The values of the equilibrium constant and the pK comes from the research of Singh

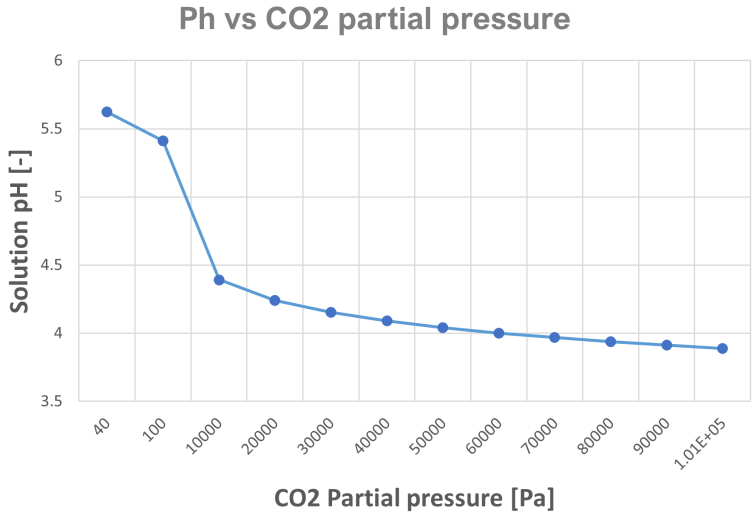

Figure 3.3: pH values vs CO_2 partiale pressure without a buffer.

and coworkers [91] which are equals to $K_{c1} = 4.26$ x 10^{-4} and $K_{c2} = 5.60$ x 10^{-8}.

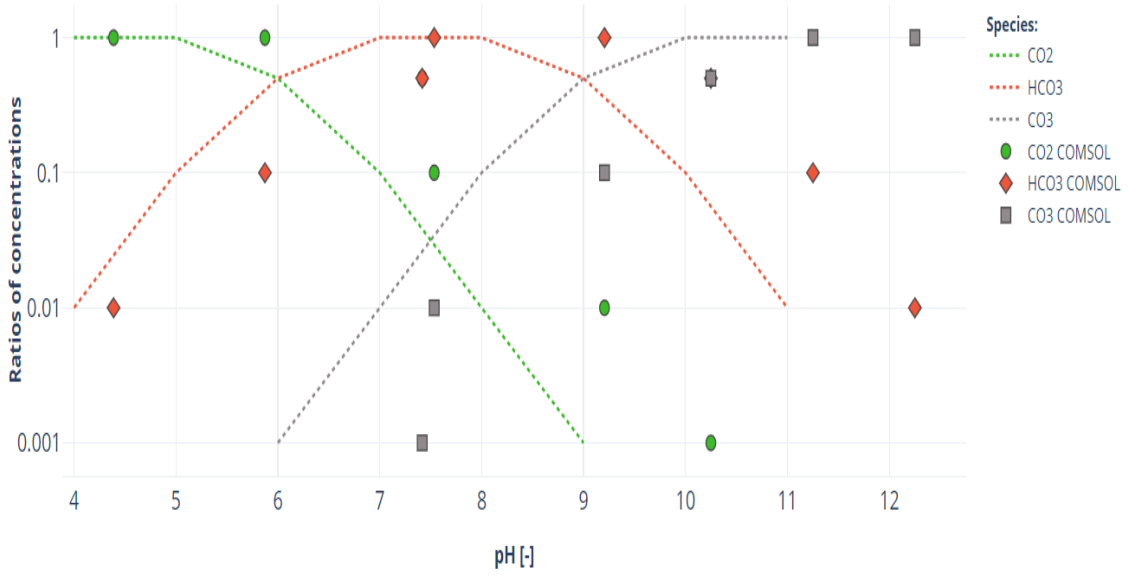

Figure 3.4: Carbonate speciation.

Moreover the diffusion coefficients are given for the different species in Table 3.2 listed in the next subsection.

For what concern the initial concentration values the water is again indicated as a solvent with a starting value of 1 mol/m^3 and the for the carbon dioxide the input is dependent on the outer partial pressure of CO_2 through the Henry's law with the constant set equals to 3.9 x 10^{-4} mol/m^3Pa [56].

In Figure 3.3 the result of the COMSOL simulation is depicted where the pH is analyzed versus the partial pressure of the CO_2 in an hypothetical surrounding environment. The figure shows what expected: with the increase in the concentration value of the carbon dioxide the solution becomes acidic which is a result reported by many studies linked to seawater pH trend [92].

Furthermore in Figure 3.4 the carbonate speciation in seawater and from the COMSOL model are given. The dotted lines come from experimental data using seawater as a solution while the dots, the rhombus and the squares represent the result of the simulation for $CO2$, HCO_3^- and CO_3^{2-}, respectively. The model is able to reproduce the trend but it is possible to notice a "translation" in the numerical values towards more basic pH values, this is reported to be connected to the seawater effect [92] while in the simulation the pure water has been taken into account.

3.1.3 Carbon dioxide dissociation model with the buffer

Eventually the two models are combined to check if the buffer solution is able to stabilize a solution where the CO_2 dissociation is taking place. The model is still time-dependent with the time ranging from 0 to 900s with a time-step of 1 second, the geometry is also the same of the previous two simulations.

Table 3.2: Diffusion and concentration input values for COMSOL model.

Species -	Concentration mol/m^3	Diffusion coeff. m^2/s
Hydrogen (H^+)	1 x 10^{-4}	9.311 x 10^{-9}
Hydroxide (OH^-)	1 x 10^{-4}	5.273 x 10^{-9}
Water (H_2O)	1	2 x 10^{-9}
Phosphate buffer (H_3PO_4)	0	1 x 10^{-9}
Phosphate buffer ($H_2PO_4^-$)	50	1 x 10^{-9}
Phosphate buffer (HPO_4^{2-})	50	1 x 10^{-9}
Phosphate buffer (PO_4^{3-})	0	1 x 10^{-9}
Hydrogencarbonate (HCO_3^-)	0	1.185 x 10^{-9}
Carbon trioxide (CO_3^{2-})	0	0.923 x 10^{-9}
Carbon dioxide (CO_2)	P_{CO_2} * H_{CO_2}	1.91 x 10^{-9}

The reactions implemented are the same too, including the water dissociation, in Table 3.2 the initial concentration values and the diffusion coefficents are listed, the diffusivity parameters are taken from the supplementary information of this reference [91].

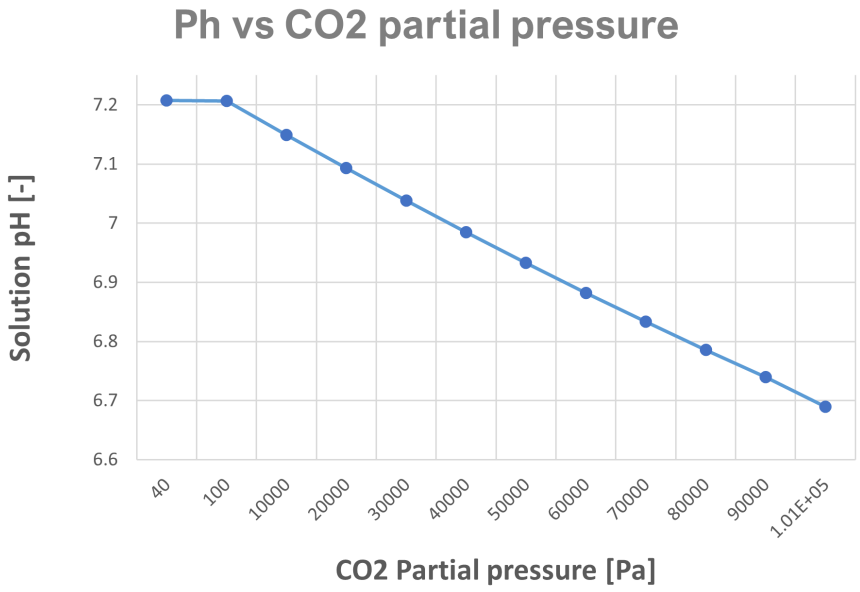

Figure 3.5: pH values vs CO_2 partial pressure with a buffer.

As shown in the table the initial value of for the carbon dioxide inside the liquid 1D line is not given because it is free to move according to the gaseous CO_2 partial pressure. The input values for the buffer are taken randomly in a way that the starting pH was already 7 as shown in Figure 3.2. The result of this simulation is

depicted in Figure 3.5, it is possible to notice that the buffer implemented is able to keep the solution pH almost neutral even when the partial pressure of CO_2 is equal to 101,325 Pascal (1 atm).

This simulation is useful because it provide a working model of a buffer, moreover the carbon dioxide dissociation may be implemented in further simulations.

3.2 Validation of the 0D model

The chosen paper deals with photocatalytic water oxidation in a homogeneous reactor, the aim of validation is to reproduce the oxygen evolution measured experimentally respect to time. Since it is homogeneous it possible to simulate it thorough a 0D model which is for sure easier to implement.

3.2.1 Microkinetic model for homogeneous photocatalytic water oxidation

The examined paper's name is "*Rate and Stability of Photocatalytic Water Oxidation using* $[Ru(bpy)_3]^{2+}$ *as Photosensitizer*" by Limburg and coworkers [93]. The research on water oxidation has been becoming of focal importance recently, in thirty years the turnover frequencies (TOF) of catalysts has been reported to be increased from 2 x 10^{-3} s^{-1} to more than 300 s^{-1}, together with their stability, reflected by turnover numbers (TONs) which increased over 3 orders of magnitude [94]. In this case a photocatalytic process is used and many parameters are introduced like those linked to the oxidative quenching of the photosensitizer (PS), its concententration and the light intensity. Furthermore the type of buffer, the pH solution value and the the catalyst concentration must be optimized because everyone can play a role in the stability and rate production of the system. The aim of this scientific article is to understand what the bottleneck of photocatalytic water oxidation is. In Fig-

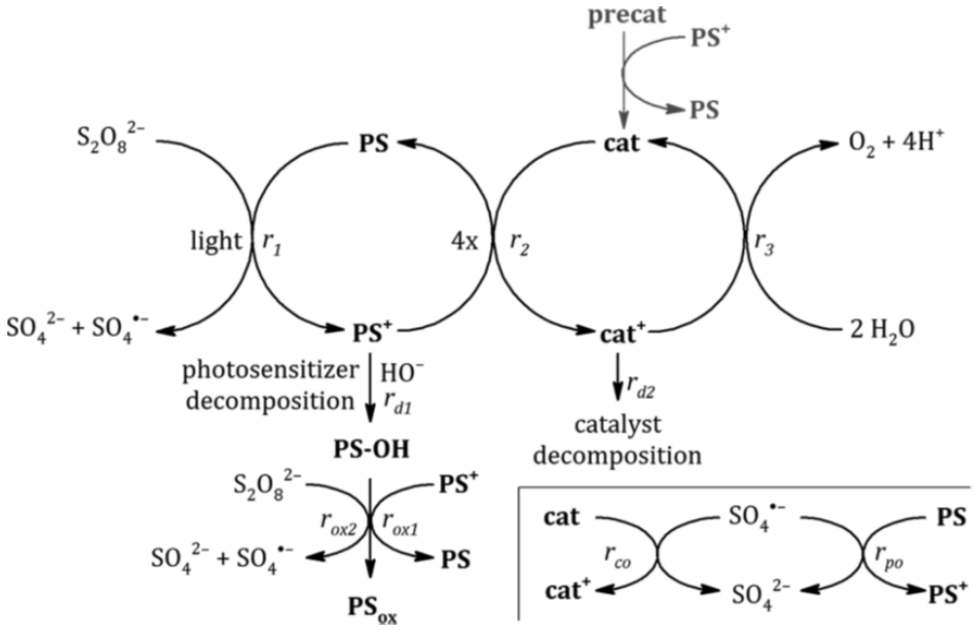

Figure 3.6: A global kinetic overview of the photocatalytic water oxidation process, the three intermediate steps are written as r_1, r_2 and r_3, r_{d1} and r_{d2} are the rates of photosensitizer decomposition and catalyst decomposition, respectively. The initial photosensitizer decomposition products can be further decomposed with rates r_{ox1} and r_{ox2} [93].

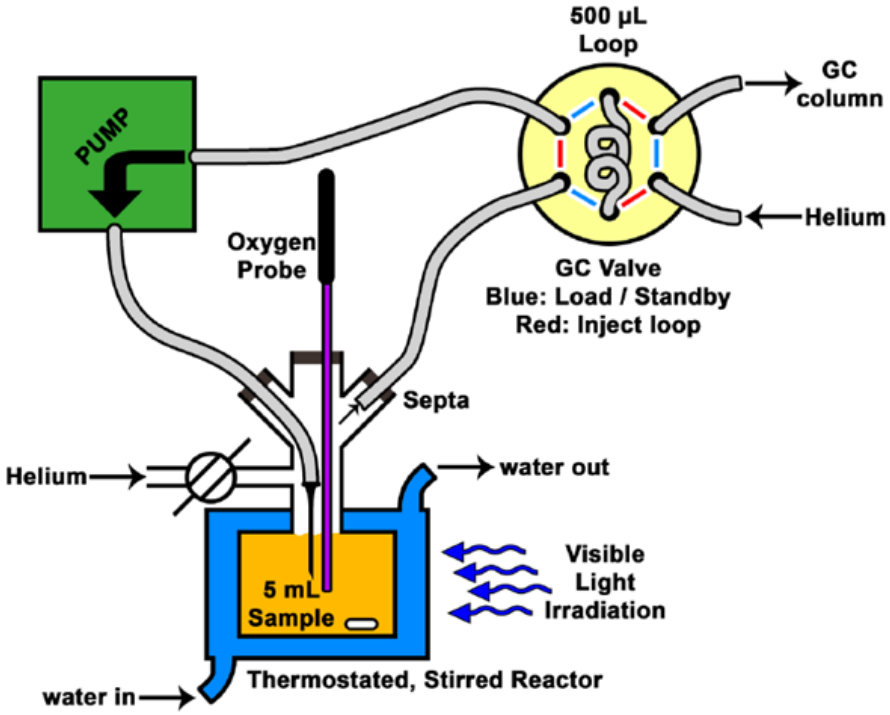

Figure 3.7: Oxygen evolution setup (scheme) [95].

ure 3.6 the three steps of the overall oxidation process are depicted, the first step is formation of PS^+ (standard molecules which lost an electron) via light absorption of PS (standard molecule) state of the photosensitizer and oxidative quenching of PS^* (excited molecule) by $S_2O_8^{2-}$, the peroxydisulfate is the electron acceptor. The second step depicted by the rate r_2 is the electron transfer from the catalyst to PS^+, the third step is O_2 production at the water oxidation catalyst. Besides it is possible to notice that both the catalyst and the photosensitizer can suffer from decomposition and SO_4^- can oxidize both. At the top right there is a precatalyst reaction which is needed to create active species for some kind of water-oxidizing catalysts. In the experiments $[Ru(bpy)_3]^{2+}$ is utilized as photosensitizer and three different water catalysts were employed, based on Ruthenium, Cobalt and Iridium. In all cases the solvent was a phosphate buffer at pH 7.0.

In Figure 3.7 it is shown a scheme of the oxygen evolution setup, the reactor is magnetically stirred and thermostated by a constant flow of water kept at 298K, from the side visible light is performed by a blue LED. It is important that the reaction is stirred so that there are not issues linked to diffusion limited reactions. The samples are taken by switching the gas chromatography valve from load position to inject position, the GC is enclosed in a helium-purged housing to prevent air leaking. Silicon septa is used as a connection, it is possible to deaerate by opening the helium inlet on the reactor and loosening one of the septa, while having the pump running. Before each measurement the system was deaerated for 15 minutes, after that a datapoint was collected every 5 seconds from the probe after tightening by screwing

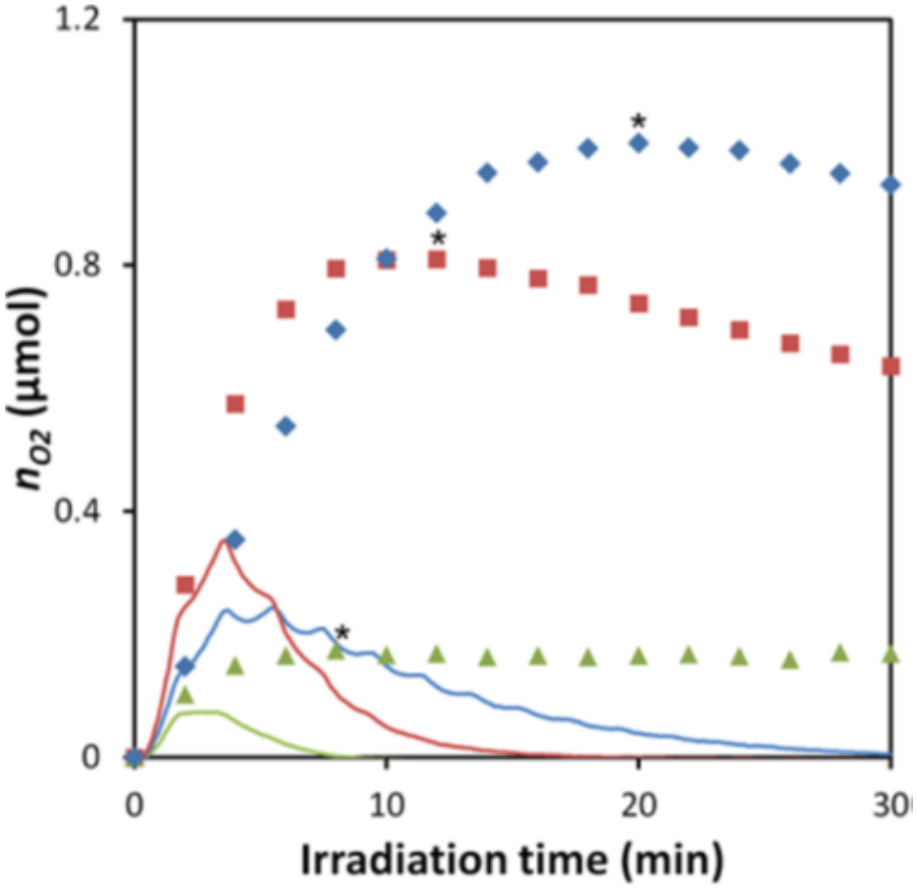

Figure 3.8: Photocatalytic dioxygen evolution as measured in solution by a fluorescence probe (solid curves) and in the headspace by GC (data points) [93].

on the loose septum and closing the helium inlet. Even so the air leakage could not be prevented probably because of the silicon septa moreover for every measurement by GC, 0.5 mL were consumed and replaced with the same amount of helium. Hence the the actual amount of oxygen produced is higher than the one measured [95]. The aim of validation is to reproduce the amount of dioxygen with the following reactor features: 5 μM of catalyst, 50 μM of photosensitizer and 2.5 mM of $Na_2S_2O_8$ in a volume of 5 mL of phosphate buffer (10 mM, pH = 7.0). In Figure 3.8 the outcome of the probe detection is shown, the solid curves represents the O_2 for the three catalysts, the red one is the homogeneous one while the data points are the values detected in the headspace by GC. The gas cromatography results are affected by the air leaking from the outside environment which is atmospheric ais then made of o_2 and N_2 Moreover the dips manifested in the signal are dued to GC measurement equilibium, the pump is turned on to fast equilibrate the dissolved O_2 and the headspace. The paper is further describing the process discovering that the electron transfer is the bottleneck of the reaction. Eventually the decomposition of PS^+ is taken into account, the values reports that the concentration of PS^+ reaches its

maximum after 100 seconds, subsequently it starts to decrease. [93, 95].

3.2.2 0D COMSOL model

As stated, the reactor is stirred thus the model has been studied with a 0D formulation, it means that all the points are equal without the presence of boundaries. Mainly in this section the three steps depicted in Figure 3.7 are simulated and then a parametric sweep is adopted to find the best fit for the experimental curves shown in Figure 3.8. The study is time dependent, with the time ranging from 0s to 900s (15 minutes), to simulate the decomposition and the different timing of the processes different ramp functions were implemented to be multiplied with the reactions rate.

The first step which is the oxidative quenching of PS^* by the electron acceptor (step 1, rate r_1) is simulated with the following reaction:

$$2\,\mathrm{PS^*} + 2\,\mathrm{H^+} + \mathrm{Q} \xrightarrow{\mathrm{k_1}} 2\,\mathrm{PS^+} + \mathrm{H_2Q} \tag{3.8}$$

Where PS is the photosensitizer, the couple formed by Q and H_2Q represent the hydroquinone which, for the purpose of this simulation, represents the electron acceptor to keep the balance of electrical charges.

The second step, the electron transfer from the catalyst to the excited photosensitizer (step 2, rate r_2) is represent by the following reaction:

$$2\,\mathrm{PS^+} + \mathrm{C} \xrightarrow{\mathrm{k_2}} 2\,\mathrm{PS} + \mathrm{C^{2+}} \tag{3.9}$$

Where C is the catalyst and C^{2+} is oxidized.

The third step is the catalytic oxidation of water (step 3, rate r_3) which is represented by the following reaction:

$$2\,\mathrm{C^{2+}} + 2\,\mathrm{H_2O} \xrightarrow{\mathrm{k_3}} \mathrm{O_2} + 4\,\mathrm{H^+} + 2\,\mathrm{C} \tag{3.10}$$

Besides other reactions were introduced to simulate the excited photosensitizer decomposition which is reported to be the rate limiting step of the process, and the oxygen exhausted. The decomposed photosensitizer species is referred to as PS_{esa} and takes part in the following reaction:

$$\mathrm{PS^+} \xrightarrow{\mathrm{k_{PS}}} \mathrm{PS_{esa}} \tag{3.11}$$

In the same way the O_2 leakage is delineated:

$$\mathrm{O_2} \xrightarrow{\mathrm{k_{O_2}}} \mathrm{O_{2leak}} \tag{3.12}$$

Furthermore two chemical equilibria were added to the model to reproduce the water dissociation and the phosphate buffer. Water has an amphoteric behaviour; it means that it can act both as an acid or a base in a solution. The water dissociation

Table 3.3: Initial input values for the validation 0D COMSOL model.

Species -	Concentration mol/m^3
Photosensitizer (PS)	50 x 10^{-3}
Photosensitizer (PS^+)	0
Catalyst (C)	5 x 10^{-3}
Catalyst (C^{2+})	0
Electron relay (Q)	2.5
Electron relay (H_2Q)	0
Dioxygen (O_2)	0
Hydrogen (H^+)	1 x 10^{-4}
Hydroxide (OH^-)	1 x 10^{-4}
Water (H_2O)	55.5
Phosphate buffer (H_3PO_4)	10
Phosphate buffer ($H_2PO_4^-$)	75 x 10^4
Phosphate buffer (HPO_4^{2-})	46.5 x 10^4
Phosphate buffer (PO_4^{3-})	1.116
Decomposed Photosensitizer (PS_{esa})	0
Oxygen dump (O_{2leak})	0

takes place in pure water or in aqueous solutions when a molecule of H_2O becomes hydroxide OH^-. The equilibrium can be merely written as:

$$H_2O \xleftrightarrow{K_{H_2O}} OH^- + H^+ \tag{3.13}$$

The water is a really weak electrolyte which means it does not completely dissociate in aqueous solutions, this is proven by the equilibrium constant that in diluted solutions can be written as:

$$K_{H_2O} = \frac{[OH^-][H^+]}{[H_2O]} \tag{3.14}$$

The constant is equal to 1.8 x 10^{-16} when the concentrations are taken in mol/L, in this case for the thesis applications it will be written in mol/m^3, it means that K_{H_2O} = 1.8 x 10^{-13}. Given the low value of the equilibrium constant the reaction does not have a significant impact on the solutions.

For what concerns the buffer, the phosphate buffer is simulated as an equilibrium between the following reactions:

$$H_3PO_4 \xleftrightarrow{K_{ph1}} H^+ + H_2PO_4{}^- \tag{3.15}$$

$$H_2PO_4{}^- \xleftrightarrow{K_{ph2}} H^+ + HPO_4{}^{2-} \tag{3.16}$$

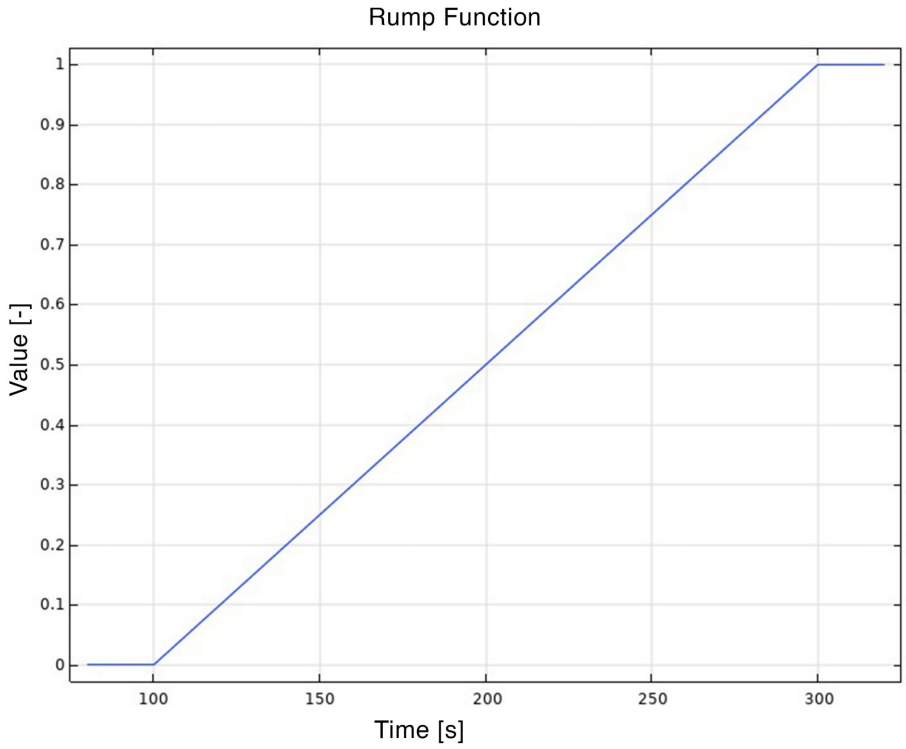

Figure 3.9: Rump function: located at time step 100 with a slope of 1/200 and cutoff equals to 1.

$$HPO_4{}^{2-} \overset{K_{ph3}}{\longleftrightarrow} H^+ + PO_4{}^{3-} \tag{3.17}$$

The main role of the buffer is to keep the pH of the solution stable over the time. The values for the equilibrium constants are taken from the previous simulation.

Over the simulation different parametric sweeps have been carried out to find the values of k_1, k_2 and k_3 in addition to the decomposition rates. For the last couple there are the difficulties to match the timing because the decomposition is not active since the beginning, in order to reproduce the phenomena two different rump functions were employed, in Figure 3.9 the rump function for the decomposed photosensitizer specie is shown. The reaction starts after 100 seconds (referred to the time given in the subsection 3.2.1) and reach its maximum after 300 seconds. While for the O_{2esa} the reaction starts after 200 seconds and reach it maximum after 500 seconds, the cutoff in this case is equal to 2 to cope the oxygen measurements. Regarding the other rates of the model a wide range of parameters have been tried with a step of 10 going from 1 x 10^{-3} to 1 x 10^{10}.

3.2.3 Validation results

In this subsection the results of the simulations are displayed: switching the parameters the model shows that the rate k_3 is not the rate determining step, in fact

Table 3.4: Output rates from the COMSOL model.

Rate constant	Value
k_1	1 x 10^7
k_2	1 x 10^4
k_3	0.1
k_{PS}	1 x 10^{-2}
k_{O2}	1 x 10^{-2}

keeping the other parameters fixed the oxygen production is not affected through all the magnitude changes, the only difference reported is a subtle discrepancy in the catalyst behavior which can undergo rapid changes because of calculation errors. On the other hand, k_1 and k_2 are more difficult to be identified since only few results are acceptable, after many simulations k_2 has been selected as the main constant it means that the second step rate is the one which is determining the order of magnitude of oxygen production. While k_1 can increase or decrease the oxygen yield but if too high can give instability issues to the photosensitizer which tends be consumed instantly, with the values that can fluctuate below the zero, which is obviously not realistic. In table 3.4 the final reaction constants are summarized. Moreover the

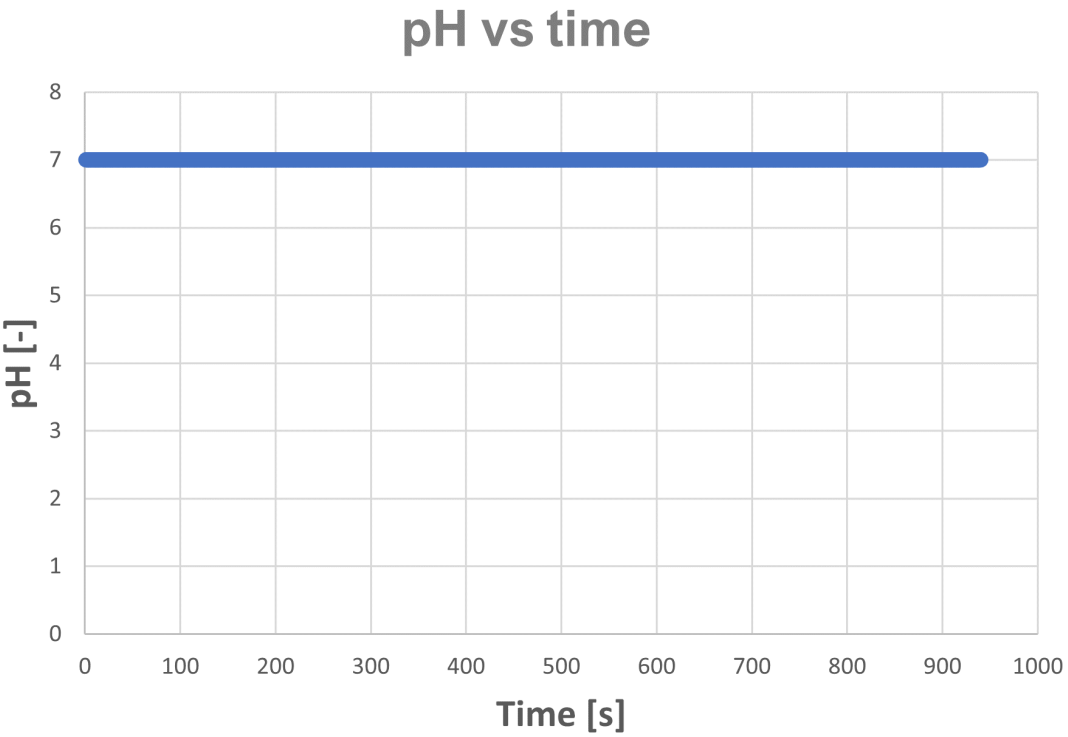

Figure 3.10: pH trend in the validation model versus time.

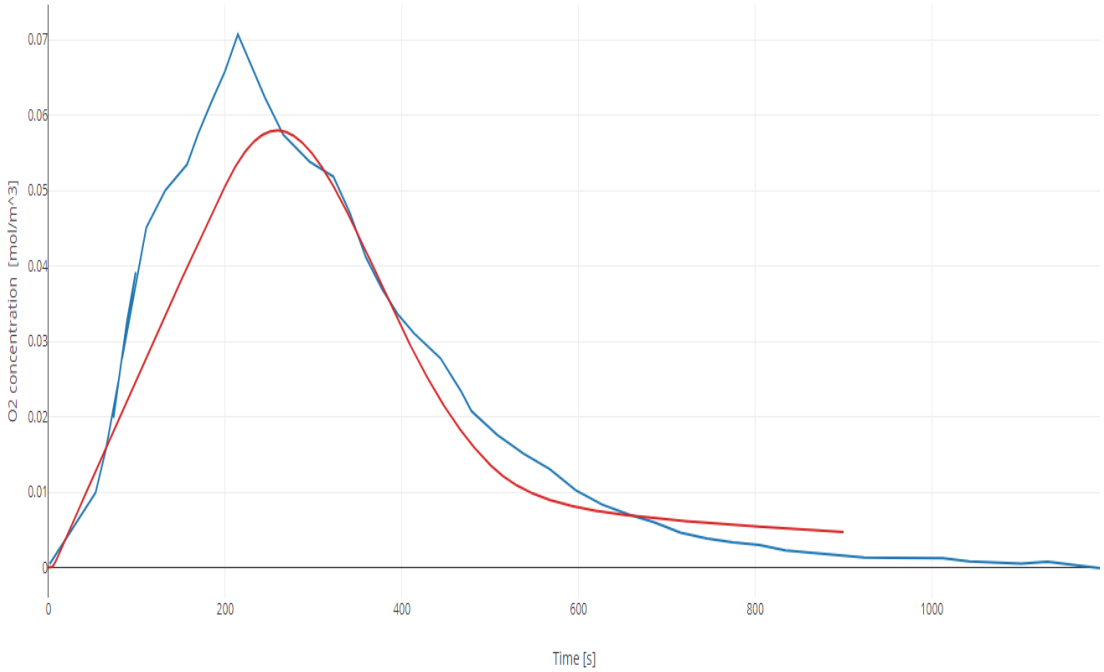

Figure 3.11: O_2 concentration evolution respct to time, blue solid line: paper, red solid line: COMSOL.

graphic results show that the simulated phosphate buffer is reproducing the results in a good manner, in Figure 3.10 the pH values respect to time is plotted with the abscissa on a log scale. From the graphic it is possible to notice that the H^+ molar concentration display a strong stability with little fluctuation around 100 seconds which are not even spotted in the figure, they could be negligible variation dued to the response of the decomposition PS^+ process.

Eventually in Figure 3.11 the concentration of oxygen in mol/m^3 vs the time expressed in seconds is displayed. The paper's values (blue line) are given in terms of moles therefore they have been divided by the reactor volume. On the other hand the red line comes from COMSOL simulation, there are differences in the maximum level between the two lines, this is due to the experimental procedures which, as stated, depict dips because of GC measurement and there is no interest in reproducing that response. Altogether the simulation manage to represent well the dioxygen evolution over time with a difference on the peak and with a subtle lag, the shown curve is the best fit that has been found keeping into account all the the parameters at stake.

3.3 1D model of a surfactant monolayer

In the following paragraphs the model of a surfactant monolayer will be discussed. The post processing of the results was performed with COMSOL itself or Seaborn, a Python data visualization library based on matplotlib. In this section it is given most of the importance to the parametric search to understand how the constants influence the model especially in those case where there is a lack of experimental data.

3.3.1 Model description

In Figure 3.12 the concept of the soap film is depicted: the elements with a green head and a yellow tail are the surfactant molecules, with the head which dispose at the border of the film coloured in light blue, together with the particles that take part in the simulation carried out in this thesis. The model which has been

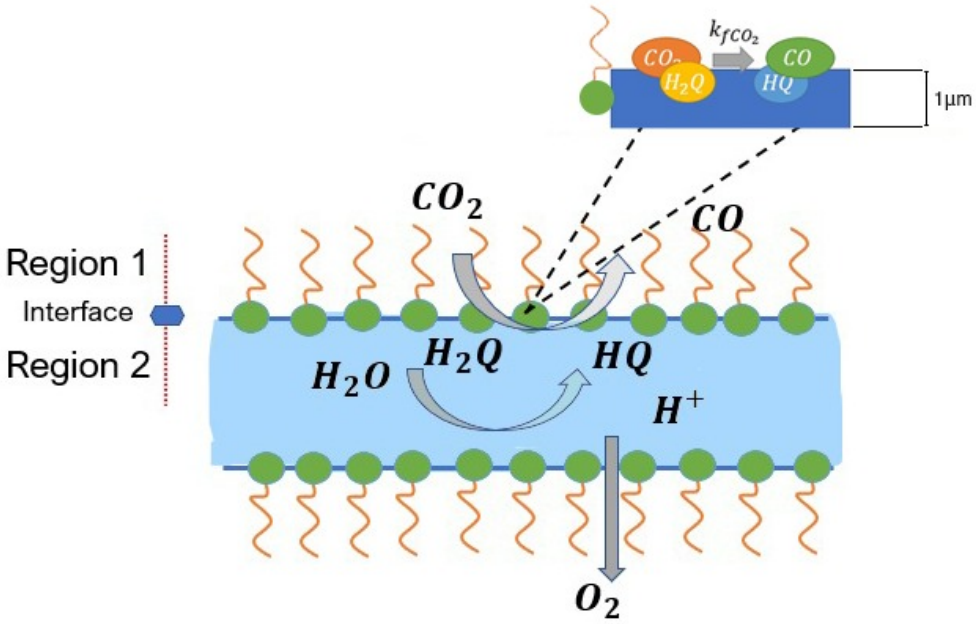

Figure 3.12: Conceptual representation of the soap film with the species involved.

implemented is illustrated with the red dotted line on the left, the 1D line simulated in COMSOL can be divided in 2 different regions: Region 1 which is a gaseous region while Region 2 is a liquid domain, the species can adsorb at the interface between the gas and the liquid phase. On the top right the main reaction is shown, the reaction is mainly between adsorbed species, so it takes place at the interface. In Table 3.5 it is possible to the sizes of the species involved displayed with the molecular diameters

Table 3.5: Molecular diameters of the species.

Species -	Molecular diameter nm	Reference -
CO_2	0.334	[96]
CO	0.319	[97]
Electron relays	0.750	[98]
H_2O	0.275	[99]
Tail C12E6	0.975	[100, 101]
Head C12E6	0.950	[100, 101]

value with their reference, for what concerns the electron relays it has been taken the diameter linked to the phenol molecule which is quite similar to the hydroquinone, the surfactant is subdivided in head and tail, in this case it may be more appropriate to talk about thickness instead of diameter. The model is simplified respect to the actual reactions which actually take place, but it can be a good starting point to gain knowledge and to understand the order of magnitude of the different players. By hypothesis gaseous species can permeate the membrane while what is in solution inside the monolayer can not go in the gas region. All the species take part to reactions at the interface. The main reactions simulated are basically of two kinds, an overall reaction which is responsible for the production of the carbon monoxide through carbon dioxide reduction, and adsorption and desorption equilibria. The reduction process is represented by the following reaction:

$$\mathrm{CO_2}^{\mathrm{ads}} + 2\,\mathrm{H_2Q} \xrightarrow{\mathrm{k}_f} \mathrm{CO}^{\mathrm{ads}} + 2\,\mathrm{HQ} + \mathrm{H_2O} \tag{3.18}$$

The reaction is irreversible, and it is regulated by the forward reaction rate indicated as k_f. The value of the rate it is unknown and it is a important value because it gives an idea of the overall velocity of the process. The adsorption reaction can be generally defined as:

$$\mathrm{Species} \xleftarrow[\mathrm{k_{des}}]{\mathrm{k_{ads}}} \mathrm{Species}^{\mathrm{ads}} \tag{3.19}$$

specifically the reactions implemented are:

$$\mathrm{CO_2} \xleftarrow[\mathrm{k_{des}}CO_2]{\mathrm{k_{ads}}CO_2} \mathrm{CO_2}^{\mathrm{ads}} \tag{3.20}$$

$$\mathrm{CO} \xleftarrow[\mathrm{k_{des}}CO]{\mathrm{k_{ads}}CO} \mathrm{CO}^{\mathrm{ads}} \tag{3.21}$$

$$\mathrm{H_2Q} \xleftarrow[\mathrm{k_{des}}H_2Q]{\mathrm{k_{ads}}H_2Q} \mathrm{H_2Q}^{\mathrm{ads}} \tag{3.22}$$

$$\mathrm{HQ} \xleftarrow[\mathrm{k_{des}}HQ]{\mathrm{k_{ads}}HQ} \mathrm{HQ}^{\mathrm{ads}} \tag{3.23}$$

For the carbon dioxide reduction simulated the hydroquinone is an electron donor.

3.3.2 Initial values

Since a innovative subject is treated, the initial values assigned to the variables become very important to reach satisfactory results. They have not always been found values that start exactly from the same assumptions of the simulated model, in this case it has been taken in consideration the data in the more similar situation. At first it is relevant to define the initial concentrations for the species in the different domains. The initial values for the CO_2 is calculated assuming that the outer atmosphere of the monolayer is equivalent to 1 atmosphere of carbon dioxide only, thus it is evaluated with:

$$c_{CO_2}^{reg1} = \frac{P}{RT} \tag{3.24}$$

With T equals to 293.15 K, P corresponding to 1atm and R = 8.206 x 10^{-5} $\frac{m^3 atm}{K mol}$.For what concerns the CO it was taken a deliberately low chosen value of 1 x 10^{-10}. To determine the initial concentration values of CO_2 and CO the Henry's law is utilized:

$$c_{CO_2}^{reg2} = c_{CO_2}^{reg1} * H_{CO_2} \tag{3.25}$$

$$c_{CO}^{reg2} = c_{CO}^{reg1} * H_{CO} \tag{3.26}$$

The initial concentrations of the H_2Q and HQ is defined by the author as 100 mol/m^3 and 1 x 10^{-10}, respectively. In Table 3.6 the concentration input values are summarized.

Table 3.6: Input concentration values and Henry's constants.

Species -	H -	c^{reg1} $\frac{mol}{m^3}$	c^{reg2} $\frac{mol}{m^3}$
CO_2	0.952	41.571	39.575
CO	2.56 x 10^{-2}	1 x 10^{-10}	2.56 x 10^{-12}
H_2Q	-	-	100.0
HQ	-	-	1 x 10^{-10}
H_2O	-	-	55555

Among the values that the model requires in input there are the Langmuir parameters, mainly the surface excess at saturated coverage Γ_s and K_{eq} which is the equilibrium constant, the ratio between the adsorption k_{ads} and desorption k_{des} constants. The input Langmuirian values for what concerns CO_2, CO, O_2 and H_2 are taken from the paper of Massoudi and King [102]. This work reports many results obtained in measuring interfacial tension as a function of pressure for a wide variety of gas-water systems at 25°C. the isotherms are derived beginning from the interfacial tension which follows, at constant temperature the following polynomial:

$$\gamma = \gamma_0 + BP + CP^2 + DP^3 \tag{3.27}$$

Table 3.7: Interfacial tension parameters as function of pressure for pure water with various gases at 25°C.

Species	B	C	D
-	$\frac{dyn}{cm\ \ atm}$	$\frac{dyn}{cm\ \ atm^2}$	$\frac{dyn}{cm\ \ atm^3}$
CO_2	-0.7789	+0.00543	-0.000042
CO	-0.1041	+0.000239	-

Where γ_0 is equal to 71.98 dyn/cm. In Table 3.7 the paper coefficients are reported.

The surface excess for the various gases is calculated according to the convention which pones the Gibbs plane in order that the surface excess of water is equal to zero using the following equation which is the same equation described in equation 1.20 with R written as $R = Zk$:

$$\left(\frac{\delta\gamma}{\delta P}\right)_T = -\Gamma\left(\frac{ZkT}{P}\right) \tag{3.28}$$

Where there is the partial derivative of interfacial tension respect to pressure with fixed temperature equals to the surface excess multplied to Z which is the compressibility factor, k that defines the Boltzmann constant and T designates the temperature, divided by the pressure.

The experimental environment seems to be similar to the one intended to be simulated, gas adsorbing at a liquid aqueous interface, so the results obtained, given in Table 3.8, are considered reliable. Regarding the electron transfers the values reported in Table 3.8 come from the article by Yıldız and coworkers [81], the reason why this paper is taken as a reference are outilend in the previous chapter (Chapter 2). As stated The aim of the paper is to find cost-effective and efficient ways to adsorb hydroquinones and benzoic acid from aqueous solutions, the study utilizes natural clay bentonites which usually form from weathering of volcanic ash in seawater [103]. The particularity is that ion-exchange can deeply modify the surface properties. When organic cations (cationic surfactants) are applied superficially, they tend to occupy the sites of bentonite clay making the surface switch from hydrophilic to hydrophobic. Experimentally in the cited study, organobentonites containing different organic cations (octadecyltrimethyle ammonium, ODTMA, hexadecyltrimethylammonium, HDTMA) were synthesized. The adsorbed amount of benzoic acid and hydroquinone were calculated as the difference between the added value compared to what remains at different temperatures and pH values in order to plot adsorption isotherms. The results showed that decreasing uptake with increasing pH while an increasing uptake with increasing temperatures as shown in Figure 3.13. The considered values are those referred to a temperature of 25 °C in neutral pH for ODTMA, numerically the surface excess is equal to 12.0482 mg/g and the equilibrium adsorption constant equivalent to 0.0090 L/mg. The values are

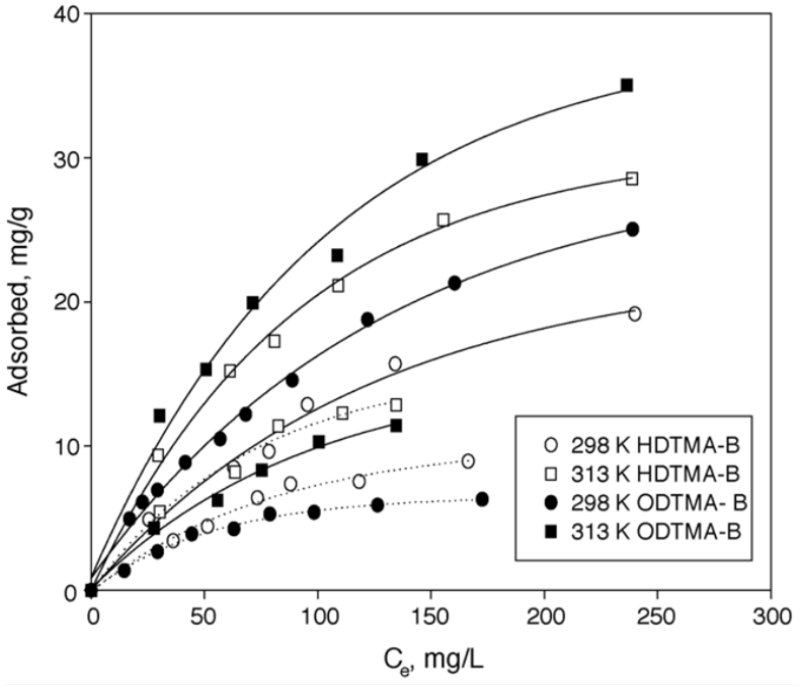

Figure 3.13: Sorption of benzoic acid and hydroquinone by ODTMA-B and HDTMA-B at various temperatures (line; benzoic acid, dotted; hydroquinone) [81].

converted into the desired units of measurement through the bentonite BET 28.92 m^2/g and the hydroquinone molecular mass 1.10 x 10^5 mg/mol.

Table 3.8: Input Langmuir parameters.

Species -	K_{eq} $\frac{m^3}{mol}$	Γ_s $\frac{mol}{m^2}$
CO_2	1.26 x 10^{-4}	5.24 x 10^{-5}
CO	1.60 x 10^{-4}	6.59 x 10^{-6}
H_2Q	0.991	3.78 x 10^{-6}
HQ	0.991	3.78 x 10^{-6}

With the Langmuirian parameters and the initial concentrations the surface reaction physics is able to work. Nevertheless some other parameters are needed as a input to simulate the transport of diluted species in the liquid and solid domains, the diffusion process is modeled using the Fick's law while the flux at the interface that can permeate the membrane is implemented using the surfactant monolayer permeability. The flux calculated in $\frac{mol}{m^2 s}$, coming from the gaseous region to the liquid region is modeled as:

$$J_{species}^{gas->liq} = k_{species}^{perm} * \left(c_{species}^{reg1} - \frac{c_{species}^{reg2}}{H_{species}} \right) \tag{3.29}$$

Table 3.9: Input values for species transport.

Species -	Diff. coeff. in gas m^2/s	Diff. coeff. in liquid m^2/s	Surf. monolayer perm. m/s
CO_2	1.35 x 10^{-5}	1.67 x 10^{-9}	0.164
CO	1.90 x 10^{-5}	2.03 x 10^{-9}	0.593
H_2Q	-	1 x 10^{-10}	-
HQ	-	1 x 10^{-10}	-
H_2O	-	2 x 10^{-9}	-

On the other hand the flux coming from the liquid domain to the gaseous one follows the next equation:

$$J_{species}^{liq->gas} = k_{species}^{perm} * \left(c_{species}^{reg2} - H_{species} * c_{species}^{reg1}\right) \tag{3.30}$$

In Table 3.9 it is possible to see the input values regarding the transport of diluted species with the diffusion coefficiente both in the gas and liquid domain. The values in Table 3.9 and the Henry's constant in Table 3.6 are taken from this reference [56].

Eventually the value of the reaction rate k_f represented in equation 3.18 is set equal to 1 x 10^{12}.

3.3.3 Time and space distributions

Given an overview of the model and the input values provided to the COMSOL model, this subsection deals with the first simulation carried out, a time-dependent study with the time ranging from 0 to 28,800 seconds (8 hours) with a time-step of 100 seconds.

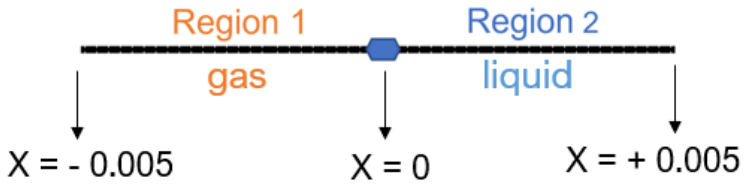

Figure 3.14: 1D model mesh graphical representation.

The 1D model mesh is shown in Figure 3.14, it is utilized for this simulation and it will be always the same from now on. The line length is 0.01 m with the origin of the axes located at the interface between the liquid and gaseous domain. The mesh is predefined, it is extremely fine with a maximum element size equals to 6.75 x 10^{-5} and a maximum element growth rate of 1.05. This study is useful to understand how to system behaves through the time, the results reported in

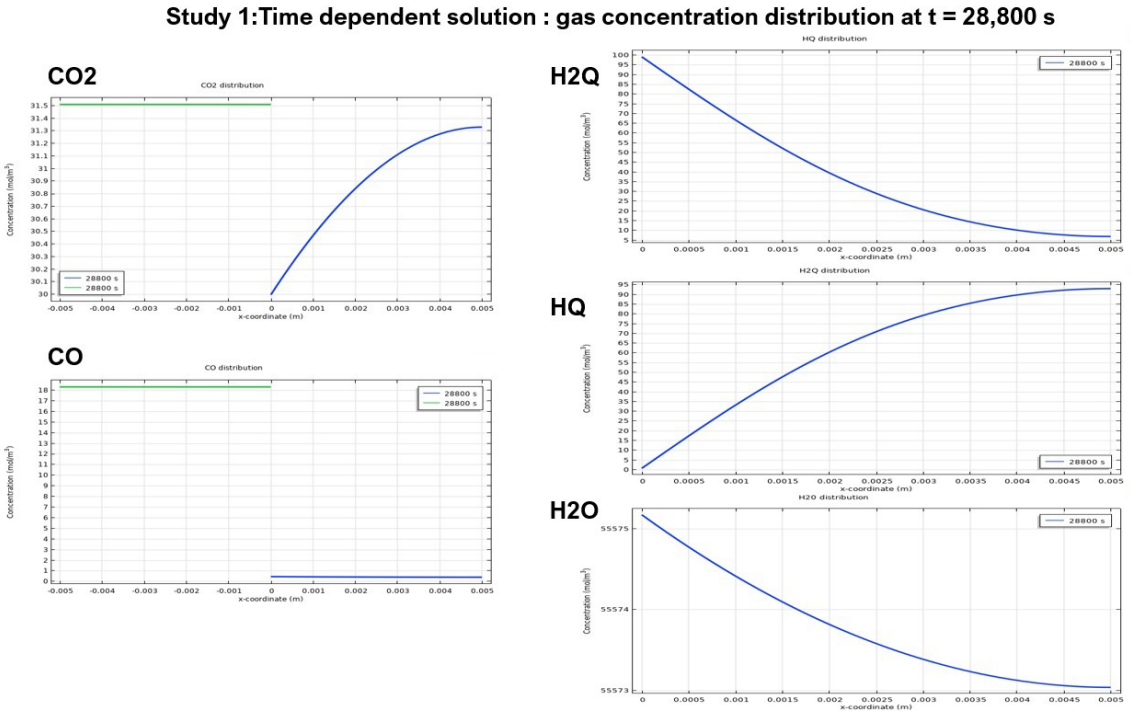

Figure 3.15: Species concentration space distribution after 8 hours.

Figure 3.15 shows the space distribution at the last time-step (after 8 hours). The

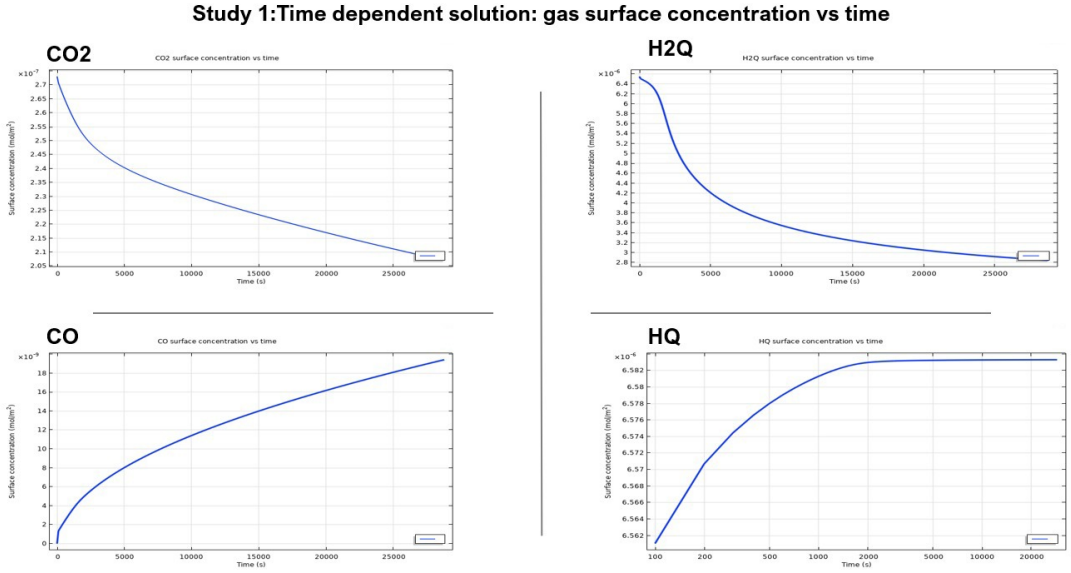

Figure 3.16: Species interfacial surface concentration evolution in time.

CO_2 concentration in the gaseous domain is constant with a value of 31.5 mol/m^3 which corresponds to 1atm pressure, in the liquid the space distribution ranges from around 30 mol/m^3 close to the interface to increase until 31.3 mol/m^3, it is a reasonable trend because the carbon dioxide is reduced at the interface to produce carbon monoxide.The CO displays a constant value in the gaseous region of around 18 mol/m^3, in the liquid region the concentration amount is lower because it is linked to the Henry's law, the different behaviors in the gas and liquid phase is a result of the different values of the diffusion coefficient. The hydroquinone concentration is shown only in Region 2, the H_2Q is almost equals to zero at the interface with a maximum value of around 95 mol/m^3 at the endpoint, logically the HQ concentration is behaving in the opposite way, displaying maximum at the interface of approximately mol/m^3 and a minimum of 5 mol/m^3. This first output presented the electron transfer concentration attaining the value of zero, different simulations have been carried out to better describe the trend, it came out that the value approaches significantly to zero after 52 minutes and 30 seconds not due to a lack of it but limited by diffusion instead (the python codes to develop an animated simulation with the hydroquinone concentration can be found in Appendix A). It is clear from Figure 3.15 that the concentration at the endpoint is still similar to the initial value even after 8 hours. Moreover, in Figure 3.16 the surface concentration, measured in mol/m^2, is represented at the interface respect to time, the reagent CO_2 and H_2Q decrease over time while the products, CO and HQ increase their concentration at liquid-gas interface. The carbon dioxide ranges from 2.8 x 10^{-7} to 2 x 10^{-7}, the H_2Q starts from a value of 6.4 x 10^{-6} to reach 2.8 x 10^{-6}. The model reported the total surface concentration in the two regions follow the interface trend

for the four species. For what concerns the products, numerically the CO surface interfacial concentration goes from 0 to around 20 10^{-9}, while the HQ ranges from 0 to 6.6 10^{-6}. The species interface surface concentration values show an expected behaviour with the two couples, CO_2/CO and H_2Q/HQ moving accordingly. In all the simulations the principle of mass has been checked and respected, the difference between the values calculated at the time equals to zero compared to all the other time steps were monitored, this difference was never reported to be higher than 5 x 10^{-11} which is considered within the calculation errors.

3.3.4 Forward reaction rate tuning

The forward reaction rate is one of the most relevant parameters because it gives an idea on the carbon monoxide production. If the reagents are present, the value of the general reaction has the task of regulating the speed of generation, therefore the yield of the process. As already mentioned in the previous sections the starting value is 1 x 10^{12}, despite most of the other parameters, this number is taken as a guessing value without a supporting article or a reference, it was a number that was considered acceptable before the very first simulation because capable of producing a change. After this first guess the parameters underwent a parametric sweep where the value is tested by modifying the order of magnitude, over the simulation the number varies from 1 x 10^{9} to 1 x 10^{15}. The aim of the simulation is having a deeper understanding on how the value is affecting the number of moles of the different species in the two domains and at the interface, the result considered is a line average calculated in mol/m^2 for the gaseous and the liquid region in order to compare the numbers with the superficial concentration calculated for x equals zero. The graphs reported hereinafter are taken in the half of simulation, it means after 14,400 seconds (4 hours), this decision about time-step is simply an author decision. The chosen color are: green for the gaseous region, blue for the liquid region and red for the concentration at the interface. The y axis is in linear scale while the

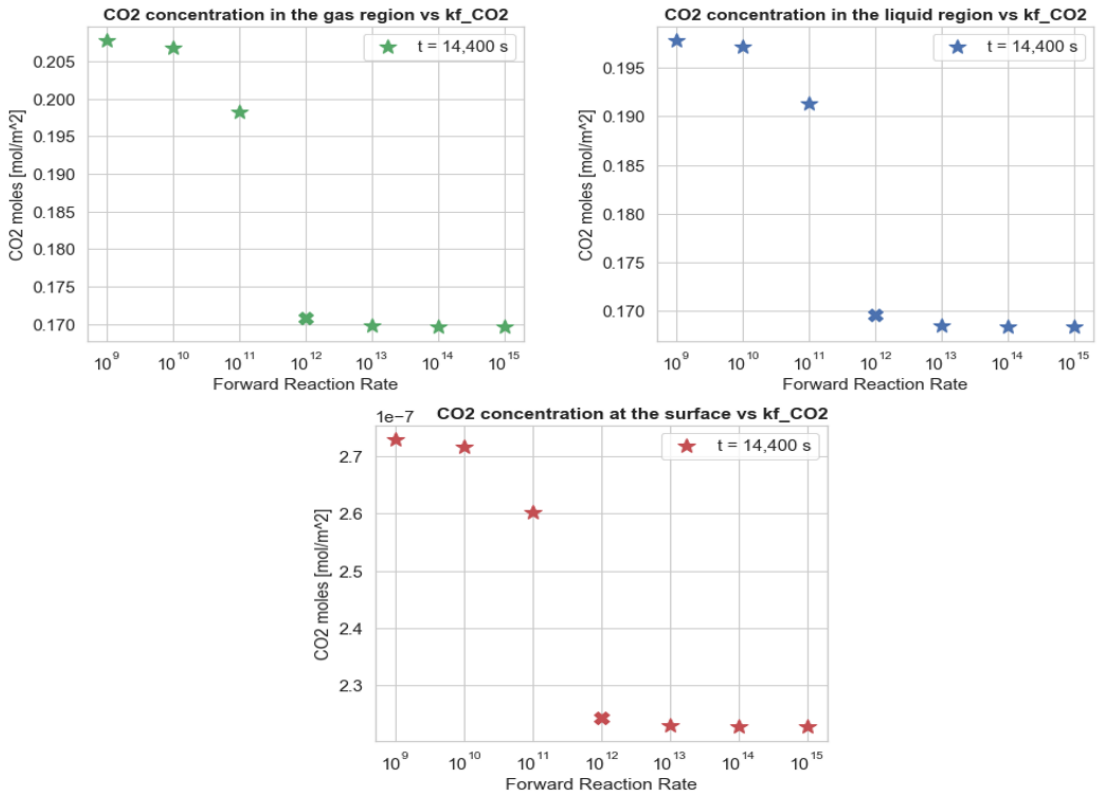

Figure 3.17: CO_2 line average and surface concentration vs k_f.

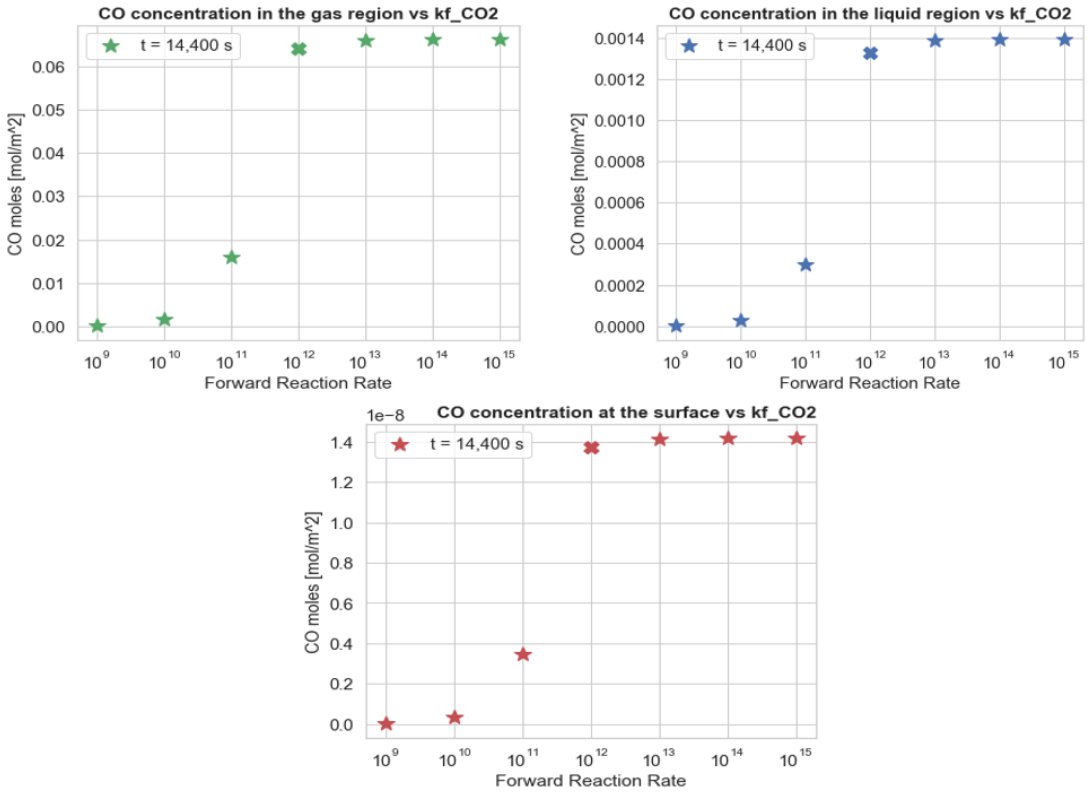

Figure 3.18: CO line average and surface concentration vs k_f.

x axis is in logarithmic scale. In Figure 3.17 the three charts for CO_2 are given, as could be expected the main trend is generally downwards, the carbon dioxide concentration is decreasing as the number of the forward reaction rate is increasing. It is possible to spot a "switching value", in fact from 1 x 10_{10} to 1 x 10^{12} the concentration of CO_2 undergoes a relative reduction of 15% while before and after 1 x 10^{11} the values seem to be stable, especially for higher reaction rate numerical values. The concentration of CO behaves accordingly, as the CO_2 is decreasing the CO is increasing, in Figure 3.18 the concentration of carbon monoxide is depicted. Still in this case the value of 1 x 10^{11} is a switching value but numerically the yield is greater of six times in the gas region and it reaches higher relative differences in the liquid region and at the surface interface. This graph shows once again the importance of the forward reaction rate, the amount of CO produced is changing its order of magnitude changing the rate of the reaction. Eventually in Figure 3.19 it is possible to see the concentration of H_2Q and HQ in the liquid region and at the interface. The same dualism reported before for the couple CO_2/CO exists for the hydroquinone and its product of the redox reaction, still the the trend represented in Figure 3.17 and Figure 3.18 is respected with the subtle difference that the HQ interfacial concentration corresponding to the x value of 1 x 10^{11} has already reached the "stable" part of the graphic. After this graphical results a simulation has been

carried out which shows that hourly the concentration values smaller than 1 x 10^{11} does not change over time, they are similar after the first hour while the larger values are continuously decreasing.

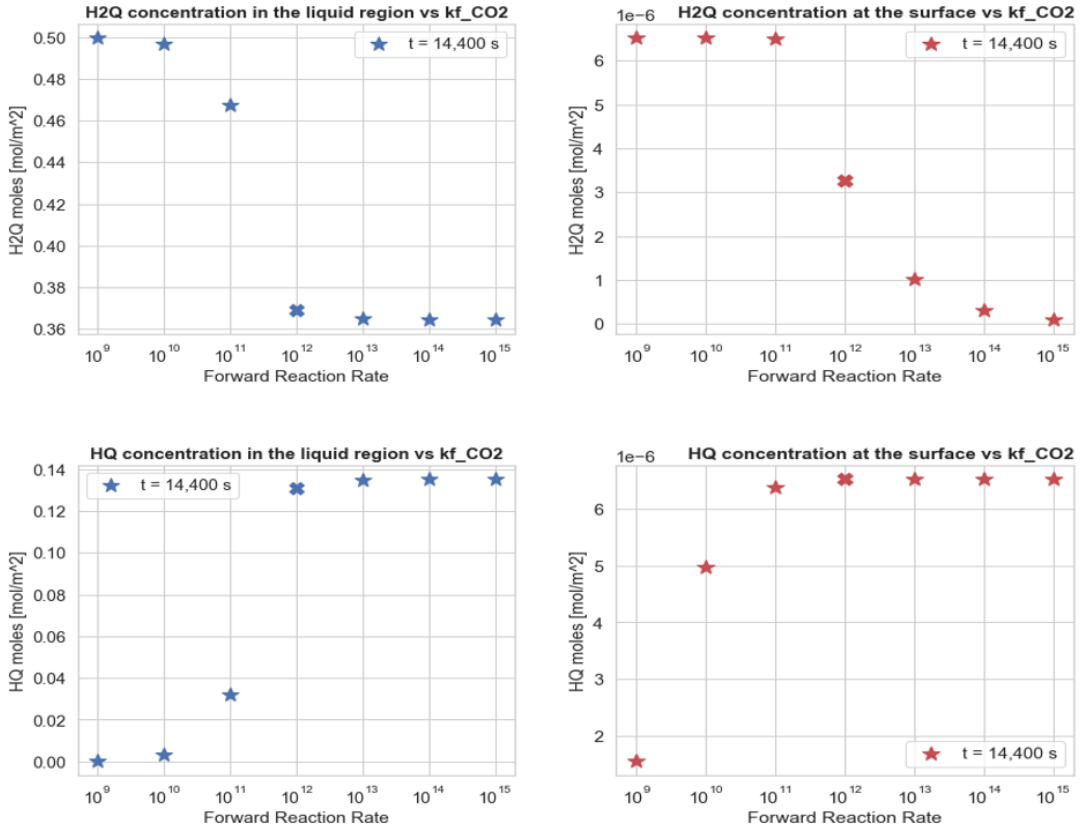

Figure 3.19: H_2Q and HQ line average and surface concentration vs k_f.

3.3.5 Adsorption phenomenons parameters

Together with the forward reaction rate the adsorption equilibrium constants are the factors that regulate the whole production of carbon monoxide. Over these simulation again a parametric sweep has been applied but in this case on two parameters: the surface excess and the rate of adsorption or desorption. Generally the simulations are of two kinds, in the first case the equilibium adsorption constant of the different species is kept fixed with the values of k_{ads} and k_{des} free to move accordingly. In the second case the desorption constant is set while the adsorption constant is changing and consequently the K_{eq}. All species have been taken into account except water for the computation. The results are given with the shape of heatmaps with the surface excess Γ_s on the x-axis and the adsorption reaction rate on the y-axis, the values can be read as a matrix with coordinates (Γ_s,k_{ads}). The values considered are the sum of concentrations in the liquid and gaseous domains

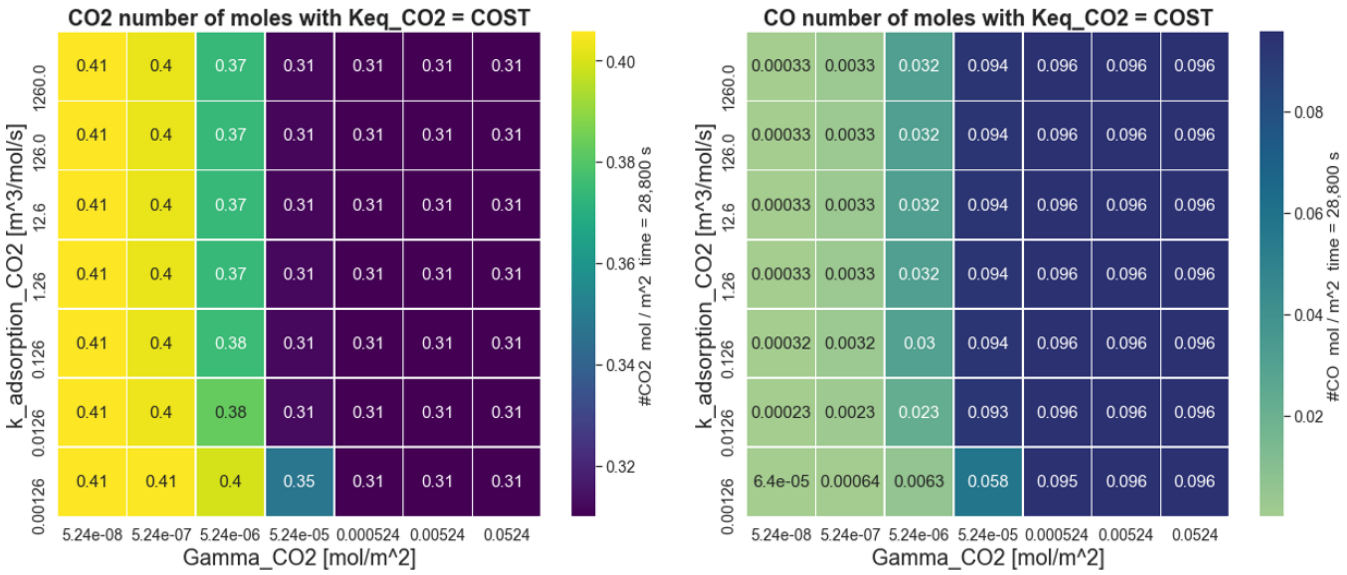

Figure 3.20: Heatmaps for CO_2 and CO when $K_{eq}^{CO_2}$ is fixed.

(line average) with the superficial concentration at the interface at the last time step, 28,800 s (8 hours). In Figure 3.20 the first couple of heatmaps is depicted, in this case the equilibrium constant of CO_2 is fixed, and it is possible to see the concentration values for CO_2 on the left and CO on the right. The tile in the middle of the square matrix is the actual value coming from the input values discussed in the other sections while the surrounding numbers represents all the possible combinations. For both cases it is possible to see that for the $K_{eq}^{CO_2}$ fixed the number of

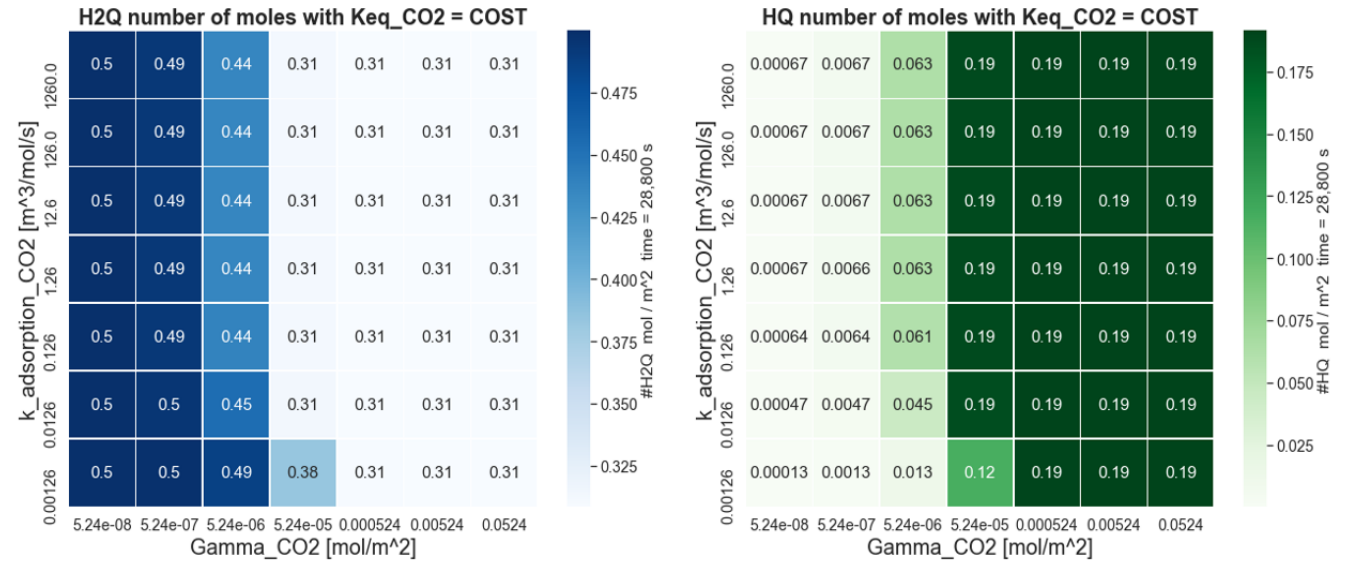

Figure 3.21: Heatmaps for H_2Q and HQ when $K_{eq}^{CO_2}$ is fixed.

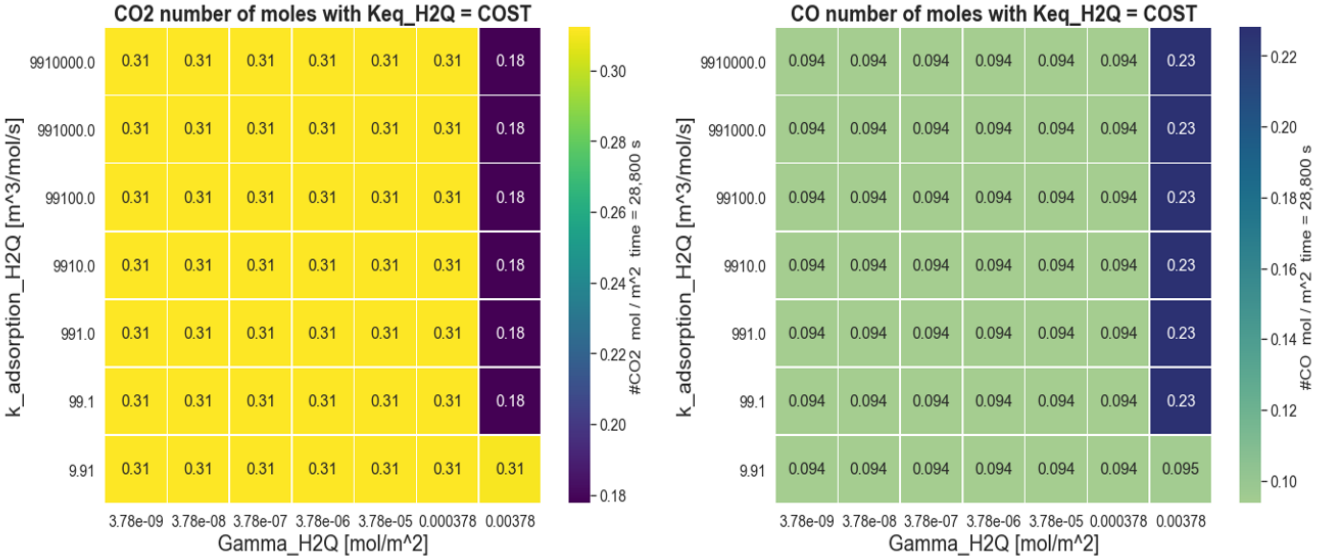

Figure 3.22: Heatmaps for CO_2 and CO when $K_{eq}^{H_2Q}$ is fixed.

active sites is more influential than the speed of adsorption, the numerical outputs are changing across the columns but not on the rows. In Figure 3.21 the other two maps for the electron transfer are displayed, the trend is similar to the one of carbon compounds, the parameter which is mainly affecting the calculation is $\Gamma_s^{CO_2}$, nevertheless there is a limit, for the largest number of active sites there are not changes in the concentration values. In Figure 3.22 and in Figure 3.23 the next simulation is depicted, this time the species which undergoes the parametrics weep

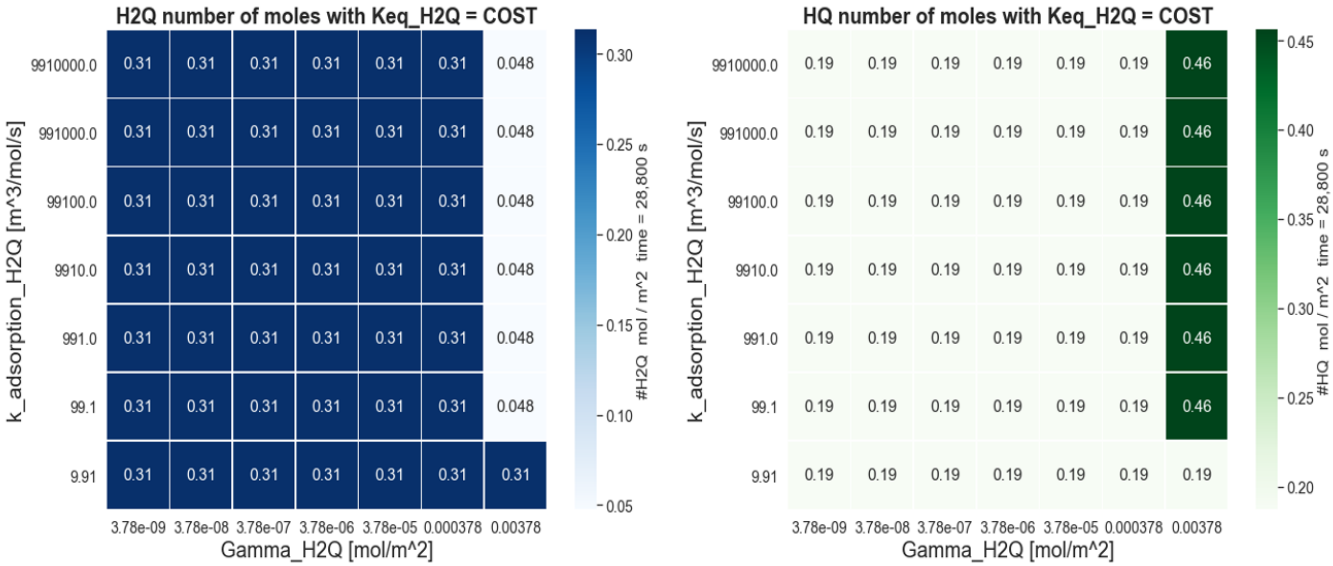

Figure 3.23: Heatmaps for H_2Q and HQ when $K_{eq}^{H_2Q}$ is fixed.

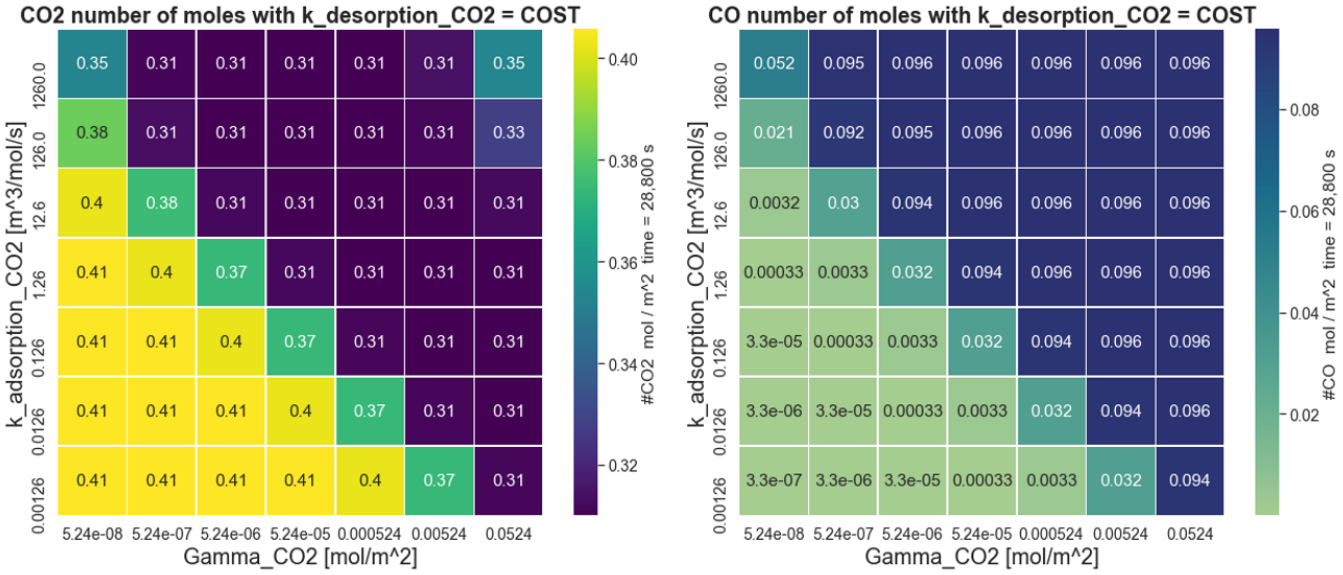

Figure 3.24: Heatmaps for CO_2 and CO when $k_{des}^{CO_2}$ is fixed.

is the hydroquinone. In this case variations are spotted for the highest value of $\Gamma_s^{H_2Q}$ only which is likely unrealistic. In Figure 3.24 the heatmap belonging to the second type of simulations is depicted, in this case the desorption constant is fixed thus the uptake is faster. Despite the previous simulation the velocity of adsorption of the specie at the interface become more important, producing a change over the rows. Both for CO_2, CO and H_2Q, HQ in Figure 3.25 it is possible to identify

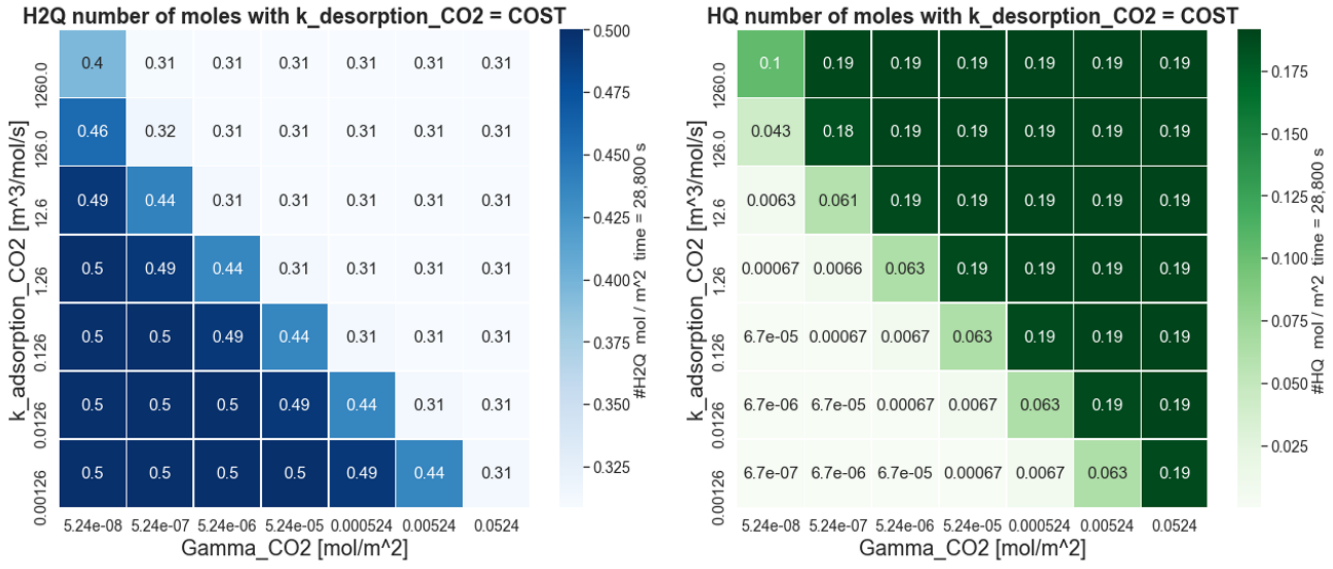

Figure 3.25: Heatmaps for H_2Q and HQ when $k_{des}^{CO_2}$ is fixed.

two regions inside the maps divided by a diagonal which points out the importance of both $\Gamma_s^{CO_2}$ and $k_{ads}^{CO_2}$. At the top corner of the carbon dioxide heatmap some unexpected values can be found, there is no physical explanation for these numbers so the value can be attributed to a computational error due to the high rate of CO_2 adsorption, further simulations can be useful to better understand this behaviour. Generally from these results it can be said that a wide combination of parameters is actually giving the same final output. In Figure 3.26 and Figure 3.27 the heatmaps

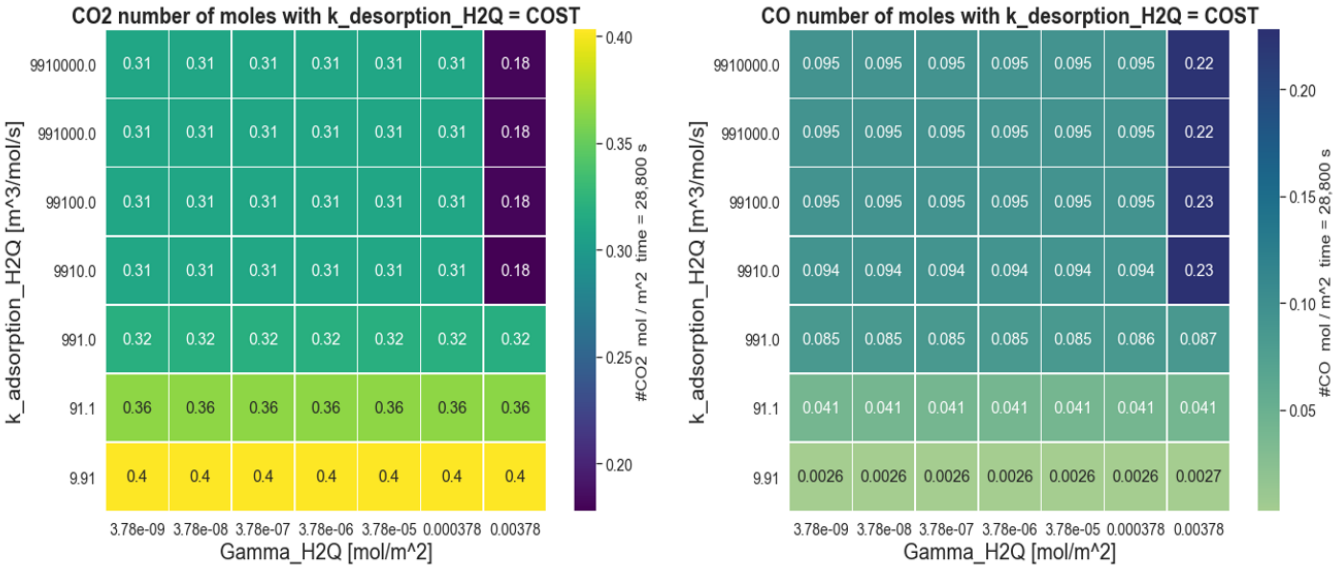

Figure 3.26: Heatmaps for CO_2 and CO when $k_{des}^{H_2Q}$ is fixed.

with the fixed value of the desorption constant of H_2Q are displayed. In all four graphs there is no variation on the columns but only through the rows with a difference in the concentration values corresponding to the highest numbers of number of adsorption sites, as already hinted probably this values are not physical thus the author decided not to dig deeper into the matter. The parametric sweeps carried out for CO and HQ are not reported here because with these two species all the maps were showing the same values of the central square tile of the other heatmaps. Nevertheless this is a result for the reason that these parameters does not affect the global process making further studies easier since two parameters may be excluded in deeper researches. The concentration values were taken at the last time step in order to generally understand more or less the order of magnitude of the amount of moles present inside the surfactant monolayer. With a few exceptions the numerical output seem reasonable: with a meaning in the production of reaction product and a sense in the consumption of reagents, the COMSOL model is then able to reproduce the phenomenon. The aim of these simulation in particular were to understand the influence of the langmuirian parameters to have an outlook on carbon monoxide production, this subsection together with subsection 2.2.4 not only gives the idea of the yield of carbon monoxide but also shows how the model can respond

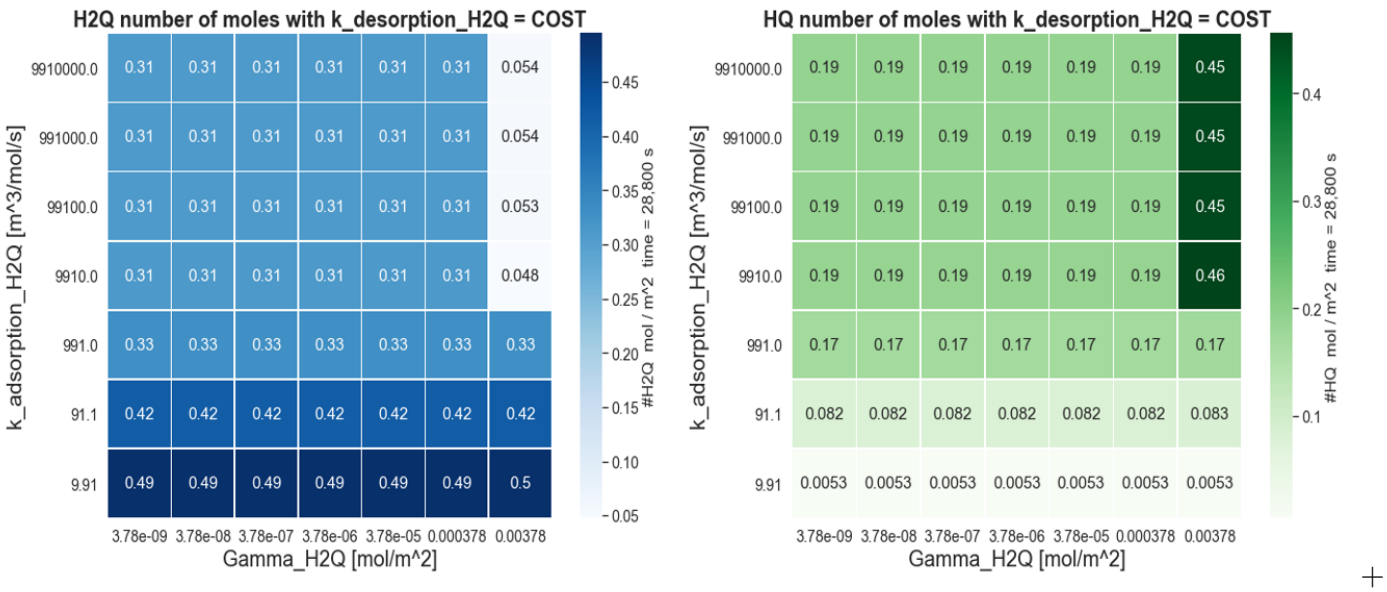

Figure 3.27: Heatmaps for H_2Q and HQ when $k_{des}^{H_2Q}$ is fixed.

to parameters shifting.

3.3.6 Experimental data fitting

The parametric sweeps are useful to understand the model and how the parameters are affecting the simulation. In the previous subsections all the calculation has been focused on describing the software part with no matches with real results. In this part of the work the most uncertain input values like the forward reaction rate and the electron transfer equilibrium adsorption coefficient are modified in order to fit the experimental data. These values are received from the researchers collaborating in the SoFiA project and they are representing the concentration of CO over the time, the experimental results come from a case where no surfactant is applied at the surface and in the case where $C_{12}E_6$ (Dodecyl hexaethylene glycol) is present on the surface at the gas-liquid interface. As mentioned one of the first parameters which is modified in order to fit the experimental data is the forward reaction rate. In

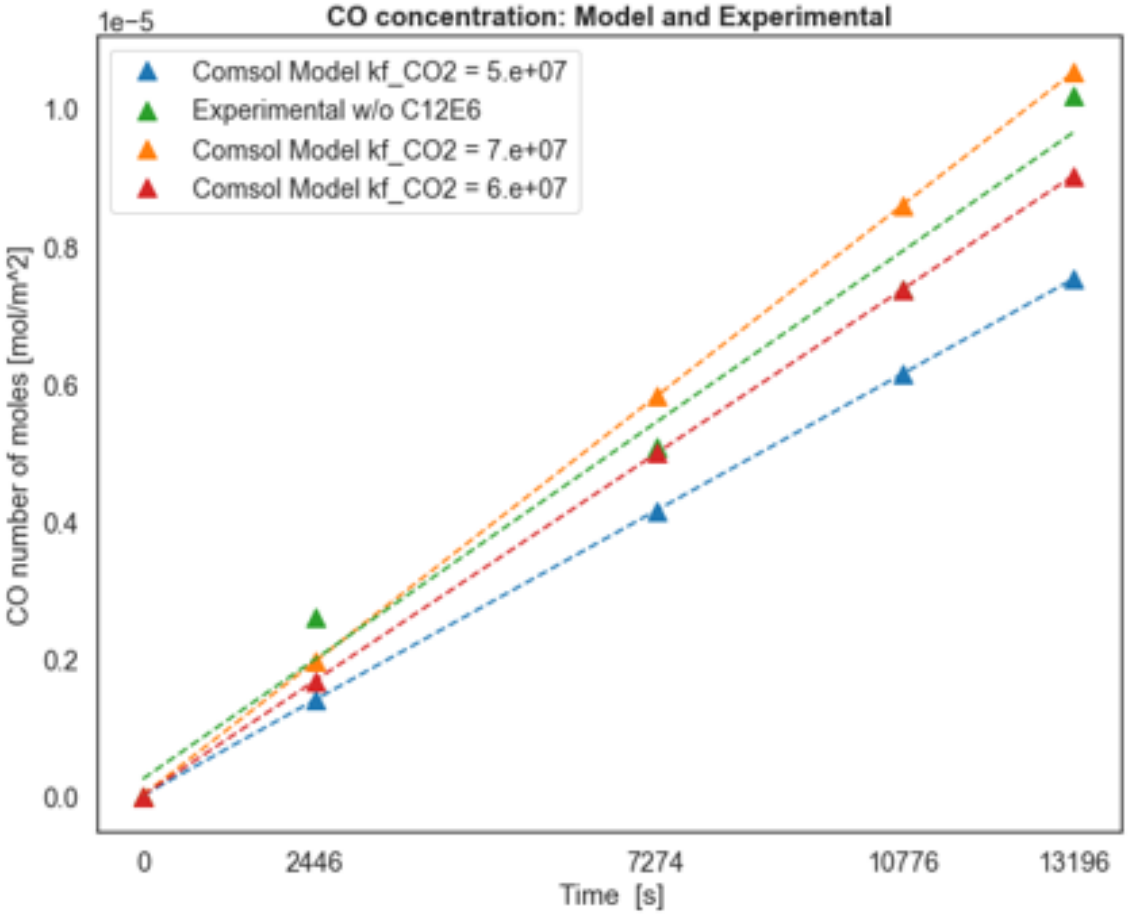

Figure 3.28: Forward reaction rate fitting experimental data without the surfactant.

Figure 3.28 the first grpahical result is depicted, the figure shows the experimental findings without the surfactant. The experimental data have been fitted by a line because of the scarcity of a larger number of data point which could have made the fitting more accurate, this is probably due to the difficulty in collecting data.

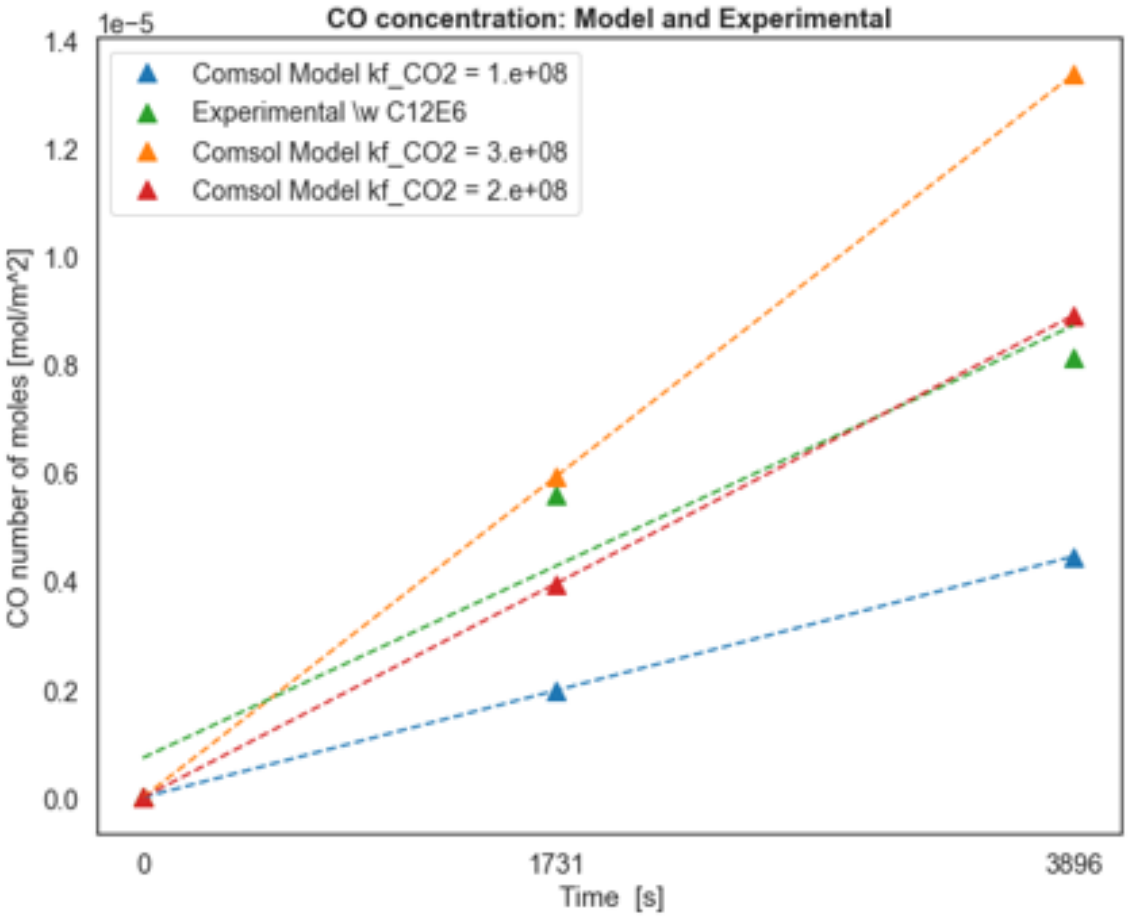

Figure 3.29: Forward reaction rate fitting experimental data with the presence of surfactant at the interface.

The time step considered are 2446, 7274, 10776 and 13196 seconds, the green line correspond to the experimental values while the blue, yellow and red lines come from the COMSOL model at different k_f values. The output which seems to be able of reproducing better the experimental curve is 7 x 10^7 which is lower than what was utilized in the previous simulations. In Figure 3.29 the same simulation is displayed, in this case the surfactant $C_{12}E_6$ is present at the interface, compared to the result without the surfactant, the data point are even less, in fact only two concentration values are available corresponding to 1731 and 3896 seconds, which can be attributed to the instability issued linked to the measuremnt process. Compared to the another case, in this simulation a larger value of k_f is needed to fit the experimental data, the numerber whihc is equal to 2 x 10^8 correspond to the red line. The second parameter shown together with the experimental data in this subsection is the equilibrium adsorption constant of H_2Q, in these simulations the values of the forward reaction rate is kept equal to the starting input value. The results shown in Figure 3.30 and Figure 3.31 display fitting values for the constant which are way lower compared to

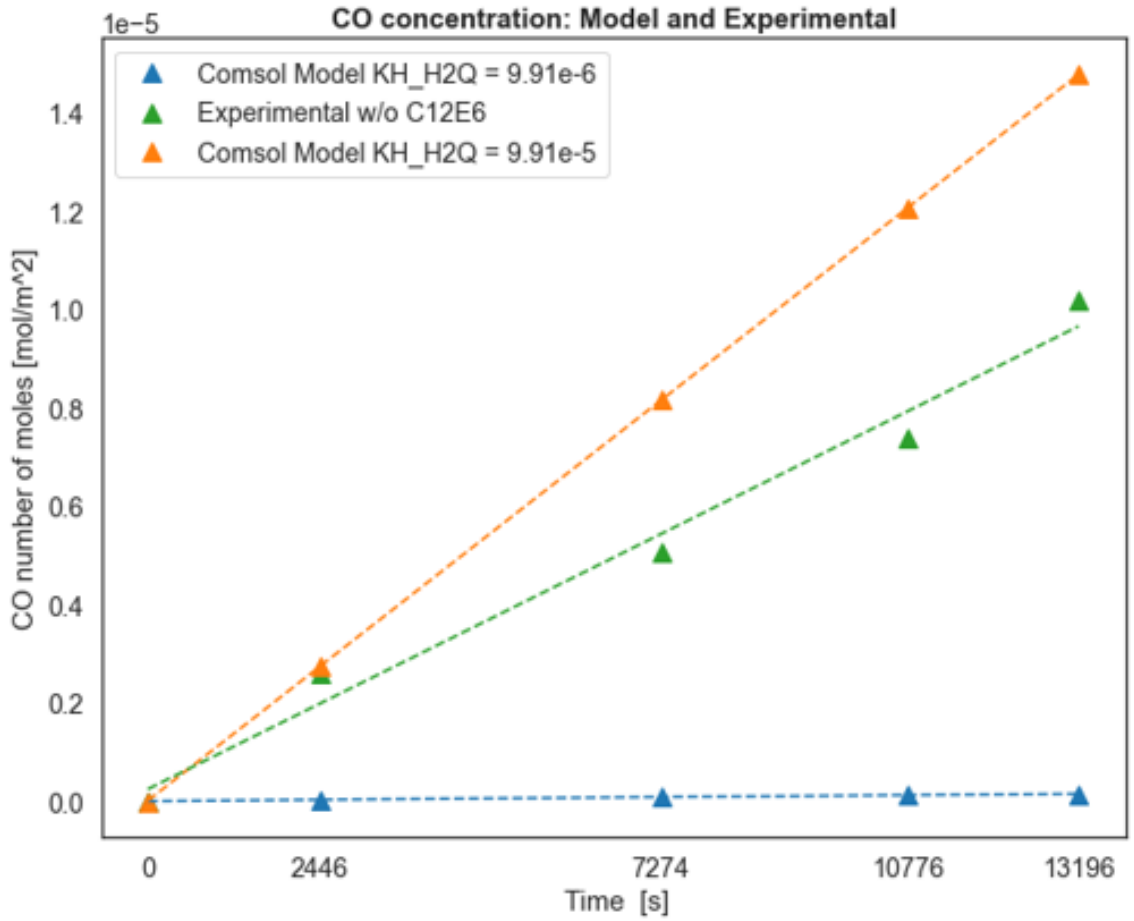

Figure 3.30: Equilibrium adsorption constant of H_2Q fitting experimental data without the surfactant.

the input values taken from the paper where the hydroquinones were adsorbing on chemically modified bentonites. The outputs equal to 1 x 10^{-5} in the case without surfactant and 1 x 10^{-3} with the presence of $C_{12}E_6$ are demonstrating that further studies are needed to gain a deeper knowledge on the initial values but also on the experimental data that are available in a too small number to determine a priori whether the values found in the bibliography can be used in modelling the examined monolayer or not.

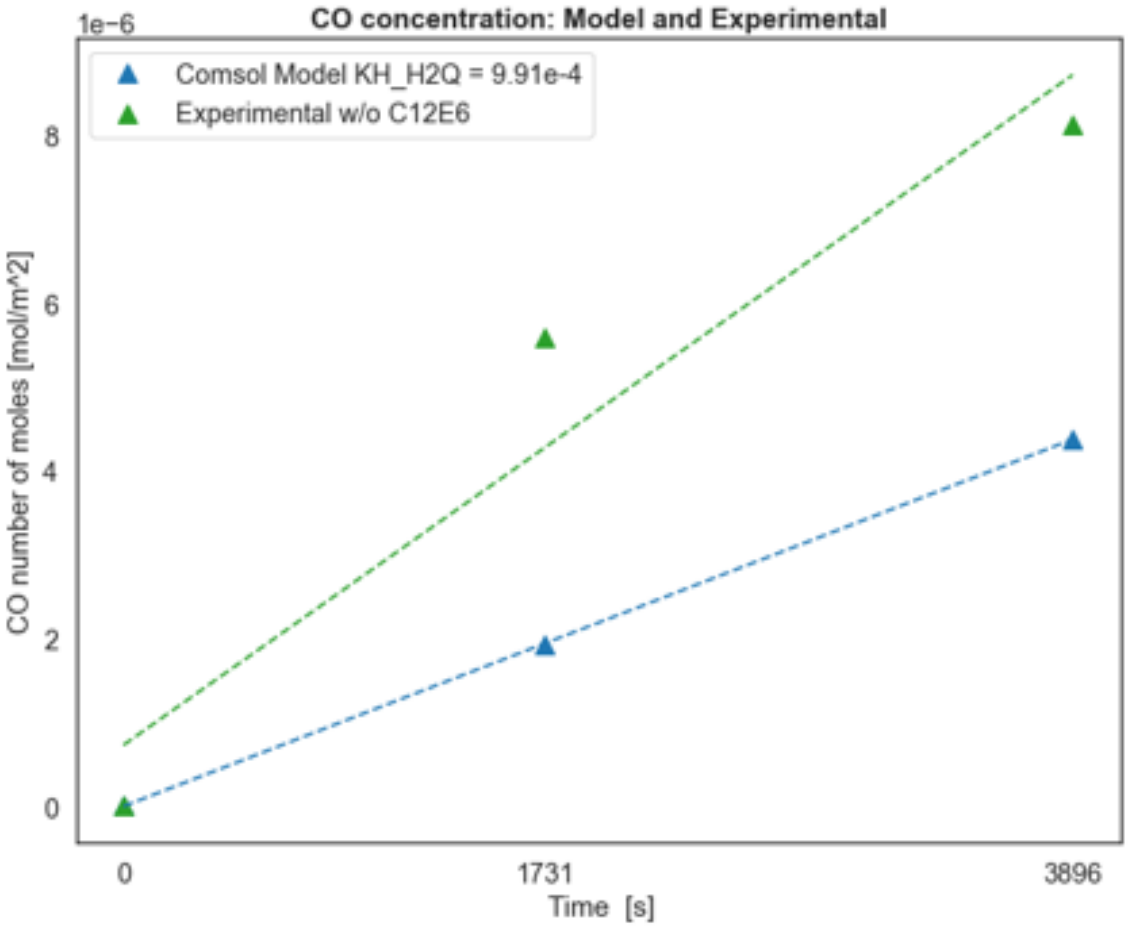

Figure 3.31: Equilibrium adsorption constant of H_2Q fitting experimental data with the presence of surfactant at the interface.

All the Python codes written for the data handling and the production of the majority of the listed figures are given in Appendix A.

Conclusions

This study is focused on the importance of fuels to meet the global energy demand. At the beginning the concept a photosynthetic membranes for solar fuels production has been analyzed and described under a theoretical standpoint. A literature research of the possible electron relays has been conducted in order to find the parameter needed to carry out a COMSOL simulation. At first a validate model of a buffer has been implemented and employed to describe the carbon dioxide dissociation in aqueous solution, subsequently a 0D COMSOL model validated a microkinetic model for homogeneous photocatalytic water oxidation with a view on proving the reliability of the utilized software modules to reproduce the physical phenomena. Subsequently a 1D COMSOL simulation of a surfactant monolayer, where the carbon dioxide is reduced, has been carried out to analyze the main parameters that are affecting the yield of the process, particular attention has been paid to the reduction forward reaction rate and on the Langmuir isotherm parameters of the electron relays chosen and in the end the model parameter has been put into comparison with experimental data. There is still much work to do to find experimental numerical input values for the model at the gas-liquid interface and future studies may be addressed to the insertion of photoactive molecules inside the COMSOL model of the 1D surfactant monolayer.

Python codes

Mass Conservation

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


data = pd.read_fwf('data_comsol_mol.txt')
data


# In[ ]:


data.columns


# ## CO2

# In[ ]:


CO2_sum = []
for i in range (0,data.shape[0]):
    CO2_sum.append(data['CO2_conc_gas'][i] + data['CO2_conc_liq'][i] + data['CO2_surf_conc'][i])

data['CO2_sum']=CO2_sum


# In[ ]:


plt.figure(figsize = (12,8))
sns.set_style('whitegrid')
a = sns.lineplot(x = 'Time', y = 'CO2_sum', data = data, palette = 'rainbow', linewidth = 3)
sns.lineplot(x = 'Time', y = 'CO2_conc_gas', data = data, linewidth = 2 )
sns.lineplot(x = 'Time', y = 'CO2_conc_liq', data = data, linewidth = 2)
sns.lineplot(x = 'Time', y = 'CO2_surf_conc', data = data, linewidth = 2)

a.set(yscale = 'log')
plt.xlabel('Time [s]', size = 15)
plt.ylabel('CO2 concentration [mol/m^2]', size = 15)
plt.title('CO2 moles sum ', fontweight = "bold", size = 15)
plt.legend(['CO2_sum','CO2_gas', 'CO2_liquid', 'CO2_surf'])
plt.xlim(-0.001,30000)

#plt.savefig('CO2_mc.PNG',bbox_inches = "tight")
plt.savefig('CO2log_mc.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


CO_sum = []
for i in range (0,data.shape[0]):
    CO_sum.append(data['CO_conc_gas'][i] + data['CO_conc_liq'][i] + data['CO_surf_conc'][i])

data['CO_sum']=CO_sum


# In[ ]:


plt.figure(figsize = (12,8))
sns.set_style('whitegrid')
a = sns.lineplot(x = 'Time', y = 'CO_sum', data = data, palette = 'rainbow', linewidth = 3)
sns.lineplot(x = 'Time', y = 'CO_conc_gas', data = data, linewidth = 2 )
sns.lineplot(x = 'Time', y = 'CO_conc_liq', data = data, linewidth = 2)
sns.lineplot(x = 'Time', y = 'CO_surf_conc', data = data, linewidth = 2)

a.set(yscale = 'log')
plt.xlabel('Time [s]', size = 15)
```

```
plt.ylabel('CO concentration [mol/m^2]', size = 15)
plt.title('CO moles sum ', fontweight = "bold", size = 15)
plt.legend(['CO_sum','CO_gas', 'CO_liquid', 'CO_surf'])
plt.xlim(-0.001,30000)

#plt.savefig('CO_mc.PNG',bbox_inches = "tight")
plt.savefig('COlog_mc.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


H2Q_sum = []
for i in range (0,data.shape[0]):
    H2Q_sum.append(data['H2Q_conc_liq'][i] + data['H2Q_surf_conc'][i])

data['H2Q_sum']=H2Q_sum


# In[ ]:


plt.figure(figsize = (12,8))
sns.set_style('whitegrid')
a = sns.lineplot(x = 'Time', y = 'H2Q_sum', data = data, palette = 'rainbow', linewidth = 3)

sns.lineplot(x = 'Time', y = 'H2Q_conc_liq', data = data, linewidth = 2)
sns.lineplot(x = 'Time', y = 'H2Q_surf_conc', data = data, linewidth = 2)

a.set(yscale = 'log')
plt.xlabel('Time [s]', size = 15)
plt.ylabel('H2Q concentration [mol/m^2]', size = 15)
plt.title('H2Q moles sum ', fontweight = "bold", size = 15)
plt.legend(['H2Q_sum', 'H2Q_liquid', 'H2Q_surf'])
plt.xlim(-0.001,30000)

#plt.savefig('H2Q_mc.PNG',bbox_inches = "tight")
plt.savefig('H2Qlog_mc.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


HQ_sum = []
for i in range (0,data.shape[0]):
    HQ_sum.append(data['HQ_conc_liq'][i] + data['HQ_surf_conc'][i])

data['HQ_sum']=HQ_sum


# In[ ]:


plt.figure(figsize = (12,8))
sns.set_style('whitegrid')
a = sns.lineplot(x = 'Time', y = 'HQ_sum', data = data, palette = 'rainbow', linewidth = 3)

sns.lineplot(x = 'Time', y = 'HQ_conc_liq', data = data, linewidth = 2)
sns.lineplot(x = 'Time', y = 'HQ_surf_conc', data = data, linewidth = 2)

a.set(yscale = 'log')
plt.xlabel('Time [s]', size = 15)
plt.ylabel('HQ concentration [mol/m^2]', size = 15)
plt.title('HQ moles sum ', fontweight = "bold", size = 15)
plt.legend(['HQ_sum', 'HQ_liquid', 'HQ_surf'])
plt.xlim(-0.001,30000)

#plt.savefig('HQ_mc.PNG',bbox_inches = "tight")
plt.savefig('HQlog_mc.PNG',bbox_inches = "tight")


# ## H2O

# In[ ]:


plt.figure(figsize = (12,8))
sns.set_style('whitegrid')


sns.lineplot(x = 'Time', y = 'H2O_conc_liq', data = data, linewidth = 2)


a.set(yscale = 'log')
plt.xlabel('Time [s]', size = 15)
plt.ylabel('H2O concentration [mol/m^2]', size = 15)
plt.title('H2O moles sum ', fontweight = "bold", size = 15)
plt.legend([ 'H2O_liquid'])
plt.xlim(-0.001,30000)

#plt.savefig('H2O_mc.PNG',bbox_inches = "tight")
plt.savefig('H2Olog_mc.PNG',bbox_inches = "tight")


# In[ ]:


pd.options.display.max_rows = None
data


```

```
# In[ ]:


col = ['CO2_mass [g/m^2]', 'CO_mass [g/m^2]', 'H2Q_mass [g/m^2]', 'HQ_mass [g/m^2]', 'H2O_mass [g/m^2]']
mass = pd.DataFrame(columns = col, index = list(range (0, data.shape[0])))
PM = {'PM_CO2':44, 'PM_CO':28, 'PM_H2Q':110, 'PM_HQ':109, 'PM_H2O':18}
for i in range(0, mass.shape[0]):
    mass[col[0]][i] = data['CO2_sum'][i] * PM['PM_CO2']
    mass[col[1]][i] = data['CO_sum'][i] * PM['PM_CO']
    mass[col[2]][i] = data['H2Q_sum'][i] * PM['PM_H2Q']
    mass[col[3]][i] = data['HQ_sum'][i] * PM['PM_HQ']
    mass[col[4]][i] = data['H2O_conc_liq'][i] * PM['PM_H2O']


# In[ ]:


mass['SUM [g/m^2]'] = mass.sum(axis = 1)

mass = pd.concat([data['Time'],mass], axis = 1)


# In[ ]:


delta = []

for i in range(0,mass.shape[0]):
    delta.append(mass['SUM [g/m^2]'][i]-mass['SUM [g/m^2]'][0])


# In[ ]:


mass['delta [g/m^2]'] = delta


# In[ ]:


mass.set_index(['Time'], drop = True, inplace = True)
mass
```

Animation H2Q each hour

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


from matplotlib import animation
from IPython.display import HTML
import numpy as np
from matplotlib import pyplot
get_ipython().run_line_magic('matplotlib', 'inline')
import pandas as pd


# In[ ]:


start_data = pd.read_fwf('H2Q_space_hour.txt')
start_data


# In[ ]:


x = start_data['X']
col = start_data.columns
y_0 = start_data[col[1]]


# In[ ]:


fig = pyplot.figure(figsize=(8.0, 6.0))
pyplot.xlabel('X - coordinate [m]', size = 12)
pyplot.ylabel('H2Q conc [mol/m^2]', size = 12)
pyplot.title('h                H2Q spatial distribution', size = 12, loc = 'right')
pyplot.grid()

pyplot.ylim(0.0, 105.0)
pyplot.xlim(0.0,0.005)
line = pyplot.plot(x, y_0, color='tab:blue',linewidth = 4)[0]
#line = pyplot.semilogx(x, y_1h,
                   #color='CO', linestyle='dotted', linewidth=2)[0]

fig.tight_layout()


# In[ ]:


ts = len(col)-1
ts


# In[ ]:


y_hist = []
y = y_0.copy()
for i in range(0,ts):
```

```
        y = start_data[col[i+1]]
        y_hist.append(y)


# In[ ]:


def update_plot(n, y_hist):
    """
    Update the line y-data of the Matplotlib figure.

    Parameters
    ----------
    n : integer
        The time-step index.
    y_hist : list of numpy.ndarray objects
        The history of the numerical solution.
    """
    fig.suptitle('Time step {:0>2}'.format(n))
    line.set_ydata(y_hist[n])


# In[ ]:


anim = animation.FuncAnimation(fig, update_plot,
                               frames=ts, fargs=(y_hist,),
                               interval=1000)
# Display the video.
HTML(anim.to_html5_video())


# In[ ]:


anim.save('H2Q_eachH.MP4')
```

Animation H2Q until 1 hour

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


from matplotlib import animation
from IPython.display import HTML
import numpy as np
from matplotlib import pyplot
get_ipython().run_line_magic('matplotlib', 'inline')
import pandas as pd


# In[ ]:


start_data = pd.read_fwf('H2Q_space_firsthour.txt')
start_data


# In[ ]:


x = start_data['X']
col = start_data.columns
y_0 = start_data[col[1]]


# In[ ]:


fig = pyplot.figure(figsize=(8.0, 6.0))
pyplot.xlabel('X - coordinate [m]', size = 12)
pyplot.ylabel('H2Q conc [mol/m^2]', size = 12)
pyplot.title('H2Q spatial distribution until 1 hour', size = 12, loc = 'right')
pyplot.grid()

pyplot.ylim(0.0, 105.0)
pyplot.xlim(0.0,0.005)
line = pyplot.plot(x, y_0, color='tab:blue',linewidth = 4)[0]
#line = pyplot.semilogx(x, y_1h,
                   #color='C0', linestyle='dotted', linewidth=2)[0]

fig.tight_layout()


# In[ ]:


ts = len(col)-1
y_hist = []
y = y_0.copy()
for i in range(0,ts):
        y = start_data[col[i+1]]
        y_hist.append(y)


# In[ ]:


def update_plot(n, y_hist):
    """
    Update the line y-data of the Matplotlib figure.

    Parameters
```

```
    ----------
    n : integer
        The time-step index.
    y_hist : list of numpy.ndarray objects
        The history of the numerical solution.
    """
    fig.suptitle('Time step {:0>2}'.format(n))
    line.set_ydata(y_hist[n])


# In[ ]:


anim = animation.FuncAnimation(fig, update_plot,
                               frames=ts, fargs=(y_hist,),
                               interval=200)
# Display the video.
HTML(anim.to_html5_video())


# In[ ]:


anim.save('H2Q_until1H.MP4')
```

Forward reaction rate

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


data = pd.read_fwf('kf_each_hour.txt')
data


# ## CO2

# In[ ]:


markers_on = [0, 1, 2, 4, 5, 6]


# In[ ]:


sns.set_style('whitegrid')
plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['CO2_conc_gas'][data['Time (s)'] == 14400], 'g*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['CO2_conc_gas'][data['Time (s)'] == 14400])[3], 'gX', markersize = 12)
plt.title('CO2 concentration in the gas region vs kf_CO2', fontweight = "bold", size = 15 )
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('CO2 moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('CO2_gas_14400.PNG',bbox_inches = "tight")


# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['CO2_conc_liq'][data['Time (s)'] == 14400],'b*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['CO2_conc_liq'][data['Time (s)'] == 14400])[3], 'bX', markersize = 12)
plt.title('CO2 concentration in the liquid region vs kf_CO2', fontweight = "bold", size = 15)
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('CO2 moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('CO2_liquid_14400.PNG',bbox_inches = "tight")


# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['CO2_surf_conc'][data['Time (s)'] == 14400],'r*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['CO2_surf_conc'][data['Time (s)'] == 14400])[3], 'rX', markersize = 12)
plt.title('CO2 concentration at the surface vs kf_CO2', fontweight = "bold", size = 15, loc='right')
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('CO2 moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('CO2_surface_14400.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


sns.set_style('whitegrid')
```

```
plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['CO_conc_gas'][data['Time (s)'] == 14400], 'g*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['CO_conc_gas'][data['Time (s)'] == 14400])[3], 'gX', markersize = 12)
plt.title('CO concentration in the gas region vs kf_CO2', fontweight = "bold", size = 15 )
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('CO moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('CO_gas_14400.PNG',bbox_inches = "tight")


# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['CO_conc_liq'][data['Time (s)'] == 14400],'b*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['CO_conc_liq'][data['Time (s)'] == 14400])[3], 'bX', markersize = 12)
plt.title('CO concentration in the liquid region vs kf_CO2', fontweight = "bold", size = 15)
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('CO moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('CO_liquid_14400.PNG',bbox_inches = "tight")


# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['CO_surf_conc'][data['Time (s)'] == 14400],'r*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['CO_surf_conc'][data['Time (s)'] == 14400])[3], 'rX', markersize = 12)
plt.title('CO concentration at the surface vs kf_CO2', fontweight = "bold", size = 15, loc='right')
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('CO moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('CO_surface_14400.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['H2Q_conc_liq'][data['Time (s)'] == 14400],'b*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['H2Q_conc_liq'][data['Time (s)'] == 14400])[3], 'bX', markersize = 12)
plt.title('H2Q concentration in the liquid region vs kf_CO2', fontweight = "bold", size = 15)
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('H2Q moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('H2Q_liquid_14400.PNG',bbox_inches = "tight")


# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['H2Q_surf_conc'][data['Time (s)'] == 14400],'r*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['H2Q_surf_conc'][data['Time (s)'] == 14400])[3], 'rX', markersize = 12)
plt.title('H2Q concentration at the surface vs kf_CO2', fontweight = "bold", size = 15, loc='right')
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('H2Q moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('H2Q_surface_14400.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['HQ_conc_liq'][data['Time (s)'] == 14400],'b*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['HQ_conc_liq'][data['Time (s)'] == 14400])[3], 'bX', markersize = 12)
plt.title('HQ concentration in the liquid region vs kf_CO2', fontweight = "bold", size = 15)
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('HQ moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('HQ_liquid_14400.PNG',bbox_inches = "tight")


# In[ ]:


plt.figure(figsize = (6.5,5.5))
plt.semilogx(data['kf_CO2'].unique(), data['HQ_surf_conc'][data['Time (s)'] == 14400],'r*', markersize = 15,
                 markevery=markers_on, label = 't = 14,400 s')
plt.plot(data['kf_CO2'].unique()[3], list(data['HQ_surf_conc'][data['Time (s)'] == 14400])[3], 'rX', markersize = 12)
plt.title('HQ concentration at the surface vs kf_CO2', fontweight = "bold", size = 15, loc='right')
plt.xlabel('Forward Reaction Rate', size = 15)
plt.ylabel('HQ moles [mol/m^2]', size = 15)
plt.legend()
plt.savefig('HQ_surface_14400.PNG',bbox_inches = "tight")
```

Forward reaction rate animation through time

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

```

```
from matplotlib import animation
from IPython.display import HTML
import numpy as np
from matplotlib import pyplot
get_ipython().run_line_magic('matplotlib', 'inline')
import pandas as pd


# In[ ]:


pd.options.display.max_rows = None
pyplot.rcParams['font.family'] = 'serif'
pyplot.rcParams['font.size'] = 16
start_data = pd.read_fwf('ROBBA.txt')
start_data


# In[ ]:


start_data['CO2_sum'] = start_data[['CO2_conc_gas','CO2_conc_liq','CO2_surf_conc']].sum(axis=1)
start_data['CO_sum'] = start_data[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
start_data['H2Q_sum'] = start_data[['H2Q_conc_liq','H2Q_surf_conc']].sum(axis=1)
start_data['HQ_sum'] = start_data[['HQ_conc_liq','HQ_surf_conc']].sum(axis=1)


# ## CO2

# In[ ]:


x = start_data['kf_CO2'].unique()
y_1h = start_data['CO2_sum'][start_data['Time (s)'] == 0]


# In[ ]:


fig = pyplot.figure(figsize=(8.0, 6.0))
pyplot.xlabel('Forward reaction rate', size = 12)
pyplot.ylabel('CO2_conc [mol/m^2]', size = 12)
pyplot.title('h      total # of CO2 moles vs kf_CO2', size = 11, loc = 'right')
pyplot.grid()

pyplot.ylim(0.300, 0.410)

line = pyplot.semilogx(x, y_1h,'g*', color='tab:orange', markersize = 15)[0]
#line = pyplot.semilogx(x, y_1h,
                   #color='C0', linestyle='dotted', linewidth=2)[0]

fig.tight_layout()


# In[ ]:


time_step = start_data['Time (s)']. unique()
ts = len(time_step)

y_hist = []
y = y_1h.copy()
for i in range(0,ts):
        y = start_data['CO2_sum'][start_data['Time (s)'] == time_step[i]]
        y_hist.append(y)


# In[ ]:


def update_plot(n, y_hist):
    """
    Update the line y-data of the Matplotlib figure.

    Parameters
    ----------
    n : integer
        The time-step index.
    y_hist : list of numpy.ndarray objects
        The history of the numerical solution.
    """
    fig.suptitle('Time step {:0>2}'.format(n))
    line.set_ydata(y_hist[n])


# In[ ]:


anim = animation.FuncAnimation(fig, update_plot,
                               frames=ts, fargs=(y_hist,),
                               interval=1000)
# Display the video.
HTML(anim.to_html5_video())


# In[ ]:


anim.save('CO2_kf.MP4')


# ## CO

# In[ ]:
```

```

y_1h = start_data['CO_sum'][start_data['Time (s)'] == 0]


# In[ ]:


fig = pyplot.figure(figsize=(8.0, 6.0))
pyplot.xlabel('Forward reaction rate', size = 12)
pyplot.ylabel('CO_conc [mol/m^2]', size = 12)
pyplot.title('h      total # of CO moles vs kf_CO2', size = 11, loc = 'right')
pyplot.grid()

pyplot.ylim(-0.010, 0.100)

line = pyplot.semilogx(x, y_1h,'g*', color='tab:purple', markersize = 15)[0]
#line = pyplot.semilogx(x, y_1h,
                   #color='CO', linestyle='dotted', linewidth=2)[0]

fig.tight_layout()


# In[ ]:


time_step = start_data['Time (s)']. unique()
ts = len(time_step)

y_hist = []
y = y_1h.copy()
for i in range(0,ts):
        y = start_data['CO_sum'][start_data['Time (s)'] == time_step[i]]
        y_hist.append(y)


# In[ ]:


def update_plot(n, y_hist):
    fig.suptitle('Time step {:0>2}'.format(n))
    line.set_ydata(y_hist[n])


# In[ ]:


anim = animation.FuncAnimation(fig, update_plot,
                               frames=ts, fargs=(y_hist,),
                               interval=1000)
# Display the video.
HTML(anim.to_html5_video())


# In[ ]:


anim.save('CO_kf.MP4')


# ## H2Q

# In[ ]:


y_1h = start_data['H2Q_sum'][start_data['Time (s)'] == 0]


# In[ ]:


fig = pyplot.figure(figsize=(8.0, 6.0))
pyplot.xlabel('Forward reaction rate', size = 12)
pyplot.ylabel('H2Q_conc [mol/m^2]', size = 12)
pyplot.title('h      total # of H2Q moles vs kf_CO2', size = 11, loc = 'right')
pyplot.grid()

pyplot.ylim(0.300, 0.510)

line = pyplot.semilogx(x, y_1h,'g*', color='tab:cyan', markersize = 15)[0]
#line = pyplot.semilogx(x, y_1h,
                   #color='CO', linestyle='dotted', linewidth=2)[0]

fig.tight_layout()


# In[ ]:


time_step = start_data['Time (s)']. unique()
ts = len(time_step)

y_hist = []
y = y_1h.copy()
for i in range(0,ts):
        y = start_data['H2Q_sum'][start_data['Time (s)'] == time_step[i]]
        y_hist.append(y)


# In[ ]:


def update_plot(n, y_hist):
    fig.suptitle('Time step {:0>2}'.format(n))
    line.set_ydata(y_hist[n])
```

```

# In[ ]:


anim = animation.FuncAnimation(fig, update_plot,
                               frames=ts, fargs=(y_hist,),
                               interval=1000)
# Display the video.
HTML(anim.to_html5_video())


# In[ ]:


anim.save('H2Q_kf.MP4')


# ## HQ

# In[ ]:


y_1h = start_data['HQ_sum'][start_data['Time (s)'] == 0]


# In[ ]:


fig = pyplot.figure(figsize=(8.0, 6.0))
pyplot.xlabel('Forward reaction rate', size = 12)
pyplot.ylabel('HQ_conc [mol/m^2]', size = 12)
pyplot.title('h      total # of HQ moles vs kf_CO2', size = 11, loc = 'right')
pyplot.grid()

pyplot.ylim(-0.010, 0.200)

line = pyplot.semilogx(x, y_1h,'g*', color='tab:olive', markersize = 15)[0]
#line = pyplot.semilogx(x, y_1h,
                  #color='C0', linestyle='dotted', linewidth=2)[0]

fig.tight_layout()


# In[ ]:


time_step = start_data['Time (s)']. unique()
ts = len(time_step)

y_hist = []
y = y_1h.copy()
for i in range(0,ts):
        y = start_data['HQ_sum'][start_data['Time (s)'] == time_step[i]]
        y_hist.append(y)


# In[ ]:


def update_plot(n, y_hist):
    fig.suptitle('Time step {:0>2}'.format(n))
    line.set_ydata(y_hist[n])


# In[ ]:


anim = animation.FuncAnimation(fig, update_plot,
                               frames=ts, fargs=(y_hist,),
                               interval=1000)
# Display the video.
HTML(anim.to_html5_video())


# In[ ]:


anim.save('HQ_kf.MP4')
```

Heatmap CO2 case 1

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_CO2_KH_Gamma.txt')
start_data


# In[ ]:

```

```
CO2_sum = []
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


start_data


# In[ ]:


data1 = start_data[['Gamma_sCO2','KH_CO2','CO2_sum']]


# In[ ]:


k_des = int(10000)
K_ads = []
for i in range (0,data1.shape[0]):
    K_ads.append(data1['KH_CO2'][i] *k_des)

data1['K_adsorption'] = pd.Series(K_ads)


# In[ ]:


data1.drop(['KH_CO2'], axis = 1, inplace = True)


# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
    row.append(round(K_ads[i],6))

col = data1['Gamma_sCO2'].unique()
row.reverse()


# In[ ]:


data2 = pd.DataFrame(columns = col, index = row)
data2


# In[ ]:


start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data2, annot = True, cmap = 'viridis', linewidths=.5, cbar_kws={'label': '#CO2  mol / m^2  time = 28,800 s'})
plt.title('CO2 number of moles with k_desorption_CO2 = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
#plt.savefig('CO2_3_heatmap_varCO2.PNG',bbox_inches = "tight")


# In[ ]:


data3 = data2.copy()
start = 0
```

```
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'crest', linewidths=.5, cbar_kws={'label': '#CO  mol / m^2  time = 28,800 s'})
plt.title('CO number of moles with k_desorption_CO2 = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
#plt.savefig('CO_3_heatmap_varCO2.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


data4 = data2.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = H2Q_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Blues', linewidths=.5, cbar_kws={'label': '#H2Q  mol / m^2  time = 28,800 s'})
plt.title('H2Q number of moles with k_desorption_CO2 = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
#plt.savefig('H2Q_3_heatmap_varCO2.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


data5 = data2.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data5[col[i]] = todf
    start = start+n_values
    end = end+n_values

data5


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data5, annot = True, cmap = 'Greens', linewidths=.5, cbar_kws={'label': '#HQ  mol / m^2  time = 28,800 s'})
plt.title('HQ number of moles with k_desorption_CO2 = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
#plt.savefig('HQ_3_heatmap_varCO2.PNG',bbox_inches = "tight")
```

Heatmap CO case 1

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_CO_KH_Gamma.txt')
start_data


# In[ ]:


CO2_sum = []
```

```
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


k_des = int(10000)
K_ads = []
for i in range (0,start_data.shape[0]):
    K_ads.append(start_data['KH_CO'][i] *k_des)



# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
    row.append(round(K_ads[i],6))

col = start_data['Gamma_sCO'].unique()
col = np.round_(col, decimals = 11)
row.reverse()


# ## CO2

# In[ ]:


data1 = pd.DataFrame(columns = col, index = row)
data1


# In[ ]:


start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data1[col[i]] = todf
    start = start+n_values
    end = end+n_values

data1


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data1, annot = True, cmap = 'viridis', vmax = max(CO2_sum), vmin = max(CO2_sum), linewidths=.5, cbar_kws={'label': '#CO2  mol / m^2
     time = 28,800 s'})
plt.title('CO2 number of moles with k_desorption_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('CO2_3_heatmap_varCO.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


data2 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
```

```
sns.heatmap(data2, annot = True, cmap = 'crest', vmax = max(CO_sum), vmin = max(CO_sum), linewidths=.5, cbar_kws={'label': '#CO  mol / m^2  time = 28,800 s'})
plt.title('CO number of moles with k_desorption_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('CO_3_heatmap_varCO.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


data3 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = H2Q_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'Blues', vmax = max(H2Q_sum), vmin = max(H2Q_sum), linewidths=.5, cbar_kws={'label': '#H2Q  mol / m^2  time = 28,800 s'})
plt.title('H2Q number of moles with k_desorption_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('H2Q_3_heatmap_varCO.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


data4 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Greens', vmax = max(HQ_sum), vmin = max(HQ_sum), linewidths=.5, cbar_kws={'label': '#HQ  mol / m^2  time = 28,800 s'})
plt.title('HQ number of moles with k_desorption_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('HQ_3_heatmap_varCO.PNG',bbox_inches = "tight")
```

Heatmap H2Q case 1

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_H2Q_KH_Gamma.txt')
start_data


# In[ ]:


CO2_sum = []
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
```

```
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


k_des = int(10000)
K_ads = []
for i in range (0,start_data.shape[0]):
    K_ads.append(start_data['KH_H2Q'][i] *k_des)


# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
    row.append(round(K_ads[i],6))

col = start_data['Gamma_sH2Q'].unique()
col = np.round_(col, decimals = 11)
row.reverse()


# In[ ]:


data1 = pd.DataFrame(columns = col, index = row)


# ## CO2

# In[ ]:


start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data1[col[i]] = todf
    start = start+n_values
    end = end+n_values

data1


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data1, annot = True, cmap = 'viridis', linewidths=.5, cbar_kws={'label': '#CO2  mol / m^2  time = 28,800 s'})
plt.title('CO2 number of moles with k_desorption_H2Q = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('CO2_3_heatmap_varH2Q.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


data2 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data2, annot = True, cmap = 'crest', linewidths=.5, cbar_kws={'label': '#CO  mol / m^2  time = 28,800 s'})
plt.title('CO number of moles with k_desorption_H2Q = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('CO_3_heatmap_varH2Q.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


data3 = data1.copy()
start = 0
end = n_values
```

```
for i in range (0,len(col)):
    todf = H2Q_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'Blues', linewidths=.5, cbar_kws={'label': '#H2Q  mol / m^2  time = 28,800 s'})
plt.title('H2Q number of moles with k_desorption_H2Q = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('H2Q_3_heatmap_varH2Q.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


data4 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Greens', linewidths=.5, cbar_kws={'label': '#HQ  mol / m^2  time = 28,800 s'})
plt.title('HQ number of moles with k_desorption_H2Q = COST', size = 20, fontweight="bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('HQ_3_heatmap_varH2Q.PNG',bbox_inches = "tight")
```

Heatmap HQ case 1

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_HQ_KH_Gamma.txt')
start_data


# In[ ]:


CO2_sum = []
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


k_des = int(10000)
K_ads = []
for i in range (0,start_data.shape[0]):
```

```
    K_ads.append(start_data['KH_HQ'][i] *k_des)


# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
    row.append(round(K_ads[i],6))

col = start_data['Gamma_sHQ'].unique()
col = np.round_(col, decimals = 11)
row.reverse()


# In[ ]:


data1 = pd.DataFrame(columns = col, index = row)


# ## CO2

# In[ ]:


start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data1[col[i]] = todf
    start = start+n_values
    end = end+n_values

data1


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data1, annot = True, cmap = 'viridis', vmax = max(CO2_sum), vmin = max(CO2_sum), linewidths=.5, cbar_kws={'label': '#CO2  mol / m^2
     time = 28,800 s'})
plt.title('CO2 number of moles with k_desorption_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('CO2_3_heatmap_varHQ.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


data2 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data2, annot = True, cmap = 'crest', vmax = max(CO_sum), vmin = max(CO_sum), linewidths=.5, cbar_kws={'label': '#CO  mol / m^2  time =
     28,800 s'})
plt.title('CO number of moles with k_desorption_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('CO_3_heatmap_varHQ.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


data3 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = H2Q_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'Blues', vmax = max(H2Q_sum), vmin = max(H2Q_sum), linewidths=.5, cbar_kws={'label': '#H2Q  mol / m^2  time
     = 28,800 s'})
```

```
plt.title('H2Q number of moles with k_desorption_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('H2Q_3_heatmap_varHQ.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


data4 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Greens', vmax = max(HQ_sum), vmin = max(HQ_sum), linewidths=.5, cbar_kws={'label': '#HQ  mol / m^2  time =
    28,800 s'})
plt.title('HQ number of moles with k_desorption_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('HQ_3_heatmap_varHQ.PNG',bbox_inches = "tight")
```

Heatmap CO2 case 2

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_CO2_Kads_Gamma.txt')
start_data


# In[ ]:


CO2_sum = []
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
    row.append(round(start_data['k_adsCO2'][i],6))

col = start_data['Gamma_sCO2'].unique()
row.reverse()


# In[ ]:


data2 = pd.DataFrame(columns = col, index = row)

start = 0
end = n_values
```

```
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data2, annot = True, cmap = 'viridis', linewidth=.5, cbar_kws={'label': '#CO2  mol / m^2  time = 28,800 s'})
plt.title('CO2 number of moles with Keq_CO2 = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
plt.savefig('CO2_2_heatmap_varCO2.PNG',bbox_inches = "tight")


# In[ ]:


data3 = data2.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'crest', linewidth=.5, cbar_kws={'label': '#CO  mol / m^2  time = 28,800 s'})
plt.title('CO number of moles with Keq_CO2 = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
plt.savefig('CO_2_heatmap_varCO2.PNG',bbox_inches = "tight")


## H2Q

# In[ ]:


data4 = data2.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = H2Q_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Blues', linewidth=.5, cbar_kws={'label': '#H2Q  mol / m^2  time = 28,800 s'})
plt.title('H2Q number of moles with Keq_CO2 = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
plt.savefig('H2Q_2_heatmap_varCO2.PNG',bbox_inches = "tight")


## HQ

# In[ ]:


data5 = data2.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data5[col[i]] = todf
    start = start+n_values
    end = end+n_values

data5


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data5, annot = True, cmap = 'Greens', linewidth=.5, cbar_kws={'label': '#HQ  mol / m^2  time = 28,800 s'})
plt.title('HQ number of moles with Keq_CO2 = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO2 [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO2 [m^3/mol/s]', size = 20)
plt.savefig('HQ_2_heatmap_varCO2.PNG',bbox_inches = "tight")
```

Heatmap CO case 2

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_CO_Kads_Gamma.txt')
start_data


# In[ ]:


CO2_sum = []
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
    row.append(round(start_data['k_adsCO'][i],6))

col = start_data['Gamma_sCO'].unique()
col = np.round_(col, decimals = 11)
row.reverse()


# ## CO2

# In[ ]:


data1 = pd.DataFrame(columns = col, index = row)

start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data1[col[i]] = todf
    start = start+n_values
    end = end+n_values

data1


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data1, annot = True, cmap = 'viridis', vmax = max(CO2_sum), vmin = max(CO2_sum), linewidths=.5, cbar_kws={'label': '#CO2  mol / m^2
     time = 28,800 s'})
plt.title('CO2 number of moles with Keq_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('CO2_2_heatmap_varCO.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


data2 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
```

```
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data2, annot = True, cmap = 'crest', vmax = max(CO_sum), vmin = max(CO_sum), linewidths=.5, cbar_kws={'label': '#CO  mol / m^2  time = 28,800 s'})
plt.title('CO number of moles with Keq_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('CO_2_heatmap_varCO.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


data3 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = H2Q_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'Blues', vmax = max(H2Q_sum), vmin = max(H2Q_sum), linewidths=.5, cbar_kws={'label': '#H2Q  mol / m^2  time = 28,800 s'})
plt.title('H2Q number of moles with Keq_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('H2Q_2_heatmap_varCO.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


data4 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Greens', vmax = max(HQ_sum), vmin = max(HQ_sum), linewidths=.5, cbar_kws={'label': '#HQ  mol / m^2  time = 28,800 s'})
plt.title('HQ number of moles with Keq_CO = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_CO [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_CO [m^3/mol/s]', size = 20)
plt.savefig('HQ_2_heatmap_varCO.PNG',bbox_inches = "tight")
```

Heatmap H2Q case 2

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_H2Q_Kads_Gamma.txt')
start_data


# In[ ]:


CO2_sum = []
```

```
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
    row.append(round(start_data['k_adsH2Q'][i],6))

col = start_data['Gamma_sH2Q'].unique()
col = np.round_(col, decimals = 11)
row.reverse()


# ## CO2

# In[ ]:


data1 = pd.DataFrame(columns = col, index = row)

start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data1[col[i]] = todf
    start = start+n_values
    end = end+n_values

data1


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data1, annot = True, cmap = 'viridis', linewidth=.5, cbar_kws={'label': '#CO2  mol / m^2  time = 28,800 s'})
plt.title('CO2 number of moles with Keq_H2Q = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('CO2_2_heatmap_varH2Q.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


data2 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data2, annot = True, cmap = 'crest', linewidth=.5, cbar_kws={'label': '#CO  mol / m^2  time = 28,800 s'})
plt.title('CO number of moles with Keq_H2Q = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('CO_2_heatmap_varH2Q.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


data3 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
```

```
    todf = H2Q_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'Blues', linewidth=.5, cbar_kws={'label': '#H2Q  mol / m^2  time = 28,800 s'})
plt.title('H2Q number of moles with Keq_H2Q = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('H2Q_2_heatmap_varH2Q.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


data4 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Greens', linewidth=.5, cbar_kws={'label': '#HQ  mol / m^2  time = 28,800 s'})
plt.title('HQ number of moles with Keq_H2Q = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_H2Q [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_H2Q [m^3/mol/s]', size = 20)
plt.savefig('HQ_2_heatmap_varH2Q.PNG',bbox_inches = "tight")
```

Heatmap HQ case 2

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale = 1.3)


# In[ ]:


start_data = pd.read_fwf('all_HQ_Kads_Gamma.txt')
start_data


# In[ ]:


CO2_sum = []
for i in range (0,start_data.shape[0]):
    CO2_sum.append(start_data['CO2_conc_gas'][i] + start_data['CO2_conc_liq'][i] + start_data['CO2_surf_conc'][i])

start_data['CO2_sum']=CO2_sum

CO_sum = []
for i in range (0,start_data.shape[0]):
    CO_sum.append(start_data['CO_conc_gas'][i] + start_data['CO_conc_liq'][i] + start_data['CO_surf_conc'][i])

start_data['CO_sum']=CO_sum

H2Q_sum = []
for i in range (0,start_data.shape[0]):
    H2Q_sum.append(start_data['H2Q_conc_liq'][i] + start_data['H2Q_surf_conc'][i])

start_data['H2Q_sum']=H2Q_sum

HQ_sum = []
for i in range (0,start_data.shape[0]):
    HQ_sum.append(start_data['HQ_conc_liq'][i] + start_data['HQ_surf_conc'][i])

start_data['HQ_sum']=HQ_sum


# In[ ]:


row =[]
n_values = 7
for i in range (0,n_values):
```

```
    row.append(round(start_data['k_adsHQ'][i],6))

col = start_data['Gamma_sHQ'].unique()
col = np.round_(col, decimals = 11)
row.reverse()


# ## CO2

# In[ ]:


data1 = pd.DataFrame(columns = col, index = row)

start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO2_sum[start:end]
    todf.reverse()
    data1[col[i]] = todf
    start = start+n_values
    end = end+n_values

data1


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data1, annot = True, cmap = 'viridis', vmax = max(CO2_sum), vmin = max(CO2_sum), linewidths=.5, cbar_kws={'label': '#CO2  mol / m^2
     time = 28,800 s'})
plt.title('CO2 number of moles with Keq_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('CO2_2_heatmap_varHQ.PNG',bbox_inches = "tight")


# ## CO

# In[ ]:


data2 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = CO_sum[start:end]
    todf.reverse()
    data2[col[i]] = todf
    start = start+n_values
    end = end+n_values

data2


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data2, annot = True, cmap = 'crest', vmax = max(CO_sum), vmin = max(CO_sum), linewidths=.5, cbar_kws={'label': '#CO  mol / m^2  time =
     28,800 s'})
plt.title('CO number of moles with Keq_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('CO_2_heatmap_varHQ.PNG',bbox_inches = "tight")


# ## H2Q

# In[ ]:


data3 = data1.copy()
start = 0
end = n_values
for i in range (0,len(col)):
    todf = H2Q_sum[start:end]
    todf.reverse()
    data3[col[i]] = todf
    start = start+n_values
    end = end+n_values

data3


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data3, annot = True, cmap = 'Blues', vmax = max(H2Q_sum), vmin = max(H2Q_sum), linewidths=.5, cbar_kws={'label': '#H2Q  mol / m^2  time
     = 28,800 s'})
plt.title('H2Q number of moles with Keq_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('H2Q_2_heatmap_varHQ.PNG',bbox_inches = "tight")


# ## HQ

# In[ ]:


data4 = data1.copy()
start = 0
```

```
end = n_values
for i in range (0,len(col)):
    todf = HQ_sum[start:end]
    todf.reverse()
    data4[col[i]] = todf
    start = start+n_values
    end = end+n_values

data4


# In[ ]:


plt.figure(figsize = (10,8))
sns.heatmap(data4, annot = True, cmap = 'Greens', vmax = max(HQ_sum), vmin = max(HQ_sum), linewidths=.5, cbar_kws={'label': '#HQ  mol / m^2  time =
    28,800 s'})
plt.title('HQ number of moles with Keq_HQ = COST', size = 20, fontweight = "bold")
plt.xlabel('Gamma_HQ [mol/m^2]', size = 20)
plt.ylabel('k_adsorption_HQ [m^3/mol/s]', size = 20)
plt.savefig('HQ_2_heatmap_varHQ.PNG',bbox_inches = "tight")
```

FIT forward reaction rate test number 1

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import seaborn as sns
sns.set(font_scale = 1.3)
sns.set_style("white")


# In[ ]:


data = pd.read_fwf('per_fitting.txt')


# In[ ]:


data.head(100)


# In[ ]:


data['kf_CO2'].unique()


# In[ ]:


data.columns


# In[ ]:


data0 = data[data['kf_CO2'] == data['kf_CO2'].unique()[0]]
data1 = data[data['kf_CO2'] == data['kf_CO2'].unique()[1]]
data2 = data[data['kf_CO2'] == data['kf_CO2'].unique()[2]]
data3 = data[data['kf_CO2'] == data['kf_CO2'].unique()[3]]
data4 = data[data['kf_CO2'] == data['kf_CO2'].unique()[4]]
data5 = data[data['kf_CO2'] == data['kf_CO2'].unique()[5]]
data6 = data[data['kf_CO2'] == data['kf_CO2'].unique()[6]]


# In[ ]:


data0.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data1.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data2.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data3.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data4.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data5.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data6.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)


# In[ ]:


data_0 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 2445],data0[data0['Time(s)'] == 7274],
                    data0[data0['Time(s)'] == 10776],data0[data0['Time(s)'] == 13195]], axis = 0)


# In[ ]:


```

```
data_1 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 2445],data1[data1['Time(s)'] == 7274],
                    data1[data1['Time(s)'] == 10776],data1[data1['Time(s)'] == 13195]], axis = 0)


# In[ ]:


data_2 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 2445],data2[data2['Time(s)'] == 7274],
                    data2[data2['Time(s)'] == 10776],data2[data2['Time(s)'] == 13195]], axis = 0)


# In[ ]:


data_3 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 2445],data3[data3['Time(s)'] == 7274],
                    data3[data3['Time(s)'] == 10776],data3[data3['Time(s)'] == 13195]], axis = 0)


# In[ ]:


data_4 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 2445],data4[data4['Time(s)'] == 7274],
                    data4[data4['Time(s)'] == 10776],data4[data4['Time(s)'] == 13195]], axis = 0)


# In[ ]:


data_5 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 2445],data5[data5['Time(s)'] == 7274],
                    data5[data5['Time(s)'] == 10776],data5[data5['Time(s)'] == 13195]], axis = 0)


# In[ ]:


data_6 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 2445],data6[data6['Time(s)'] == 7274],
                    data6[data6['Time(s)'] == 10776],data6[data6['Time(s)'] == 13195]], axis = 0)


# In[ ]:


Value_to_add = [0.00000000, 0.00000261,0.00000507,0.00000737,0.0000102]
data_0['experimental'] = Value_to_add


# In[ ]:


data_1['experimental'] = Value_to_add
data_2['experimental'] = Value_to_add
data_3['experimental'] = Value_to_add
data_4['experimental'] = Value_to_add
data_5['experimental'] = Value_to_add
data_6['experimental'] = Value_to_add


# In[ ]:


CO_sum = data_0[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_0['CO_sum']= CO_sum
CO_sum = data_1[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_1['CO_sum']= CO_sum
CO_sum = data_2[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_2['CO_sum']= CO_sum
CO_sum = data_3[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_3['CO_sum']= CO_sum
CO_sum = data_4[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_4['CO_sum']= CO_sum
CO_sum = data_5[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_5['CO_sum']= CO_sum
CO_sum = data_6[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_6['CO_sum']= CO_sum


# ## kf_CO2 = 10^9

# In[ ]:


data_0


# In[ ]:



x_values = data_0['Time(s)']
y_values = data_0['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_0['Time(s)']
```

```
y_values = data_0['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:



plt.figure(figsize = (10,8))
plt.plot(data_0['Time(s)'], data_0['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+09')
plt.plot(data_0['Time(s)'], data_0['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_0['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()

plt.savefig('CO_w_kf9.PNG',bbox_inches = "tight")


# In[ ]:




# ## kf_CO2 = 10^10

# In[ ]:


data_1


# In[ ]:


x_values = data_1['Time(s)']
y_values = data_1['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_1['Time(s)']
y_values = data_1['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_1['Time(s)'], data_1['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+10')
plt.plot(data_1['Time(s)'], data_1['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_w_kf10.PNG',bbox_inches = "tight")


# In[ ]:




# In[ ]:


```

```
# In[ ]:




# ## kf_CO2 = 10^11

# In[ ]:


data_2


# In[ ]:


x_values = data_2['Time(s)']
y_values = data_2['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_2['Time(s)']
y_values = data_2['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_2['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+11')
plt.plot(data_2['Time(s)'], data_2['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_2['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_w_kf11.PNG',bbox_inches = "tight")


# In[ ]:




# ## kf_CO2 = 10^12

# In[ ]:


data_3


# In[ ]:


x_values = data_3['Time(s)']
y_values = data_3['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_3['Time(s)']
y_values = data_3['CO_sum']
def objective(x, a, b, c):
    return a * x + b
```

```
popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_3['Time(s)'], data_3['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+12')
plt.plot(data_3['Time(s)'], data_3['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_3['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_w_kf12.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^13

# In[ ]:


data_4


# In[ ]:


x_values = data_4['Time(s)']
y_values = data_4['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_4['Time(s)']
y_values = data_4['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_4['Time(s)'], data_4['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+13')
plt.plot(data_4['Time(s)'], data_4['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_4['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_w_kf13.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^14

# In[ ]:


data_5


# In[ ]:


x_values = data_5['Time(s)']
y_values = data_5['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
```

```
y_exp = y_new


# In[ ]:


x_values = data_5['Time(s)']
y_values = data_5['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_5['Time(s)'], data_5['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+14')
plt.plot(data_5['Time(s)'], data_5['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_5['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_w_kf14.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^15

# In[ ]:


data_6


# In[ ]:


x_values = data_6['Time(s)']
y_values = data_6['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_6['Time(s)']
y_values = data_6['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_6['Time(s)'], data_6['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+15')
plt.plot(data_6['Time(s)'], data_6['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_6['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_w_kf15.PNG',bbox_inches = "tight")


# ##  SECOND SET OF EXPERIMENTAL DATA

#


#


#

# In[ ]:
```

```
data_00 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 1732],data0[data0['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_01 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 1732],data1[data1['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_02 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 1732],data2[data2['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_03 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 1732],data3[data3['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_04 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 1732],data4[data4['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_05 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 1732],data5[data5['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_06 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 1732],data6[data6['Time(s)'] == 3896]],axis = 0)

# In[ ]:

Value_to_add2 = [0.00000000, 0.00000559, 0.00000814]

# In[ ]:

data_00['experimental'] = Value_to_add2
data_01['experimental'] = Value_to_add2
data_02['experimental'] = Value_to_add2
data_03['experimental'] = Value_to_add2
data_04['experimental'] = Value_to_add2
data_05['experimental'] = Value_to_add2
data_06['experimental'] = Value_to_add2

# In[ ]:

CO_sum = data_00[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_00['CO_sum']= CO_sum
CO_sum = data_01[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_01['CO_sum']= CO_sum
CO_sum = data_02[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_02['CO_sum']= CO_sum
CO_sum = data_03[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_03['CO_sum']= CO_sum
CO_sum = data_04[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_04['CO_sum']= CO_sum
CO_sum = data_05[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_05['CO_sum']= CO_sum
CO_sum = data_06[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_06['CO_sum']= CO_sum


# ## kf_CO2 = 10^09

# In[ ]:


data_00


# In[ ]:


x_values = data_00['Time(s)']
y_values = data_00['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_00['Time(s)']
```

```
y_values = data_00['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_00['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+09')
plt.plot(data_00['Time(s)'], data_00['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_c12e6_kf9.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^10

# In[ ]:


x_values = data_01['Time(s)']
y_values = data_01['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_01['Time(s)']
y_values = data_01['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_01['Time(s)'], data_01['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+10')
plt.plot(data_01['Time(s)'], data_01['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_01['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_c12e6_kf10.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^11

# In[ ]:


x_values = data_02['Time(s)']
y_values = data_02['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_02['Time(s)']
y_values = data_02['CO_sum']
```

```
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_02['Time(s)'], data_02['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+11')
plt.plot(data_02['Time(s)'], data_02['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_02['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_c12e6_kf11.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^12

# In[ ]:


x_values = data_03['Time(s)']
y_values = data_03['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_03['Time(s)']
y_values = data_03['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_03['Time(s)'], data_03['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+12')
plt.plot(data_03['Time(s)'], data_03['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_03['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_c12e6_kf12.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^13

# In[ ]:


x_values = data_04['Time(s)']
y_values = data_04['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_04['Time(s)']
y_values = data_04['CO_sum']
def objective(x, a, b, c):
```

```
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_04['Time(s)'], data_04['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+13')
plt.plot(data_04['Time(s)'], data_04['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_04['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_c12e6_kf13.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^14

# In[ ]:


x_values = data_05['Time(s)']
y_values = data_05['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_05['Time(s)']
y_values = data_05['CO_sum']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_05['Time(s)'], data_05['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+14')
plt.plot(data_05['Time(s)'], data_05['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_05['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_c12e6_kf14.PNG',bbox_inches = "tight")


# ## kf_CO2 = 10^15

# In[ ]:


x_values = data_06['Time(s)']
y_values = data_06['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_06['Time(s)']
y_values = data_06['CO_sum']
def objective(x, a, b, c):
    return a * x + b
```

```
popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_06['Time(s)'], data_06['CO_sum'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+15')
plt.plot(data_06['Time(s)'], data_06['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_06['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_c12e6_kf15.PNG',bbox_inches = "tight")
```

FIT forward reaction rate test number 2

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import seaborn as sns
sns.set(font_scale = 1.3)
sns.set_style("white")


# In[ ]:


data = pd.read_fwf('fitting_kf_piccoli.txt')


# In[ ]:


data['kf_CO2'].unique()


# In[ ]:


data0 = data[data['kf_CO2'] == data['kf_CO2'].unique()[0]]
data1 = data[data['kf_CO2'] == data['kf_CO2'].unique()[1]]
data2 = data[data['kf_CO2'] == data['kf_CO2'].unique()[2]]
data3 = data[data['kf_CO2'] == data['kf_CO2'].unique()[3]]
data4 = data[data['kf_CO2'] == data['kf_CO2'].unique()[4]]
data5 = data[data['kf_CO2'] == data['kf_CO2'].unique()[5]]
data6 = data[data['kf_CO2'] == data['kf_CO2'].unique()[6]]
data7 = data[data['kf_CO2'] == data['kf_CO2'].unique()[7]]


# In[ ]:


data0.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data1.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data2.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data3.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data4.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data5.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data6.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data7.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)


# In[ ]:


data_0 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 2446],data0[data0['Time(s)'] == 7274],
                    data0[data0['Time(s)'] == 10776],data0[data0['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_1 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 2446],data1[data1['Time(s)'] == 7274],
                    data1[data1['Time(s)'] == 10776],data1[data1['Time(s)'] == 13196]], axis = 0)


# In[ ]:
```

```

data_2 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 2446],data2[data2['Time(s)'] == 7274],
                    data2[data2['Time(s)'] == 10776],data2[data2['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_3 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 2446],data3[data3['Time(s)'] == 7274],
                    data3[data3['Time(s)'] == 10776],data3[data3['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_4 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 2446],data4[data4['Time(s)'] == 7274],
                    data4[data4['Time(s)'] == 10776],data4[data4['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_5 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 2446],data5[data5['Time(s)'] == 7274],
                    data5[data5['Time(s)'] == 10776],data5[data5['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_6 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 2446],data6[data6['Time(s)'] == 7274],
                    data6[data6['Time(s)'] == 10776],data6[data6['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_7 = pd.concat([data7[data7['Time(s)'] == 0],data7[data7['Time(s)'] == 2446],data7[data7['Time(s)'] == 7274],
                    data7[data7['Time(s)'] == 10776],data7[data7['Time(s)'] == 13196]], axis = 0)


# In[ ]:


Value_to_add = [0.00000000, 0.00000261,0.00000507,0.00000737,0.0000102]


# In[ ]:


data_0['experimental'] = Value_to_add
data_1['experimental'] = Value_to_add
data_2['experimental'] = Value_to_add
data_3['experimental'] = Value_to_add
data_4['experimental'] = Value_to_add
data_5['experimental'] = Value_to_add
data_6['experimental'] = Value_to_add
data_7['experimental'] = Value_to_add


# In[ ]:


CO_sum = data_0[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_0['CO_sum']= CO_sum
CO_sum = data_1[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_1['CO_sum']= CO_sum
CO_sum = data_2[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_2['CO_sum']= CO_sum
CO_sum = data_3[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_3['CO_sum']= CO_sum
CO_sum = data_4[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_4['CO_sum']= CO_sum
CO_sum = data_5[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_5['CO_sum']= CO_sum
CO_sum = data_6[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_6['CO_sum']= CO_sum
CO_sum = data_7[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_7['CO_sum']= CO_sum


# In[ ]:


data_0['Ratio_g_s']= data_0['CO_conc_gas']/data_0['CO_sum']
data_1['Ratio_g_s']= data_1['CO_conc_gas']/data_1['CO_sum']
data_2['Ratio_g_s']= data_2['CO_conc_gas']/data_2['CO_sum']
data_3['Ratio_g_s']= data_3['CO_conc_gas']/data_3['CO_sum']
data_4['Ratio_g_s']= data_4['CO_conc_gas']/data_4['CO_sum']
data_5['Ratio_g_s']= data_5['CO_conc_gas']/data_5['CO_sum']
data_6['Ratio_g_s']= data_6['CO_conc_gas']/data_6['CO_sum']
data_7['Ratio_g_s']= data_7['CO_conc_gas']/data_7['CO_sum']


# ## kf_CO2 = 10^1

# In[ ]:


data_0


# In[ ]:


```

```
x_values = data_0['Time(s)']
y_values = data_0['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_0['Time(s)']
y_values = data_0['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_0['Time(s)'], data_0['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+01')
plt.plot(data_0['Time(s)'], data_0['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_0['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^2

# In[ ]:


data_1


# In[ ]:


x_values = data_1['Time(s)']
y_values = data_1['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_1['Time(s)']
y_values = data_1['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_1['Time(s)'], data_1['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+02')
plt.plot(data_1['Time(s)'], data_1['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^3

```

```
# In[ ]:


data_2


# In[ ]:


x_values = data_2['Time(s)']
y_values = data_2['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_2['Time(s)']
y_values = data_2['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_2['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+03')
plt.plot(data_2['Time(s)'], data_2['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^4

# In[ ]:


data_3


# In[ ]:


x_values = data_3['Time(s)']
y_values = data_3['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_3['Time(s)']
y_values = data_3['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_3['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+04')
plt.plot(data_2['Time(s)'], data_3['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
```

```
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# In[ ]:


##kf_CO2 = 10^5


# In[ ]:


data_4


# In[ ]:


x_values = data_4['Time(s)']
y_values = data_4['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_4['Time(s)']
y_values = data_4['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_4['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+05')
plt.plot(data_2['Time(s)'], data_4['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^6

# In[ ]:


data_5


# In[ ]:


x_values = data_5['Time(s)']
y_values = data_5['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_5['Time(s)']
y_values = data_5['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
```

```
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_5['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+06')
plt.plot(data_2['Time(s)'], data_5['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^7

# In[ ]:


data_6


# In[ ]:


x_values = data_6['Time(s)']
y_values = data_6['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_6['Time(s)']
y_values = data_6['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_6['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+07')
plt.plot(data_2['Time(s)'], data_6['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^8

# In[ ]:


data_7


# In[ ]:


x_values = data_7['Time(s)']
y_values = data_7['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_7['Time(s)']
```

```
y_values = data_7['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_7['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+08')
plt.plot(data_2['Time(s)'], data_7['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_1['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


#

#

#

#

# In[ ]:


data_00 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 1731],data0[data0['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_01 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 1731],data1[data1['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_02 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 1731],data2[data2['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_03 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 1731],data3[data3['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_04 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 1731],data4[data4['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_05 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 1731],data5[data5['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_06 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 1731],data6[data6['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_07 = pd.concat([data7[data7['Time(s)'] == 0],data7[data7['Time(s)'] == 1731],data7[data7['Time(s)'] == 3896]],axis = 0)


# In[ ]:


Value_to_add2 = [0.00000000, 0.00000559, 0.00000814]


# In[ ]:


data_00['experimental'] = Value_to_add2
data_01['experimental'] = Value_to_add2
data_02['experimental'] = Value_to_add2
data_03['experimental'] = Value_to_add2
data_04['experimental'] = Value_to_add2
data_05['experimental'] = Value_to_add2
data_06['experimental'] = Value_to_add2
data_07['experimental'] = Value_to_add2


# In[ ]:
```

```
CO_sum = data_00[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_00['CO_sum']= CO_sum
CO_sum = data_01[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_01['CO_sum']= CO_sum
CO_sum = data_02[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_02['CO_sum']= CO_sum
CO_sum = data_03[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_03['CO_sum']= CO_sum
CO_sum = data_04[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_04['CO_sum']= CO_sum
CO_sum = data_05[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_05['CO_sum']= CO_sum
CO_sum = data_06[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_06['CO_sum']= CO_sum
CO_sum = data_07[['CO_conc_gas','CO_conc_liq','CO_surf_conc']].sum(axis=1)
data_07['CO_sum']= CO_sum


# In[ ]:


data_00['Ratio_g_s']= data_00['CO_conc_gas']/data_00['CO_sum']
data_01['Ratio_g_s']= data_01['CO_conc_gas']/data_01['CO_sum']
data_02['Ratio_g_s']= data_02['CO_conc_gas']/data_02['CO_sum']
data_03['Ratio_g_s']= data_03['CO_conc_gas']/data_03['CO_sum']
data_04['Ratio_g_s']= data_04['CO_conc_gas']/data_04['CO_sum']
data_05['Ratio_g_s']= data_05['CO_conc_gas']/data_05['CO_sum']
data_06['Ratio_g_s']= data_06['CO_conc_gas']/data_06['CO_sum']
data_07['Ratio_g_s']= data_07['CO_conc_gas']/data_07['CO_sum']


# ## kf_CO2 = 10^1

# In[ ]:


data_00


# In[ ]:


x_values = data_00['Time(s)']
y_values = data_00['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp = x_values
y_exp = y_new


# In[ ]:


x_values = data_00['Time(s)']
y_values = data_00['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_00['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+01')
plt.plot(data_00['Time(s)'], data_00['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^2

# In[ ]:


data_01


# In[ ]:


x_values = data_01['Time(s)']
y_values = data_01['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b
```

```
popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_01['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+02')
plt.plot(data_00['Time(s)'], data_01['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^3

# In[ ]:


data_02


# In[ ]:


x_values = data_02['Time(s)']
y_values = data_02['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_02['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+03')
plt.plot(data_00['Time(s)'], data_02['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^4

# In[ ]:


data_03


# In[ ]:


x_values = data_03['Time(s)']
y_values = data_03['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_03['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+04')
plt.plot(data_00['Time(s)'], data_03['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
```

```
plt.legend()


# ## kf_CO2 = 10^5

# In[ ]:


data_04


# In[ ]:


x_values = data_04['Time(s)']
y_values = data_04['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_04['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+05')
plt.plot(data_00['Time(s)'], data_04['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^6

# In[ ]:


data_05


# In[ ]:


x_values = data_05['Time(s)']
y_values = data_05['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_05['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+06')
plt.plot(data_00['Time(s)'], data_05['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^7

# In[ ]:


data_06


# In[ ]:


x_values = data_06['Time(s)']
y_values = data_06['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)
```

```
x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_06['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+07')
plt.plot(data_00['Time(s)'], data_06['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## kf_CO2 = 10^8

# In[ ]:


data_07


# In[ ]:


x_values = data_07['Time(s)']
y_values = data_07['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com = x_values
y_com = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_07['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+08')
plt.plot(data_00['Time(s)'], data_07['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com, y_com, 'b--', label = 'Comsol fit')
plt.plot(x_exp, y_exp, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# In[ ]:
```

FIT forward reaction rate test number 3

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import seaborn as sns
sns.set(font_scale = 1.3)
sns.set_style("white")


# In[ ]:


data = pd.read_fwf('fitting_kf_preciso.txt')


# In[ ]:


data


# In[ ]:


data['kf_CO2'].unique()


# In[ ]:


data0 = data[data['kf_CO2'] == data['kf_CO2'].unique()[0]]
data1 = data[data['kf_CO2'] == data['kf_CO2'].unique()[1]]
```

```
data2 = data[data['kf_CO2'] == data['kf_CO2'].unique()[2]]
data3 = data[data['kf_CO2'] == data['kf_CO2'].unique()[3]]
data4 = data[data['kf_CO2'] == data['kf_CO2'].unique()[4]]
data5 = data[data['kf_CO2'] == data['kf_CO2'].unique()[5]]
data6 = data[data['kf_CO2'] == data['kf_CO2'].unique()[6]]
data7 = data[data['kf_CO2'] == data['kf_CO2'].unique()[7]]
data8 = data[data['kf_CO2'] == data['kf_CO2'].unique()[8]]
data9 = data[data['kf_CO2'] == data['kf_CO2'].unique()[9]]


# In[ ]:


data0.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data1.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data2.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data3.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data4.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data5.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data6.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data7.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data8.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data9.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)


# In[ ]:


data_0 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 2446],data0[data0['Time(s)'] == 7274],
                    data0[data0['Time(s)'] == 10776],data0[data0['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_1 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 2446],data1[data1['Time(s)'] == 7274],
                    data1[data1['Time(s)'] == 10776],data1[data1['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_2 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 2446],data2[data2['Time(s)'] == 7274],
                    data2[data2['Time(s)'] == 10776],data2[data2['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_3 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 2446],data3[data3['Time(s)'] == 7274],
                    data3[data3['Time(s)'] == 10776],data3[data3['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_4 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 2446],data4[data4['Time(s)'] == 7274],
                    data4[data4['Time(s)'] == 10776],data4[data4['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_5 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 2446],data5[data5['Time(s)'] == 7274],
                    data5[data5['Time(s)'] == 10776],data5[data5['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_6 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 2446],data6[data6['Time(s)'] == 7274],
                    data6[data6['Time(s)'] == 10776],data6[data6['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_7 = pd.concat([data7[data7['Time(s)'] == 0],data7[data7['Time(s)'] == 2446],data7[data7['Time(s)'] == 7274],
                    data7[data7['Time(s)'] == 10776],data7[data7['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_8 = pd.concat([data8[data8['Time(s)'] == 0],data8[data8['Time(s)'] == 2446],data8[data8['Time(s)'] == 7274],
                    data8[data8['Time(s)'] == 10776],data8[data8['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_9 = pd.concat([data9[data9['Time(s)'] == 0],data9[data9['Time(s)'] == 2446],data9[data9['Time(s)'] == 7274],
                    data9[data9['Time(s)'] == 10776],data9[data9['Time(s)'] == 13196]], axis = 0)


```

```
# In[ ]:


Value_to_add = [0.00000000, 0.00000261,0.00000507,0.00000737,0.0000102]


# In[ ]:


data_0['experimental'] = Value_to_add
data_1['experimental'] = Value_to_add
data_2['experimental'] = Value_to_add
data_3['experimental'] = Value_to_add
data_4['experimental'] = Value_to_add
data_5['experimental'] = Value_to_add
data_6['experimental'] = Value_to_add
data_7['experimental'] = Value_to_add
data_8['experimental'] = Value_to_add
data_9['experimental'] = Value_to_add


# In[ ]:


data_0['difference'] = data_0['CO_conc_gas']-Value_to_add
data_1['difference']= data_1['CO_conc_gas']-Value_to_add
data_2['difference']= data_2['CO_conc_gas']-Value_to_add
data_3['difference']= data_3['CO_conc_gas']-Value_to_add
data_4['difference']= data_4['CO_conc_gas']-Value_to_add
data_5['difference']= data_5['CO_conc_gas']-Value_to_add
data_6['difference']= data_6['CO_conc_gas']-Value_to_add
data_7['difference']= data_7['CO_conc_gas']-Value_to_add
data_8['difference']= data_8['CO_conc_gas']-Value_to_add
data_9['difference']= data_9['CO_conc_gas']-Value_to_add


# ## scelti data2 e data3

# In[ ]:


x_values = data_2['Time(s)']
y_values = data_2['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp1 = x_values
y_exp1 = y_new


# In[ ]:


x_values = data_2['Time(s)']
y_values = data_2['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com1 = x_values
y_com1 = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_2['Time(s)'], data_2['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 7e+07')
plt.plot(data_2['Time(s)'], data_2['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com1, y_com1, 'b--', label = 'Comsol fit')
plt.plot(x_exp1, y_exp1, 'g--', label = 'Experimental fit')

ticks = np.array(data_0['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# In[ ]:


x_values = data_3['Time(s)']
y_values = data_3['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com2 = x_values
y_com2 = y_new


# In[ ]:
```

```
plt.figure(figsize = (10,8))
plt.plot(data_3['Time(s)'], data_3['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 5e+07')
plt.plot(data_3['Time(s)'], data_3['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com2, y_com2, 'b--', label = 'Comsol fit')
plt.plot(x_exp1, y_exp1, 'g--', label = 'Experimental fit')

ticks = np.array(data_0['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


#
# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:




# In[ ]:



# ## surfactant
# In[ ]:

data_00 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 1731],data0[data0['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_01 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 1731],data1[data1['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_02 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 1731],data2[data2['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_03 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 1731],data3[data3['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_04 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 1731],data4[data4['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_05 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 1731],data5[data5['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_06 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 1731],data6[data6['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_07 = pd.concat([data7[data7['Time(s)'] == 0],data7[data7['Time(s)'] == 1731],data7[data7['Time(s)'] == 3896]],axis = 0)

# In[ ]:
```

```
data_08 = pd.concat([data8[data8['Time(s)'] == 0],data8[data8['Time(s)'] == 1731],data8[data8['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_09 = pd.concat([data9[data9['Time(s)'] == 0],data9[data9['Time(s)'] == 1731],data9[data9['Time(s)'] == 3896]],axis = 0)


# In[ ]:


Value_to_add2 = [0.00000000, 0.00000559, 0.00000814]


# In[ ]:


data_00['experimental'] = Value_to_add2
data_01['experimental'] = Value_to_add2
data_02['experimental'] = Value_to_add2
data_03['experimental'] = Value_to_add2
data_04['experimental'] = Value_to_add2
data_05['experimental'] = Value_to_add2
data_06['experimental'] = Value_to_add2
data_07['experimental'] = Value_to_add2
data_08['experimental'] = Value_to_add2
data_09['experimental'] = Value_to_add2


# In[ ]:


data_00['difference'] = data_00['CO_conc_gas']-Value_to_add2
data_01['difference'] = data_01['CO_conc_gas']-Value_to_add2
data_02['difference'] = data_02['CO_conc_gas']-Value_to_add2
data_03['difference'] = data_03['CO_conc_gas']-Value_to_add2
data_04['difference'] = data_04['CO_conc_gas']-Value_to_add2
data_05['difference'] = data_05['CO_conc_gas']-Value_to_add2
data_06['difference'] = data_06['CO_conc_gas']-Value_to_add2
data_07['difference'] = data_07['CO_conc_gas']-Value_to_add2
data_08['difference'] = data_08['CO_conc_gas']-Value_to_add2
data_09['difference'] = data_09['CO_conc_gas']-Value_to_add2


# ## scelti data_05 e data_06

# In[ ]:


x_values = data_05['Time(s)']
y_values = data_05['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp01 = x_values
y_exp01 = y_new


# In[ ]:


x_values = data_05['Time(s)']
y_values = data_05['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com01 = x_values
y_com01 = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_05['Time(s)'], data_05['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+08')
plt.plot(data_05['Time(s)'], data_05['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com01, y_com01, 'b--', label = 'Comsol fit')
plt.plot(x_exp01, y_exp01, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# In[ ]:


x_values = data_06['Time(s)']
y_values = data_06['experimental']
def objective(x, a, b, c):
    return a * x + b
```

```
popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp01 = x_values
y_exp01 = y_new


# In[ ]:


x_values = data_06['Time(s)']
y_values = data_06['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com02 = x_values
y_com02 = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_06['Time(s)'], data_06['CO_conc_gas'],'b^', markersize = 10, label = 'Comsol Model kf_CO2 = 3.e+08')
plt.plot(data_06['Time(s)'], data_06['experimental'], 'gv', markersize = 10, label = 'Experimental')
plt.plot(x_com02, y_com02, 'b--', label = 'Comsol fit')
plt.plot(x_exp01, y_exp01, 'g--', label = 'Experimental fit')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()


# ## NEW_DATA

# In[ ]:


new_data = pd.read_fwf('fitting_kf_piu_preciso.txt')


# In[ ]:


new_data


# In[ ]:


new_data0 = new_data[new_data['kf_CO2'] == new_data['kf_CO2'].unique()[0]]


# In[ ]:


new_data0


# In[ ]:


new_data1 =  new_data[new_data['kf_CO2'] == new_data['kf_CO2'].unique()[1]]


# In[ ]:


new_data_0 = pd.concat([new_data0[new_data0['Time(s)'] == 0],new_data0[new_data0['Time(s)'] == 2446],new_data0[new_data0['Time(s)'] == 7274],
                    new_data0[new_data0['Time(s)'] == 10776],new_data0[new_data0['Time(s)'] == 13196]], axis = 0)


# In[ ]:


new_data_0['experimental'] = Value_to_add


# In[ ]:


new_data_0['difference'] = new_data_0['CO_conc_gas']-Value_to_add


# In[ ]:


x_values = new_data_0['Time(s)']
y_values = new_data_0['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com_new1 = x_values
```

```
y_com_new1 = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_0['Time(s)'], data_2['CO_conc_gas'],'^',color='tab:blue', markersize = 10, label = 'Comsol Model kf_CO2 = 5.e+07')
plt.plot(data_0['Time(s)'], data_0['experimental'],'^',color='tab:green', markersize = 10, label = 'Experimental w/o C12E6')
plt.plot(data_0['Time(s)'], data_3['CO_conc_gas'],'^',color = 'tab:orange', markersize = 10, label = 'Comsol Model kf_CO2 = 7.e+07')
plt.plot(data_0['Time(s)'], new_data_0['CO_conc_gas'],'^',color = 'tab:red', markersize = 10, label = 'Comsol Model kf_CO2 = 6.e+07')


plt.plot(x_com1, y_com1, '--', color = 'tab:blue')
plt.plot(x_com2, y_com2, '--',color='tab:orange')
plt.plot(x_exp1, y_exp1,'--', color= 'tab:green')
plt.plot(x_com_new1,y_com_new1, '--', color ='tab:red' )

ticks = np.array(data_0['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_kf6e7.PNG',bbox_inches = "tight")


# ## surfactant

# In[ ]:


new_data_00 = pd.concat([new_data1[new_data1['Time(s)'] == 0],new_data1[new_data1['Time(s)'] == 1731],new_data1[new_data1['Time(s)'] == 3896]],axis
    = 0)


# In[ ]:


new_data_00['experimental'] = Value_to_add2


# In[ ]:


new_data_00['difference'] = new_data_00['CO_conc_gas']-Value_to_add2


# In[ ]:


x_values = new_data_00['Time(s)']
y_values = new_data_00['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com_new01 = x_values
y_com_new01 = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_05['CO_conc_gas'],'^',color='tab:blue', markersize = 10, label = 'Comsol Model kf_CO2 = 1.e+08')
plt.plot(new_data_00['Time(s)'], new_data_00['experimental'],'^',color='tab:green', markersize = 10, label = 'Experimental \w C12E6')
plt.plot(data_00['Time(s)'], data_06['CO_conc_gas'],'^',color = 'tab:orange', markersize = 10, label = 'Comsol Model kf_CO2 = 3.e+08')
plt.plot(data_00['Time(s)'], new_data_00['CO_conc_gas'],'^',color = 'tab:red', markersize = 10, label = 'Comsol Model kf_CO2 = 2.e+08')


plt.plot(x_com01, y_com01, '--', color = 'tab:blue')
plt.plot(x_com02, y_com02, '--',color='tab:orange')
plt.plot(x_exp01, y_exp01,'--', color= 'tab:green')
plt.plot(x_com_new01,y_com_new01, '--', color ='tab:red' )

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_kf2e8.PNG',bbox_inches = "tight")
```

FIT equilibrium adsorption constant H2Q test number 1

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import seaborn as sns
sns.set(font_scale = 1.3)
sns.set_style("white")


```

```
# In[ ]:


data = pd.read_fwf('fitting_KHH2Q_e-3e3.txt')


# In[ ]:


data


# In[ ]:


data['KH_H2Q'].unique()


# In[ ]:


data0 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[0]]
data1 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[1]]
data2 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[2]]
data3 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[3]]
data4 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[4]]
data5 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[5]]
data6 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[6]]


# In[ ]:


data0.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data1.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data2.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data3.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data4.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data5.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data6.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)


# In[ ]:


data_0 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 2446],data0[data0['Time(s)'] == 7274],
                    data0[data0['Time(s)'] == 10776],data0[data0['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_1 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 2446],data1[data1['Time(s)'] == 7274],
                    data1[data1['Time(s)'] == 10776],data1[data1['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_2 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 2446],data2[data2['Time(s)'] == 7274],
                    data2[data2['Time(s)'] == 10776],data2[data2['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_3 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 2446],data3[data3['Time(s)'] == 7274],
                    data3[data3['Time(s)'] == 10776],data3[data3['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_4 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 2446],data4[data4['Time(s)'] == 7274],
                    data4[data4['Time(s)'] == 10776],data4[data4['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_5 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 2446],data5[data5['Time(s)'] == 7274],
                    data5[data5['Time(s)'] == 10776],data5[data5['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_6 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 2446],data6[data6['Time(s)'] == 7274],
                    data6[data6['Time(s)'] == 10776],data6[data6['Time(s)'] == 13196]], axis = 0)


# In[ ]:


Value_to_add = [0.00000000, 0.00000261,0.00000507,0.00000737,0.0000102]


# In[ ]:
```

```
data_0['experimental'] = Value_to_add
data_1['experimental'] = Value_to_add
data_2['experimental'] = Value_to_add
data_3['experimental'] = Value_to_add
data_4['experimental'] = Value_to_add
data_5['experimental'] = Value_to_add
data_6['experimental'] = Value_to_add

# In[ ]:

data_0['difference'] = data_0['CO_conc_gas']-Value_to_add
data_1['difference']= data_1['CO_conc_gas']-Value_to_add
data_2['difference']= data_2['CO_conc_gas']-Value_to_add
data_3['difference']= data_3['CO_conc_gas']-Value_to_add
data_4['difference']= data_4['CO_conc_gas']-Value_to_add
data_5['difference']= data_5['CO_conc_gas']-Value_to_add
data_6['difference']= data_6['CO_conc_gas']-Value_to_add

# In[ ]:

data_0

# In[ ]:

data_1

# In[ ]:

data_2

# In[ ]:

data_3

# In[ ]:

data_4

# In[ ]:

data_5

# In[ ]:

data_6

# ## surfactant
# In[ ]:

data_00 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 1731],data0[data0['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_01 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 1731],data1[data1['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_02 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 1731],data2[data2['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_03 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 1731],data3[data3['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_04 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 1731],data4[data4['Time(s)'] == 3896]],axis = 0)

# In[ ]:

data_05 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 1731],data5[data5['Time(s)'] == 3896]],axis = 0)

# In[ ]:
```

```
data_06 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 1731],data6[data6['Time(s)'] == 3896]],axis = 0)


# In[ ]:


Value_to_add2 = [0.00000000, 0.00000559, 0.00000814]


# In[ ]:


data_00['experimental'] = Value_to_add2
data_01['experimental'] = Value_to_add2
data_02['experimental'] = Value_to_add2
data_03['experimental'] = Value_to_add2
data_04['experimental'] = Value_to_add2
data_05['experimental'] = Value_to_add2
data_06['experimental'] = Value_to_add2


# In[ ]:


data_00['difference'] = data_00['CO_conc_gas']-Value_to_add2
data_01['difference'] = data_01['CO_conc_gas']-Value_to_add2
data_02['difference'] = data_02['CO_conc_gas']-Value_to_add2
data_03['difference'] = data_03['CO_conc_gas']-Value_to_add2
data_04['difference'] = data_04['CO_conc_gas']-Value_to_add2
data_05['difference'] = data_05['CO_conc_gas']-Value_to_add2
data_06['difference'] = data_06['CO_conc_gas']-Value_to_add2


# In[ ]:


data_00


# In[ ]:


data_01


# In[ ]:


data_02


# In[ ]:


data_03


# In[ ]:


data_04


# In[ ]:


data_05


# In[ ]:


data_06


# #data_00.to_pickle('dataframe_e-03.csv')
```

FIT equilibrium adsorption constant H2Q test number 1

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import seaborn as sns
sns.set(font_scale = 1.3)
sns.set_style("white")


# In[ ]:


data = pd.read_fwf('fitting_KHH2Q_e-11e-5.txt')


# In[ ]:


```

```
data


# In[ ]:


data['KH_H2Q'].unique()


# In[ ]:


data0 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[0]]
data1 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[1]]
data2 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[2]]
data3 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[3]]
data4 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[4]]
data5 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[5]]
data6 = data[data['KH_H2Q'] == data['KH_H2Q'].unique()[6]]


# In[ ]:


data0.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data1.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data2.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data3.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data4.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data5.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)
data6.drop(['CO2_conc_gas', 'CO2_conc_liq', 'H2Q_conc_liq', 'HQ_conc_liq', 'CO2_surf_conc',
            'H2Q_surf_conc', 'HQ_surf_conc'], axis = 1, inplace = True)


# In[ ]:


data_0 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 2446],data0[data0['Time(s)'] == 7274],
                    data0[data0['Time(s)'] == 10776],data0[data0['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_1 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 2446],data1[data1['Time(s)'] == 7274],
                    data1[data1['Time(s)'] == 10776],data1[data1['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_2 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 2446],data2[data2['Time(s)'] == 7274],
                    data2[data2['Time(s)'] == 10776],data2[data2['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_3 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 2446],data3[data3['Time(s)'] == 7274],
                    data3[data3['Time(s)'] == 10776],data3[data3['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_4 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 2446],data4[data4['Time(s)'] == 7274],
                    data4[data4['Time(s)'] == 10776],data4[data4['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_5 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 2446],data5[data5['Time(s)'] == 7274],
                    data5[data5['Time(s)'] == 10776],data5[data5['Time(s)'] == 13196]], axis = 0)


# In[ ]:


data_6 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 2446],data6[data6['Time(s)'] == 7274],
                    data6[data6['Time(s)'] == 10776],data6[data6['Time(s)'] == 13196]], axis = 0)


# In[ ]:


Value_to_add = [0.00000000, 0.00000261,0.00000507,0.00000737,0.0000102]


# In[ ]:


data_0['experimental'] = Value_to_add
data_1['experimental'] = Value_to_add
data_2['experimental'] = Value_to_add
data_3['experimental'] = Value_to_add
data_4['experimental'] = Value_to_add
data_5['experimental'] = Value_to_add
data_6['experimental'] = Value_to_add
```

```
# In[ ]:


data_0['difference'] = data_0['CO_conc_gas']-Value_to_add
data_1['difference']= data_1['CO_conc_gas']-Value_to_add
data_2['difference']= data_2['CO_conc_gas']-Value_to_add
data_3['difference']= data_3['CO_conc_gas']-Value_to_add
data_4['difference']= data_4['CO_conc_gas']-Value_to_add
data_5['difference']= data_5['CO_conc_gas']-Value_to_add
data_6['difference']= data_6['CO_conc_gas']-Value_to_add


# In[ ]:


data_0


# In[ ]:


data_1


# In[ ]:


data_2


# In[ ]:


data_3


# In[ ]:


data_4


# In[ ]:


data_5


# In[ ]:


data_6


# ## scelti data_5 e data_6

# In[ ]:


x_values = data_5['Time(s)']
y_values = data_5['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp1 = x_values
y_exp1 = y_new


# In[ ]:


x_values = data_5['Time(s)']
y_values = data_5['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com1 = x_values
y_com1 = y_new


# In[ ]:


x_values = data_6['Time(s)']
y_values = data_6['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com2 = x_values
y_com2 = y_new
```

```
# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_0['Time(s)'], data_5['CO_conc_gas'],'^',color='tab:blue', markersize = 10, label = 'Comsol Model KH_H2Q = 9.91e-6')
plt.plot(data_0['Time(s)'], data_0['experimental'],'^',color='tab:green', markersize = 10, label = 'Experimental w/o C12E6')
plt.plot(data_0['Time(s)'], data_6['CO_conc_gas'],'^',color = 'tab:orange', markersize = 10, label = 'Comsol Model KH_H2Q = 9.91e-5')


plt.plot(x_com1, y_com1, '--', color = 'tab:blue')
plt.plot(x_com2, y_com2, '--',color='tab:orange')
plt.plot(x_exp1, y_exp1,'--', color= 'tab:green')

ticks = np.array(data_0['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_KHH2Qe-5.PNG',bbox_inches = "tight")


# ## Surfactant

# In[ ]:


data_00 = pd.concat([data0[data0['Time(s)'] == 0],data0[data0['Time(s)'] == 1731],data0[data0['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_01 = pd.concat([data1[data1['Time(s)'] == 0],data1[data1['Time(s)'] == 1731],data1[data1['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_02 = pd.concat([data2[data2['Time(s)'] == 0],data2[data2['Time(s)'] == 1731],data2[data2['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_03 = pd.concat([data3[data3['Time(s)'] == 0],data3[data3['Time(s)'] == 1731],data3[data3['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_04 = pd.concat([data4[data4['Time(s)'] == 0],data4[data4['Time(s)'] == 1731],data4[data4['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_05 = pd.concat([data5[data5['Time(s)'] == 0],data5[data5['Time(s)'] == 1731],data5[data5['Time(s)'] == 3896]],axis = 0)


# In[ ]:


data_06 = pd.concat([data6[data6['Time(s)'] == 0],data6[data6['Time(s)'] == 1731],data6[data6['Time(s)'] == 3896]],axis = 0)


# In[ ]:


Value_to_add2 = [0.00000000, 0.00000559, 0.00000814]


# In[ ]:


data_00['experimental'] = Value_to_add2
data_01['experimental'] = Value_to_add2
data_02['experimental'] = Value_to_add2
data_03['experimental'] = Value_to_add2
data_04['experimental'] = Value_to_add2
data_05['experimental'] = Value_to_add2
data_06['experimental'] = Value_to_add2


# In[ ]:


data_00['difference'] = data_00['CO_conc_gas']-Value_to_add2
data_01['difference'] = data_01['CO_conc_gas']-Value_to_add2
data_02['difference'] = data_02['CO_conc_gas']-Value_to_add2
data_03['difference'] = data_03['CO_conc_gas']-Value_to_add2
data_04['difference'] = data_04['CO_conc_gas']-Value_to_add2
data_05['difference'] = data_05['CO_conc_gas']-Value_to_add2
data_06['difference'] = data_06['CO_conc_gas']-Value_to_add2


# In[ ]:


data_00


# In[ ]:
```

```
data_01


# In[ ]:


data_02


# In[ ]:


data_03


# In[ ]:


data_04


# In[ ]:


data_05


# In[ ]:


data_06


# ## scelti data_06 e quello che sarebbe e-03

# In[ ]:


data_07 = pd.read_pickle('dataframe_e-03.csv')


# In[ ]:


data_07


# In[ ]:


x_values = data_06['Time(s)']
y_values = data_06['experimental']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_exp1 = x_values
y_exp1 = y_new


# In[ ]:


x_values = data_06['Time(s)']
y_values = data_06['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com1 = x_values
y_com1 = y_new


# In[ ]:


x_values = data_07['Time(s)']
y_values = data_07['CO_conc_gas']
def objective(x, a, b, c):
    return a * x + b

popt, _ = curve_fit(objective, x_values, y_values)
a, b , c = popt
y_new = objective(x_values, a, b, c)

x_com2 = x_values
y_com2 = y_new


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_06['CO_conc_gas'],'^',color='tab:blue', markersize = 10, label = 'Comsol Model KH_H2Q = 9.91e-4')
plt.plot(data_00['Time(s)'], data_00['experimental'],'^',color='tab:green', markersize = 10, label = 'Experimental w/o C12E6')
plt.plot(data_00['Time(s)'], data_07['CO_conc_gas'],'^',color = 'tab:orange', markersize = 10, label = 'Comsol Model KH_H2Q = 9.91e-3')
```

```
plt.plot(x_com1, y_com1, '--', color = 'tab:blue')
plt.plot(x_com2, y_com2, '--',color='tab:orange')
plt.plot(x_exp1, y_exp1,'--', color= 'tab:green')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_KHH2Qe-4.PNG',bbox_inches = "tight")


# In[ ]:


plt.figure(figsize = (10,8))
plt.plot(data_00['Time(s)'], data_06['CO_conc_gas'],'^',color='tab:blue', markersize = 10, label = 'Comsol Model KH_H2Q = 9.91e-4')
plt.plot(data_00['Time(s)'], data_00['experimental'],'^',color='tab:green', markersize = 10, label = 'Experimental w/o C12E6')
#plt.plot(data_00['Time(s)'], data_07['CO_conc_gas'],'^',color = 'tab:orange', markersize = 10, label = 'Comsol Model KH_H2Q = 9.91e-3')


plt.plot(x_com1, y_com1, '--', color = 'tab:blue')
#plt.plot(x_com2, y_com2, '--',color='tab:orange')
plt.plot(x_exp1, y_exp1,'--', color= 'tab:green')

ticks = np.array(data_00['Time(s)'])
plt.xticks(ticks)
plt.title ('CO concentration: Model and Experimental', fontweight = "bold", size = 15 )
plt.ylabel('CO number of moles [mol/m^2]', size = 15)
plt.xlabel(' Time  [s]', size = 15)
plt.legend()
plt.savefig('CO_KHH2Qe-4_take2.PNG',bbox_inches = "tight")
```

Bibliography

[1] Florinda Martins, Carlos Felgueiras, and Miroslava Smitková. Fossil fuel energy consumption in european countries. *Energy Procedia*, 153:107–111, 2018. 5th International Conference on Energy and Environment Research, ICEER 2018, 23-27 July 2018, Prague, Czech Republic.

[2] International Energy Agency IEA. World energy outlook 2021. https://www.iea.org/reports/world-energy-outlook-2021, 2021.

[3] J Kim. C. a. henao, t. a. johnson, de dedrick, je miller, eb stechel, and ct maravelias,"methanol production from co2 using solar-thermal energy: process development and technoeconomic analysis,". *Energy Environ. Sci*, 4(9):3122, 2011.

[4] Helmut Tributsch. Photovoltaic hydrogen generation. *International Journal of Hydrogen Energy*, 33(21):5911–5930, 2008.

[5] Zhongliang Zhan, Worawarit Kobsiriphat, James R Wilson, Manoj Pillai, Ilwon Kim, and Scott A Barnett. Syngas production by coelectrolysis of co2/h2o: the basis for a renewable energy cycle. *Energy & Fuels*, 23(6):3089–3096, 2009.

[6] SF Ahmed, N Rafa, M Mofijur, IA Badruddin, A Inayat, MS Ali, O Farrok, and TM Yunus Khan. Biohydrogen production from biomass sources: Metabolic pathways and economic analysis. front. *Energy Res*, 9:753878, 2021.

[7] RJ Detz, JNH Reek, and BCC Van Der Zwaan. The future of solar fuels: when could they become competitive? *Energy & Environmental Science*, 11(7):1653–1669, 2018.

[8] Stenbjörn Styring. Artificial photosynthesis for solar fuels. *Faraday discussions*, 155:357–376, 2012.

[9] Robert E Blankenship. *Molecular mechanisms of photosynthesis*. John Wiley and Sons, 2021.

[10] Yatendra S Chaudhary. *SOLAR FUEL GENERATION*. 01 2017.

[11] JAMES R. BOLTON and DAVID O. HALL. The maximum efficiency of photosynthesis *. *Photochemistry and Photobiology*, 53(4):545–548, 1991.

[12] Robert Blankenship, David Tiede, James Barber, Gary Brudvig, Graham Fleming, Maria Ghirardi, Marilyn Gunner, Wolfgang Junge, David Kramer, Anastasios Melis, Thomas Moore, Christopher Moser, Daniel Nocera, A. Nozik, Donald Ort, William Parson, Roger Prince, and Richard Sayre. Comparing photosynthetic and photovoltaic efficiencies and recognizing the potential for improvement. *Science (New York, N.Y.)*, 332:805–9, 05 2011.

[13] Jackson Streeter, Luis De Taboada, and Uri Oron. Mechanisms of action of light therapy on acute myocardial infarction and stroke. *Mitochondrion*, 4:569–76, 10 2004.

[14] Esam M.A. Hussein. Chapter one - mechanisms. In Esam M.A. Hussein, editor, *Radiation Mechanics*, pages 1–65. Elsevier Science Ltd, Oxford, 2007.

[15] J. S. Griffith and L. E. Orgel. Ligand-field theory. *Q. Rev. Chem. Soc.*, 11:381–393, 1957.

[16] Bo You and Yujie Sun. Innovative strategies for electrocatalytic water splitting. *Accounts of Chemical Research*, 51(7):1571–1580, 2018. PMID: 29537825.

[17] Jean-Marie Lehn and Raymond Ziessel. Photochemical generation of carbon monoxide and hydrogen by reduction of carbon dioxide and water under visible light irradiation. *Proceedings of the National Academy of Sciences*, 79(2):701–704, 1982.

[18] P A Cox. *The electronic structure and chemistry of solids*. Clarendon Press, Oxford, England, June 1987.

[19] Ivan Pelant and Jan Valenta. Luminescence of disordered semiconductors. In *Luminescence Spectroscopy of Semiconductors*, pages 242–262. Oxford University Press, February 2012.

[20] I. P-Type, N-Type Semiconductors, jul 6 2021. [Online; accessed 2022-03-05].

[21] D. Gust. Chapter one - an illustrative history of artificial photosynthesis. In Bruno Robert, editor, *Artificial Photosynthesis*, volume 79 of *Advances in Botanical Research*, pages 1–42. Academic Press, 2016.

[22] Takashi Hisatomi, Jun Kubota, and Kazunari Domen. Recent advances in semiconductors for photocatalytic and photoelectrochemical water splitting. *Chemical Society Reviews*, 43(22):7520–7535, 2014.

[23] Joseph H Montoya, Linsey C Seitz, Pongkarn Chakthranont, Aleksandra Vojvodic, Thomas F Jaramillo, and Jens K Nørskov. Materials for solar fuels and chemicals. *Nature materials*, 16(1):70–81, 2017.

[24] Akihiko Kudo and Yugo Miseki. Heterogeneous photocatalyst materials for water splitting. *Chem. Soc. Rev.*, 38:253–278, 2009.

[25] Jingrun Ran, Mietek Jaroniec, and Shi-Zhang Qiao. Cocatalysts in semiconductor-based photocatalytic co2 reduction: Achievements, challenges, and opportunities. *Advanced Materials*, 30(7):1704649, 2018.

[26] Juan Amaro-Gahete, Mariia V. Pavliuk, Haining Tian, Dolores Esquivel, Francisco J. Romero-Salguero, and Sascha Ott. Catalytic systems mimicking the [fefe]-hydrogenase active site for visible-light-driven hydrogen production. *Coordination Chemistry Reviews*, 448:214172, 2021.

[27] A. Juris, V. Balzani, F. Barigelletti, S. Campagna, P. Belser, and A. von Zelewsky. Ru(ii) polypyridine complexes: photophysics, photochemistry, eletrochemistry, and chemiluminescence. *Coordination Chemistry Reviews*, 84:85–277, 1988.

[28] Markus Kärkäs, Tanja Laine, Eric Johnston, and Björn Åkermark. *Visible Light-Driven Water Oxidation Catalyzed by Ruthenium Complexes*, pages 189–219. 03 2016.

[29] Sebastian Kozuch and Jan M. L. Martin. "turning over" definitions in catalytic cycles. *ACS Catalysis*, 2(12):2787–2794, 2012.

[30] Akira Fujishima and Kenichi Honda. Electrochemical photolysis of water at a semiconductor electrode. *nature*, 238(5358):37–38, 1972.

[31] Steven Y Reece, Jonathan A Hamel, Kimberly Sung, Thomas D Jarvi, Arthur J Esswein, Joep JH Pijpers, and Daniel G Nocera. Wireless solar water splitting using silicon-based semiconductors and earth-abundant catalysts. *science*, 334(6056):645–648, 2011.

[32] Wen-Hui Cheng, Matthias H. Richter, Matthias M. May, Jens Ohlmann, David Lackner, Frank Dimroth, Thomas Hannappel, Harry A. Atwater, and Hans-Joachim Lewerenz. Monolithic photoelectrochemical device for direct water splitting with 19% efficiency. *ACS Energy Letters*, 3(8):1795–1800, 2018.

[33] David M. Fabian, Shu Hu, Nirala Singh, Frances A. Houle, Takashi Hisatomi, Kazunari Domen, Frank E. Osterloh, and Shane Ardo. Particle suspension reactors and materials for solar-driven water splitting. *Energy Environ. Sci.*, 8:2825–2850, 2015.

[34] Xian Zhang, Mihaela Cibian, Arnau Call, Kosei Yamauchi, and Ken Sakai. Photochemical co2 reduction driven by water-soluble copper(i) photosensitizer with the catalysis accelerated by multi-electron chargeable cobalt porphyrin. *ACS Catalysis*, 9(12):11263–11273, 2019.

[35] Anna Stikane, Ee Taek Hwang, Emma V. Ainsworth, Samuel E. H. Piper, Kevin Critchley, Julea N. Butt, Erwin Reisner, and Lars J. C. Jeuken. Towards compartmentalized photocatalysis: multihaem proteins as transmembrane molecular electron conduits. *Faraday Discuss.*, 215:26–38, 2019.

[36] Daryl L Logan. *A first course in the finite element method*. Cengage Learning, 2016.

[37] JN Reddy. *An introduction to the finite element method*, volume 1221. McGraw-Hill New York, 2004.

[38] Massimiliano Zanin, Nadim Atiya, José Basílio, Jan Baumach, Arriel Benis, Chandan Behera, Magda Bucholc, Filippo Castiglione, Ioanna Chouvarda, Blandine Comte, Tien-Tuan Dao, Xuemei Ding, Estelle Pujos-Guillot, Nenad Filipovic, David Finn, David Glass, Nissim Harel, Tomas lesmantas, Ilinka Ivanoska, and Harald Schmidt. An early stage researcher's primer on systems medicine terminology. 4:2–50, 11 2020.

[39] Indraneel Sen. Sofia project. //https://sofiaproject.eu/, 2020.

[40] Vance Bergeron. Forces and structure in thin liquid soap films. *Journal of Physics: Condensed Matter*, 11(19):R215–R238, jan 1999.

[41] *Surfactants, Micelles, Emulsions, and Foams*, chapter 12, pages 246–279. John Wiley Sons, Ltd, 2003.

[42] John C Berg. The role of surfactants. In *Textile Science and Technology*, volume 13, pages 149–198. Elsevier, 2002.

[43] Weizhen Huang, Julian Iseringhausen, Tom Kneiphof, Ziyin Qu, Chenfanfu Jiang, and Matthias B. Hullin. Chemomechanical simulation of soap film flow on spherical bubbles. *ACM Trans. Graph.*, 39(4), jul 2020.

[44] David G. Whitten. Photochemical reactions of surfactant molecules in condensed monolayer assemblies—environmental control and modification of reactivity. *Angewandte Chemie International Edition in English*, 18(6):440–450, 1979.

[45] Gabriele Falciani, Ricardo Franklin, Alain Cagna, Indraneel Sen, Ali Hassanali, and Eliodoro Chiavazzo. A multi-scale perspective of gas transport through soap-film membranes. *Mol. Syst. Des. Eng.*, 5:911–921, 2020.

[46] Alexandre Mamane, Eloise Chevallier, Ludovic Olanier, François Lequeux, and Cécile Monteux. Optical control of surface forces and instabilities in foam films using photosurfactants. *Soft Matter*, 13(6):1299–1305, 2017.

[47] Marc T. Thompson. Chapter 3 - review of diode physics and the ideal (and later, nonideal) diode. In Marc T. Thompson, editor, *Intuitive Analog Circuit Design (Second Edition)*, pages 53–86. Newnes, Boston, second edition edition, 2014.

[48] Dr. Adolph Fick. V. on liquid diffusion. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 10(63):30–39, 1855.

[49] *Thermodynamics of Interfaces*, chapter 3, pages 26–41. John Wiley Sons, Ltd, 2003.

[50] *Adsorption*, chapter 9, pages 177–205. John Wiley Sons, Ltd, 2003.

[51] Leszek Czepirski, Mieczyslaw R Balys, and Ewa Komorowska-Czepirska. Some generalization of langmuir adsorption isotherm. *Internet Journal of Chemistry*, 3(14):1099–8292, 2000.

[52] Lingling Liu, Xu-Biao Luo, Lin Ding, and Sheng-Lian Luo. 4 - application of nanotechnology in the removal of heavy metal from water. In Xubiao Luo and Fang Deng, editors, *Nanomaterials for the Removal of Pollutants and Resource Reutilization*, Micro and Nano Technologies, pages 83–147. Elsevier, 2019.

[53] D. J. Donaldson. Adsorption of atmospheric gases at the airwater interface. i. nh3. *The Journal of Physical Chemistry A*, 103(1):62–70, 1999.

[54] Jan H van Driel and Wolfgang Gräber. The teaching and learning of chemical equilibrium. In *Chemical education: Towards research-based practice*, pages 271–292. Springer, 2002.

[55] E.T. Denisov, O.M. Sarkisov, and G.I. Likhtenshtein. Chapter 1 - general ideas of chemical kinetics. In E.T. Denisov, O.M. Sarkisov, and G.I. Likhtenshtein, editors, *Chemical Kinetics*, pages 1–15. Elsevier Science, Amsterdam, 2003.

[56] Eliodoro Chiavazzo and Gabriele Falciani. Database of transport coefficients in Matlab environment, June 2019.

[57] Qi Li, Kazumasa Ito, Zhishen Wu, Christopher S. Lowry, and Steven P. Loheide II. Comsol multiphysics: A novel approach to ground water modeling. *Groundwater*, 47(4):480–487, 2009.

[58] Klaus-Jürgen Bathe. *Finite Element Method*, pages 1–12. John Wiley Sons, Ltd, 2008.

[59] Juan Du, Joseph J Cullen, and Garry R Buettner. Ascorbic acid: chemistry, biology and the treatment of cancer. *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer*, 1826(2):443–457, 2012.

[60] Md. Samrat Alam, Yang Wu, and Tao Cheng. Silicate minerals as a source of arsenic contamination in groundwater. *Water Air and Soil Pollution*, 225, 11 2014.

[61] D.A. Blanco-Martínez, L. Giraldo, and J.C. Moreno-Piraján. Effect of the ph in the adsorption and in the immersion enthalpy of monohydroxylated phenols from aqueous solutions on activated carbons. *Journal of Hazardous Materials*, 169(1):291–296, 2009.

[62] Emomotimi Bamuza-Pemu and Evans Chirwa. Profile of aromatic intermediates of titanium dioxide mediated degradation of phenol. volume 35, pages 1333–1338, 01 2013.

[63] https://pubchem.ncbi.nlm.nih.gov/compound/Hydroquinone. Accessed: 2022-1-27.

[64] Mohammad Rafiee and Davood Nematollahi. Voltammetry of electroinactive species using quinone/hydroquinone redox: A known redox system viewed in a new perspective. *Electroanalysis*, 19(13):1382–1386, 2007.

[65] R. S. Nicholson and Irving. Shain. Theory of stationary electrode polarography. single scan and cyclic methods applied to reversible, irreversible, and kinetic systems. *Analytical Chemistry*, 36(4):706–723, 1964.

[66] C Aquino-Binag, PJ Pigram, RN Lamb, and PW Alexander. Surface studies of quinhydrone ph sensors. *Analytica chimica acta*, 291(1-2):65–73, 1994.

[67] Ming Cheng, Xichuan Yang, Fuguo Zhang, Jianghua Zhao, and Licheng Sun. Efficient dye-sensitized solar cells based on hydroquinone/benzoquinone as a bioinspired redox couple. *Angewandte Chemie International Edition*, 51(39):9896–9899, 2012.

[68] Josephine L.Y. Kong and Paul A. Loach. Covalently-linked porphyrin quinone complexes as rc models. In P. Leslie Dutton, Jack S. Leigh, and Antonio Scarpa, editors, *Electrons to Tissues*, pages 73–82. Academic Press, 1978.

[69] Iwao Tabushi, Noboru Koga, and Mitsuhiro Yanagita. Efficient intramolecular quenching and electron transfer in tetraphenylporphyrin attached with benzoquinone or hydroquinone as a photosystem model. *Tetrahedron Letters*, 20(3):257–260, 1979.

[70] D Gust. Intramolecular photoinduced electron-transfer reactions of porphyrins. *The porphyrin handbook*, 8:153–190, 2000.

[71] Prateek Dongare, Ying Wang, Dean M. Bass, and Thomas J. Meyer. Catalytic interconversion of the quinone/hydroquinone couple by a surface-bound os(iii/ii) polypyridyl couple. *The Journal of Physical Chemistry C*, 122(28):16189–16194, 2018.

[72] Saraf Nawar, Brian Huskinson, and Michael Aziz. Benzoquinone-hydroquinone couple for flow battery. *MRS Proceedings*, 1491:mrsf12–1491–c08–09, 2013.

[73] Sundaramurthy Suresh, Vimal Chandra Srivastava, and Indra Mani Mishra. Adsorption of catechol, resorcinol, hydroquinone, and their derivatives: a review. *International Journal of Energy and Environmental Engineering*, 3(1):1–19, 2012.

[74] Marcus Franz, Hassan A Arafat, and Neville G Pinto. Effect of chemical surface heterogeneity on the adsorption mechanism of dissolved aromatics on activated carbon. *Carbon*, 38(13):1807–1819, 2000.

[75] Kai Zhou, Jiufu Zhang, Yao Xiao, Zheng Zhao, Mingming Zhang, Lu Wang, Xiaohan Zhang, and Chunhua Zhou. High-efficiency adsorption of and competition between phenol and hydroquinone in aqueous solution on highly cationic amino-poly (vinylamine)-functionalized go-(o-mwcnts) magnetic nanohybrids. *Chemical Engineering Journal*, 389:124223, 2020.

[76] Felipe Augusto Gorla, Eduardo Henrique Duarte, Elen Romao Sartori, and César Ricardo Teixeira Tarley. Electrochemical study for the simultaneous determination of phenolic compounds and emerging pollutant using an electroanalytical sensing system based on carbon nanotubes/surfactant and multivariate approach in the optimization. *Microchemical Journal*, 124:65–75, 2016.

[77] Mohd Kotaiba Abugazleh, Benjamin Rougeau, and Hashim Ali. Adsorption of catechol and hydroquinone on titanium oxide and iron (iii) oxide. *Journal of Environmental Chemical Engineering*, 8(5):104180, 2020.

[78] Vimal C Srivastava, Mahadeva M Swamy, Indra D Mall, Basheswar Prasad, and Indra M Mishra. Adsorptive removal of phenol by bagasse fly ash and activated carbon: equilibrium, kinetics and thermodynamics. *Colloids and surfaces a: physicochemical and engineering aspects*, 272(1-2):89–104, 2006.

[79] Jianhan Huang, Kelong Huang, and Cheng Yan. Application of an easily water-compatible hypercrosslinked polymeric adsorbent for efficient removal of catechol and resorcinol in aqueous solution. *Journal of hazardous materials*, 167(1-3):69–74, 2009.

[80] Jianhan Huang, Cheng Yan, and Kelong Huang. Removal of p-nitrophenol by a water-compatible hypercrosslinked resin functionalized with formaldehyde carbonyl groups and xad-4 in aqueous solution: A comparative study. *Journal of colloid and interface science*, 332(1):60–64, 2009.

[81] Nuray Yıldız, Rüya Gönülşen, Hülya Koyuncu, and Ayla Çalımlı. Adsorption of benzoic acid and hydroquinone by organically modified bentonites. *Colloids*

and Surfaces A: Physicochemical and Engineering Aspects, 260(1-3):87–94, 2005.

[82] Stephen Brunauer, P. H. Emmett, and Edward Teller. Adsorption of gases in multimolecular layers. *Journal of the American Chemical Society*, 60(2):309–319, 1938.

[83] Dorian A. H. Hanaor, Maliheh Ghadiri, Wojciech Chrzanowski, and Yixiang Gan. Scalable surface area characterization by electrokinetic analysis of complex anion adsorption. *Langmuir*, 30(50):15143–15152, 2014. PMID: 25495551.

[84] JR Lu, ZX Li, RK Thomas, BP Binks, D Crichton, PDI Fletcher, JR McNab, and J Penfold. The structure of monododecyl pentaethylene glycol monolayers with and without added dodecane at the air/solution interface: A neutron reflection study. *The Journal of Physical Chemistry B*, 102(30):5785–5793, 1998.

[85] Jnanojjal Chanda and Sanjoy Bandyopadhyay. Molecular dynamics study of a surfactant monolayer adsorbed at the air/water interface. *Journal of Chemical Theory and Computation*, 1(5):963–971, 2005.

[86] Liu Shi, Naga Rajesh Tummala, and Alberto Striolo. C12e6 and sds surfactants simulated at the vacuum-water interface. *Langmuir*, 26(8):5462–5474, 2010.

[87] Buffers, aug 15 2020. [Online; accessed 2022-03-07].

[88] https://pubchem.ncbi.nlm.nih.gov/compound/Phosphate. Accessed: 2022-3-11.

[89] Kipton J. Powell, Paul L. Brown, Robert H. Byrne, Tamás Gajda, Glenn Hefter, Staffan Sjöberg, and Hans Wanner. Chemical speciation of environmentally significant heavy metals with inorganic ligands. part 1: The hg2+– cl–, oh–, co32–, so42–, and po43– aqueous systems (iupac technical report). *Pure and Applied Chemistry*, 77(4):739–800, 2005.

[90] Hossain Azam, Seemi Alam, Mahmudul Hasan, Djigui Yameogo, Arvind Kannan, Arifur Rahman, and Man Jae Kwon. Phosphorous in the environment: characteristics with distribution and effects, removal mechanisms, treatment technologies, and factors affecting recovery as minerals in natural and engineered systems. *Environmental Science and Pollution Research*, 07 2019.

[91] Meenesh R Singh, Jason D Goodpaster, Adam Z Weber, Martin Head-Gordon, and Alexis T Bell. Mechanistic insights into electrochemical reduction of co2 over ag using density functional theory and transport models. *Proceedings of the National Academy of Sciences*, 114(42):E8812–E8821, 2017.

[92] Ole Pedersen, Timothy Colmer, and Kaj Sand-Jensen. Underwater photosynthesis of submerged plants – recent advances and methods. *Frontiers in plant science*, 4:140, 05 2013.

[93] B. Limburg, E. Bouwman, and S. Bonnet. Rate and stability of photocatalytic water oxidation using [ru(bpy)3]2+ as photosensitizer. *ACS Catalysis*, 6(8):5273–5284, 2016.

[94] Bart Limburg, Elisabeth Bouwman, and Sylvestre Bonnet. Molecular water oxidation catalysts based on transition metals and their decomposition pathways. *Coordination Chemistry Reviews*, 256:1451–1467, 08 2012.

[95] B. Limburg, E. Bouwman, and S. Bonnet. Rate and stability of photocatalytic water oxidation using [ru(bpy)3]2+ as photosensitizer, supporting information. *ACS Catalysis*, 6(8):5273–5284, 2016.

[96] H.P.S. Abdul Khalil, Chaturbhuj K. Saurabh, M.I. Syakir, M.R. Nurul Fazita, Aamir Bhat, A. Banerjee, H.M. Fizree, Samsul Rizal, and Paridah Md. Tahir. 13 - barrier properties of biocomposites/hybrid films. In Mohammad Jawaid, Mohamed Thariq, and Naheed Saba, editors, *Mechanical and Physical Testing of Biocomposites, Fibre-Reinforced Composites and Hybrid Composites*, Woodhead Publishing Series in Composites Science and Engineering, pages 241–258. Woodhead Publishing, 2019.

[97] William Betz and M. Keeler. Use of carbon molecular sieves for for chromatographic separations of permanent and other gases. 03 2022.

[98] Ewa Lorenc-Grabowska. Effect of micropore size distribution on phenol adsorption on steam activated carbons. *Adsorption*, 22(4):599–607, 2016.

[99] JOSEPH S D'Arrigo. Screening of membrane surface charges by divalent cations: an atomic representation. *American Journal of Physiology-Cell Physiology*, 235(3):C109–C117, 1978.

[100] JR Lu, ZX Li, RK Thomas, EJ Staples, I Tucker, and J Penfold. Neutron reflection from a layer of monododecyl hexaethylene glycol adsorbed at the air-liquid interface: the configuration of the ethylene glycol chain. *The Journal of Physical Chemistry*, 97(30):8012–8020, 1993.

[101] JR Lu, TJ Su, ZX Li, RK Thomas, EJ Staples, I Tucker, and J Penfold. Structure of monolayers of monododecyl dodecaethylene glycol at the air- water interface studied by neutron reflection. *The Journal of Physical Chemistry B*, 101(49):10332–10339, 1997.

[102] R. Massoudi and A. D. King. Effect of pressure on the surface tension of water. adsorption of low molecular weight gases on water at 25.deg. *The Journal of Physical Chemistry*, 78(22):2262–2266, 1974.

[103] William D Nesse. *Introduction to mineralogy*. Number 549 NES. 2012.