Distributed load-sharing algorithm for an electrochemical hydrogen compression system

van Baalen, Allard R.

Award date:
2020

Distributed load-sharing algorithm for an electrochemical hydrogen compression system

Master thesis report

A.R. van Baalen
(1020733)

Arnhem, June 2020

Supervisors
D. Goswami (TU/e)
A. Bos (HyET Hydrogen)

Abstract

There is a growing effort in using renewable energy as an energy source, but many of these sources are not always available. For example, sunlight can only be harvested during the day, and for wind energy there must be wind. There is a real need to store energy such that energy can be harvested when it is available and also used when it is not available. A possible solution is to use hydrogen as energy storage. Hydrogen can be generated using electricity by the process of electrolysis, after which the process can be inverted such that we get electricity again. The downside of using hydrogen as energy storage is its low gas density, this requires us to compress hydrogen such that we can store it efficiently.

A modern device for compressing hydrogen is an electrochemical compressor. This type of compressor can efficiently compress hydrogen up to very large pressures. A disadvantage of this type of compressor is its low flow rate. In order to use electrochemical compression for large-scale energy storage, we need many compression units working together.

In the presented thesis is a polynomial-time load-sharing algorithm for electrochemical compressors presented. The algorithm will meet a certain mass-flow rate demand with near-optimal power consumption while respecting real-world constraints.

Contents

List of Figures

List of Equations

List of Algorithms

Chapter 1

Introduction

The concept of compressing hydrogen using an electrochemical compressor has already been published in 1981, in which it was shown that a direct current through a hydrated polymer electrolyte cell can transport hydrogen from a low- to high-pressure side (Sedlak, Austin, & LaConti, 1981). It was not until 2012 before the first electrochemical hydrogen compressor could compress hydrogen up to thousand bars (HyET Hydrogen). In the meantime, the technique has matured and is now ready to be employed in a system. This gives rise to new challenges. For one, a single electrochemical hydrogen compressor has limited flow. Therefore, many compressors are needed to achieve large flow rates. Since the compression efficiencies vary over time and between compressors, it becomes increasingly difficult to meet a certain flow rate demand with optimum power consumption. In this thesis is a load-sharing algorithm presented that is capable of sharing the flow rate between the compression units in a system such that a certain overall flow rate demand is achieved with near-optimal power consumption.

1.1. Electrochemical hydrogen compressor

An electrochemical hydrogen compressor is a compressor that compresses hydrogen without mechanical pressure. Instead, we use electricity to move protons from one side of a proton exchange membrane (PEM), to the other (high pressure) side with a rate that is proportional to the electrical current. A PEM is sandwiched between two electrodes that are electrically isolated from each other by the PEM, we call this assembly a membrane electrode assembly (MEA). An electrochemical hydrogen compressor typically consists of multiple MEAs that are sandwiched between plates who are responsible of distributing the hydrogen over the MEA. The following figure shows a simplified electrochemical hydrogen compressor.

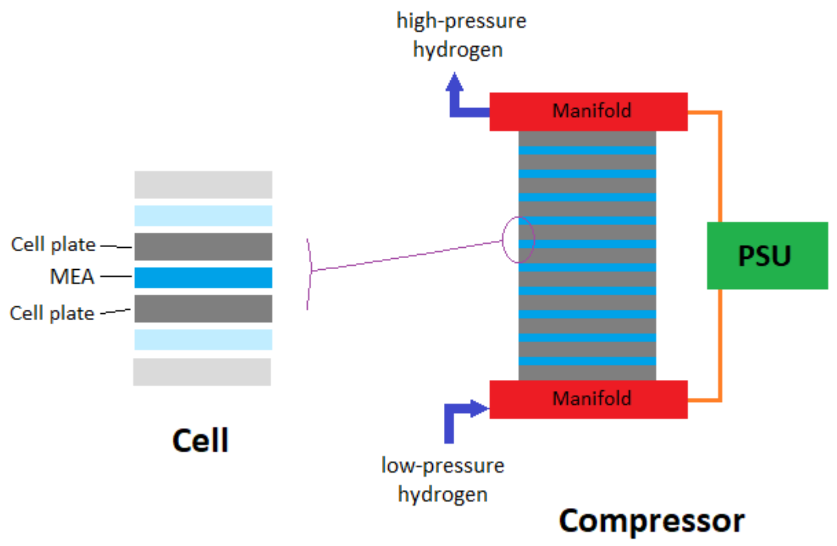

Figure 1: Electrochemical hydrogen compressor.
(Power Supply Unit (PSU), Membrane Electrode Assembly (MEA))

In figure 1 is an electrochemical compressor depicted with a low-pressure hydrogen input and a high-pressure hydrogen output. All cells are electrically connected in series and powered by a power supply unit (PSU) that powers the compressor with a direct current. For this thesis, we assume a parallel gas configuration which implies that each cell is exposed to the full pressure difference, and with each cell we increase the maximum flow rate.

1.1.1. Shunting a cell

With an electrochemical hydrogen compressor, we need to be careful that the voltage over a cell is not too high, both to overcome damaging the cell as to keep the power consumption low. It may very well be that there are one or more cells in a compressor with a voltage that is too high, requiring the compressor current to be tuned down in order to avoid damaging the cell. As an alternative, we can shunt a cell and divert a fraction of the current by placing a shunt or a variable shunt between the cell plates, patented in (United States Patent No. US9915004B2, 2014), and illustrated below.

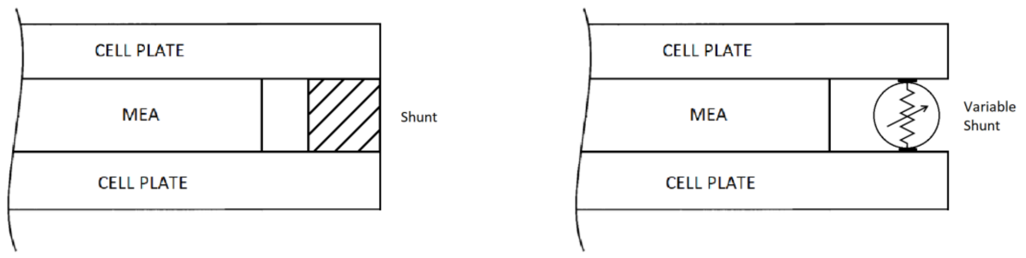

Figure 2: Shunt (left), variable shunt (right).
(Membrane Electrode Assembly (MEA))

Note that a variable shunt resistor can also imply an integrated circuit or transistor(s).

1.2. Compression system

A compression system is a system with one or more compression units and auxiliary devices required to operate the compression unit(s). These auxiliary devices include a cooling system, humidifier, hydraulics, pipe heaters, sensors and valves. For this thesis we do not take auxiliary devices into account and we focus on the compression units themselves. Each compression unit contains a power supply unit (PSU) with a power input and a processor that controls the PSU, reads the cell voltages, and communicates with external devices over a communication bus. Moreover, each compression unit has a low- and high-pressure hydrogen port for which it is assumed that all compressors are connected to the same low- and high-pressure media as depicted in the following figure.

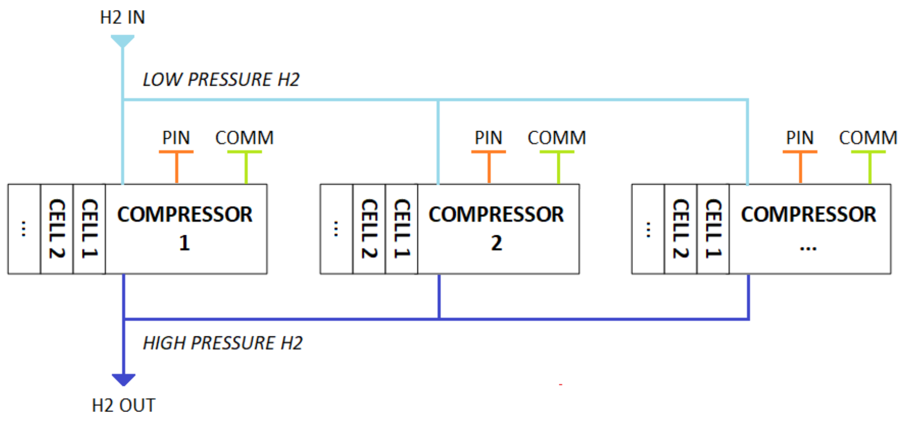

Figure 3: Hydrogen (H2) gas connections in an electrochemical hydrogen compression system. (Power input (PIN), Communication port (COMM))

During the thesis preparation phase, we have found that we should implement sections, we call such a section a skid. Every skid includes a processor that represents the skid node and one or more compression nodes who represent the compression units. The skid node acts as a gateway between the nodes in a skid and external nodes to limit the number of messages and increase the potential system size. As for the communication bus, we found that the CAN bus is a cost-effective and reliable option. However, this algorithm is communication bus independent and will work with any communication bus given that the speed is sufficient, and the following topology is implemented.

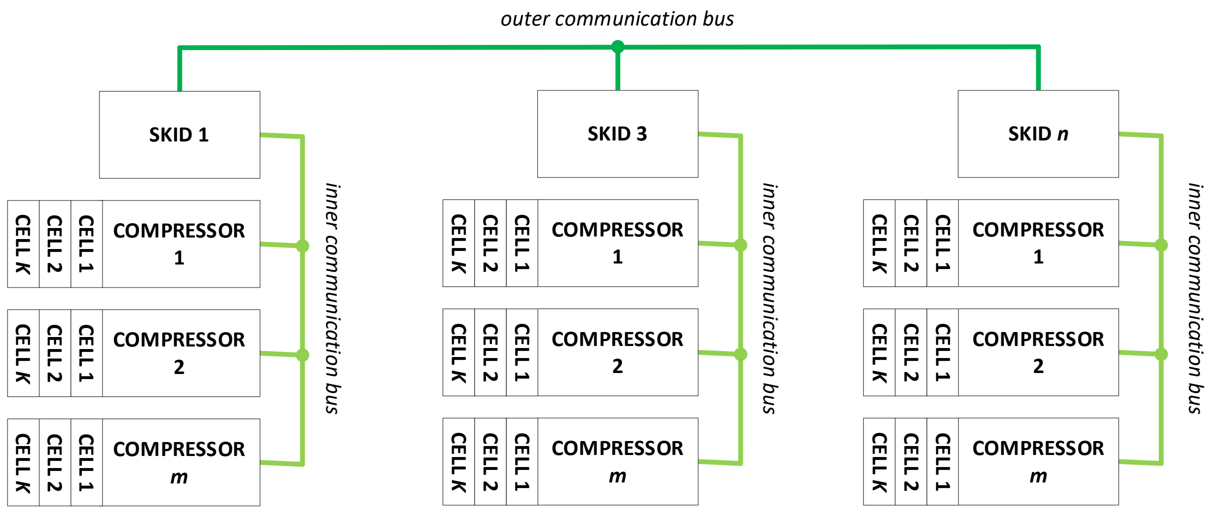

Figure 4: Compression system consisting of n skids, $n * m$ compressors, and $n * m * k$ cells

We propose two communication busses in figure 4, an outer bus that connects the skid nodes to each other, and an inner bus that connects the compressors within a skid to the skid node. We conceptionally describe the communication in the following three steps.

1. Each compressor communicates characteristics over the inner bus to the skid node.
2. The skid nodes process the characteristics and broadcast a subset over the outer network.
3. The skid nodes process the subsets and communicate a solution to the compressors within the skid.

1.3. Structure of the report

In this report is a load-sharing algorithm presented for the electrochemical hydrogen compression system as proposed in section 1.2. In the current chapter is an introduction given, followed by the problem statement in the next chapter. In the third chapter is a study done in related work, which we will use to model the problem in the fourth chapter and develop an algorithm in the fifth chapter. We will present the algorithm and define how this algorithm is to be implemented in the proposed compression system. We conclude the thesis with results and a conclusion in the sixth and seventh chapters.

Chapter 2

Problem statement

For the compression system as proposed in section 1.2, with n skids, where each skid has at most m compression units, each compression unit at most k cells and each cell an electronic shunt that can be enabled or disabled. We must decide which cells to shunt and how much current to feed through the compressors such that we meet hydrogen flow demand d with minimum power consumption. Moreover, we must take real-world limitations into account which implies that we must stay below a certain cell voltage limit such that we do not damage the electrochemical cells, and we must stay below the maximum current rating of the power supply unit. When it is not possible to meet flow rate demand d, we must return the system configuration that gives the highest flow rate. Moreover, we need to solve the problem periodically for big systems, which implies that we need to solve the problem efficiently. More precisely, we need to solve the problem in polynomial time for system size $n * m * k$.

We divide the problem in three major parts, namely.

P1. Decide for each compressor which cells to shunt and output all the relevant shunt configurations per compressor. We define a shunt configuration as a set of Booleans, one Boolean for every cell that indicates if the corresponding cell is shunted.

P2. Decide for every compressor which shunt configuration to use and how much current to feed through the compressor such that we meet flow rate demand d with minimum power consumption.

P3. Adapt the solution such that it can be implemented in the system as proposed in figure 4. This implies that we need to decide what each node must do and what data there must be communicated over the busses.

2.1. Milestones

We will first conduct a research in the behavior of an electrochemical hydrogen cell and use the results to develop a model that represents the compressor. We will also do a research in related algorithms and use this to develop an algorithm that solves the problem in polynomial time. The last part is to adapt the algorithm such that it can be implemented in the proposed system and nodes. We summarize the following milestones.

1. Conduct a research in the behavior of an electrochemical hydrogen cell, section 3.1.
2. Conduct a research in related algorithms, section 3.2.
3. Create a model that represents the electrochemical compressor, chapter 4.
4. Develop an algorithm that solves problems P1 and P2 in polynomial time, P1 in section 5.1, and P2 in section 5.2.
5. Solve problem P3 by adapting the algorithm such that it can be implemented in the proposed compression system (figure 4), section 5.3.

Chapter 3

Related work

In this chapter is a study in related work presented. We will first present results that help us model the compressor in section 3.1. And secondly, we study algorithms that solve similar problems in section 3.2.

3.1. Electrochemical cell

An electrochemical cell has already been modelled in 1971 by Macdonald (Macdonald, 1971), for which the following equivalent circuit has been suggested.

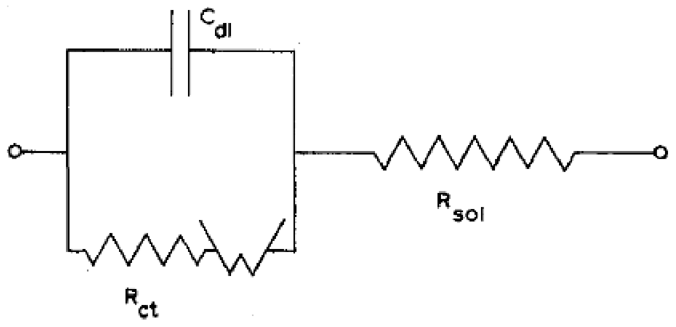

Figure 5: Equivalent circuit electrochemical cell (Macdonald, 1971), with Rsol as the solution resistance, Rct as the charge transfer resistance, Cdl as the double layer capacity and W the Warburg impedance

The equivalent circuit from figure 5 is better known as a Randles circuit (Randles, 1947) commonly used in Electrochemical impedance spectroscopy, a method to characterize electrochemical systems. In this circuit is the faradaic reaction, i.e. the reaction that causes hydrogen protons to move from one side of the PEM to the other side, represented by the Rct resistance and $Warburg$ impedance.

3.1.1. Behavior of an electrochemical cell

It has been shown by (Grigoriev, Shtatniy, Millet, Porembsky, & Fateev), (Suermann, Kiupel, Schmidt, & Buchi), (Scheepers, et al.), (Ströbel, et al.), and many more that the behavior of an electrochemical cell depends on many parameters. Namely, membrane material, temperature, humidity, gas purity, pressure difference and current density. For this thesis, we are interested in what happens when we change the current through a cell or shunt a cell, in both cases, we are effectively changing the current density. The following figures show us what happens if we change the current density of an electrochemical cell for various temperatures, gas concentrations and membrane materials.

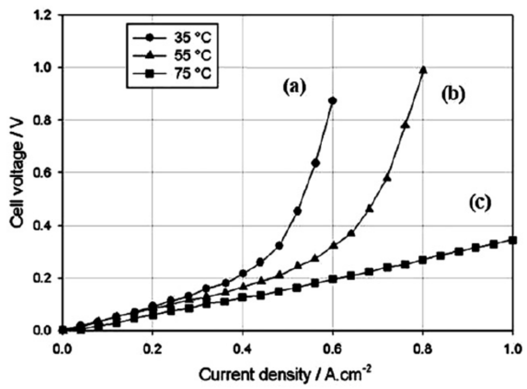

Figure 6: I-V curves for different temperatures (Grigoriev, Shtatniy, Millet, Porembsky, & Fateev, 2011)

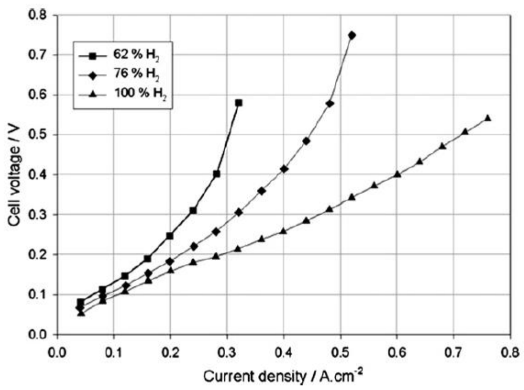

Figure 7: I-V curves for different hydrogen concentrations (Grigoriev, Shtatniy, Millet, Porembsky, & Fateev, 2011)

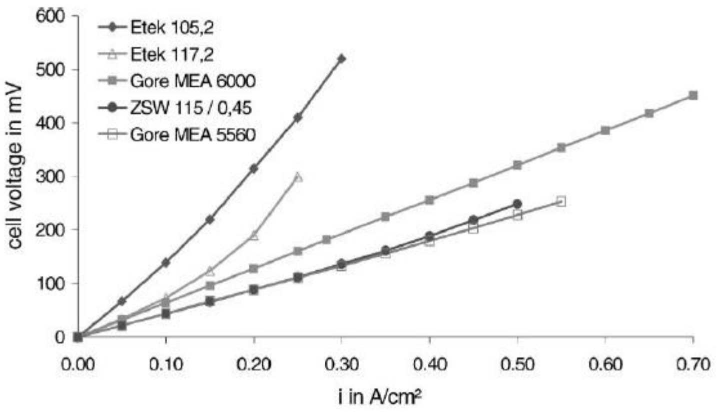

Figure 8: I-V curves for different membranes (Ströbel, et al., 2002)

We see that the electrochemical cells are first in a linear region where the cell behaves as a resistor, after which the cell voltage increases exponentially. Working outside the linear region is unadvised since a small change in current density can cause a large change in cell voltage, possibly damaging the cell.

Hydrogen flow rate

Faraday has researched electrochemistry already in 1834, from which Faraday's laws of electrolysis originated (Faraday, 1834). Faraday showed that the amount of material produced during an electrochemical reaction is directly proportional to the average current multiplied by the experiment time. We can use these laws to calculate the amount of hydrogen that is transported through the membrane as.

$$\frac{dn}{dt} = \frac{I}{2F}$$

Equation 1: Hydrogen molecule flow rate of an electrochemical cell (Rohland, Eberle, Ströbel, Scholta, & Garche, 1998)

Where:
dn/dt is the hydrogen flow rate $[mol/s]$.
I is the current though the cell $[A]$.
F is the Faraday constant $[A * s/mol]$.

NERNST voltage

Figure 6, figure 7 and figure 8 do not show the behavior when there is a pressure difference over an electrochemical cell. To this end, let us use the following expression derived from the NERNST equation to calculate the effect of pressurization on the cell voltage.

$$E = \frac{R * T}{2F} \ln \frac{P_{H2}^{HP}}{P_{H2}^{LP}}$$

Equation 2: Effect of pressurization on cell voltage according to the NERNST equation (Rohland, Eberle, Ströbel, Scholta, & Garche, 1998)

Where:
E is the cell potential $[V]$.
R is the universal gas constant $[J/K * mol]$.
T is the temperature $[\mathrm{K}]$.
F is the Faraday constant $[A * s/mol]$.
P_{H2}^{HP}/P_{H2}^{LP} is the hydrogen compression factor.

The voltage calculated in equation 2 gives the theoretical cell voltage as a function of the pressure difference. In order to pressurize hydrogen, we need to overcome the theoretical cell voltage. For a hundredfold compression factor and a temperature of 300 Kelvin, a cell voltage of 60 mV is required according to equation 2.

Back-diffusion

Another effect of pressurization is back-diffusion. When there is a pressure difference over the electrochemical cell, we have hydrogen molecules migrating from the high- to the low-pressure side. This migration can be compensated by an electrical current such that there is found an equilibrium at which there is no hydrogen flow. We use this equilibrium current together with Faradays laws to calculate the back-diffusion rate as.

$$\frac{dn}{dt} = \frac{I * V_0 * T}{2F * T_0}$$

Equation 3: Hydrogen molecule back-diffusion rate (Ströbel, et al., 2002)

Where:

dn/dt is the hydrogen back-diffusion flow rate $[mol/s]$.
I is the equilibrium current $[A]$.
V_0 is the standard molar volume $[1/mol]$.
T is the temperature $[K]$.
F is the Faraday constant $[A * s/mol]$.
T_0 is the standard temperature $[K]$.

3.2. Related algorithms

In this section are related algorithms studied to solve the second part of the problem as described in the problem statement and repeated below.

> *Decide for every compressor which shunt configuration to use and how much current to feed through the compressor such that we meet mass-flow rate demand d with minimum power consumption.*

If we ignore the part about determining how much current to feed through a compressor and the effect this has on the compression flow and power consumption, we realize that we essentially have a combinatorial optimization problem. Similar to the well-known 0-1 Knapsack problem, defined as.

> *Let there be a set of items, where each item has a value and a weight. Determine which items to put in the knapsack such that the weight is at most W and the value is maximum.*

Instead of selecting items, we select configurations. With power being the value and flow being the weight. We do now have a minimization problem as the power must be minimized, and instead of having a weight at most equal to W, we must have a flow at least equal to d.

The Knapsack problem has been researched extensively, with early works dating back to 1897 (Mathews, 1897). It has been shown that the 0-1 Knapsack problem with real values and weights is NP-complete, thus there is no known algorithm that can solve the problem both optimal and in polynomial time for any given input. However, it has been found that if the weights or values are integers, we can solve it optimally in polynomial time using Dynamic-programming, explained in section 3.2.1. To solve a problem with real values, we can use a scaling and rounding step that converts the real values to integers. Unfortunately, this introduces rounding errors, causing the solution to be potentially non-optimal. We can create a *polynomial-time approximation scheme* as explained in (de Berg, 2019), in which we regulate the rounding error using a parameter $\varepsilon > 0$. With ε, we can set the tradeoff between accuracy and speed, a higher ε gives us a faster execution at the cost of reduced accuracy and visa-versa. There is another approach in solving the Knapsack problem that does not require the weights or values to be integers, namely, a greedy algorithm for the Fractional-Knapsack problem. As the name suggests, we require that it is possible to take fractions of items. We discuss the Fractional-Knapsack problem in section 3.2.2.

3.2.1. Dynamic programming

Dynamic programming has been developed in 1950 by Richard Bellman (Bellman, 1957). The general idea behind dynamic programming is to divide a problem in smaller sub-problems where the results of these subproblems are stored such that they do not have to be recomputed at a later point. This simple concept has been implemented in dynamic programming algorithms in many different fields.

If we take the 0-1 integer Knapsack problem, for n items, we can either pack an item or not pack an item, we find that there are 2^n possible combinations of items to pack in the Knapsack which leads us to believe that there are $O(2^n)$ computations to do. It turns out that many of the computations are done multiple times. With dynamic programming we will do each computation just once. This is best explained by imagining a table with n rows and W columns. Where $cell_{i,j}$ contains the maximum value for the first i items and weight j. We can fill the table in $O(n * W)$ time, after which we can find our maximum value for weight W in $cell_{n,W}$.

3.2.2. Fractional knapsack problem

It has been found by George Dantzig in 1957 (Dantzig, 1957) that if we adapt the original 0-1 Knapsack problem such that we can take fractions of items, we can solve it optimally using a Greedy approach in $O(n \log_2 n)$ time, for input size n. The basic idea is to calculate the $value/weight$ ratio for each item and sort all the items on this ratio, highest value first. We will now take items one by one until we cannot add the next item as a whole, instead we add as much of the next item as possible such that the total weight is exactly W. This approach will always give an optimal solution and does not require the weights or values to be integers. However, it does require the weights and values to scale the same. In other words, if we have an item x and we pack $1/2\ x$, we must have a resulting value of $1/2 * value(x)$ and a resulting weight of $1/2 * weight(x)$.

3.2.2.1. Non-linear fractional knapsack problem

It turns out that we do not have a flow and power that scale equally, we have an exponential relationship in which the consumed power becomes exponentially large as the hydrogen flow increases. Such a problem is similar to the nonlinear knapsack problem, generally defined as.

$$\begin{array}{ll} Min: & f(x) \\ Such\ that: & g(x) \leq b \\ & x \in S \end{array}$$

(Bretthauer & Shetty, p. 460) addresses the following five variations of this basic problem definition, cited as.

1. *“Convex, separable, continuous*: $f(x)$ and $g(x)$ are convex separable functions, *S* includes bounds on the continuous variables.
2. *Convex, separable, integer*: Same as problem type 1 except *S* includes integrality conditions on the variables.
3. *Nonconvex, separable (continuous and integer)*: $f(x)$ and $g(x)$ are nonconvex separable functions, *S* includes bounds on the variables.

4. *Convex, separable, additional block diagonal (or GUB) constraints (continuous and integer)*: Same as problem type 1 or 2 except *S* also includes block diagonal or GUB constraints.
5. *Convex, non-separable (continuous and integer)*: $f(x)$ and $g(x)$ are convex non-separable functions, *S* includes bounds on the variables."

Our problem is closely related to the first variation, for which there are two basic approaches, *multiple search methods*, and *variable pegging methods*.

With multiple search methods, we use a set of equations to solve the problem. An example is given in (Bretthauer & Shetty, The nonlinear resource allocation problem, 1995) that solves the multiple search algorithm via a one-dimensional search by using the derivative of the functions and the *Lagrange multiplier*.

Variable pegging methods initially neglect all bounds and calculate an initial output. In the next iterations are items bounded such that we solve the problem and satisfy the limits. The generalized problem with lower and upper bounds has been solved in (Bretthauer & Shetty, A pegging algorithm for the nonlinear resource allocation problem, 2002), in which always at least one item is pegged per iteration, guaranteeing a finite amount of iterations.

Chapter 4

Model of the electrochemical compressor

To the end of modelling the electrochemical compression unit, let us first expand the compressor from figure 1 with the electronic shunts such that we get the compressor as depicted below.

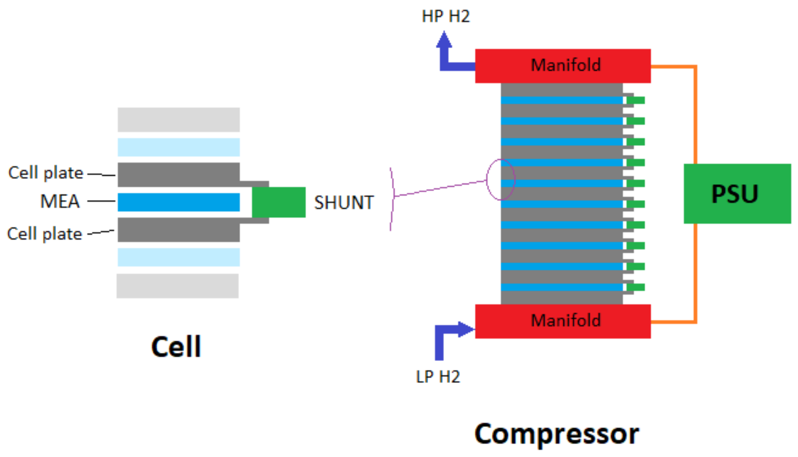

Figure 9: Electrochemical hydrogen compressor with electronic shunts. (Power Supply Unit (PSU), Membrane Electrode Assembly (MEA), Low Pressure (LP), High Pressure (HP), Hydrogen (H2))

We found in section 3.1 that an electrochemical cell behaves linearly at first if we consider the current density versus the cell voltage, after which the cell voltage rises exponentially. For this thesis, we assume to be in the linear region when staying below the cell voltage limit, one of the parameters defined in the problem statement. Moreover, since we are driving the compressor using a direct current, we can greatly simplify the equivalent circuit of the electrochemical cell shown in figure 5 and model the cell as a single resistor if we ignore the effect of pressurization. We include the effect of pressurization by realizing that the cell behaves as a voltage source for which the voltage is directly related to the pressure difference. We assume that the wire resistance from the PSU to the first/last cell is neglectable and draw the electrochemical compressor as the circuit depicted below.

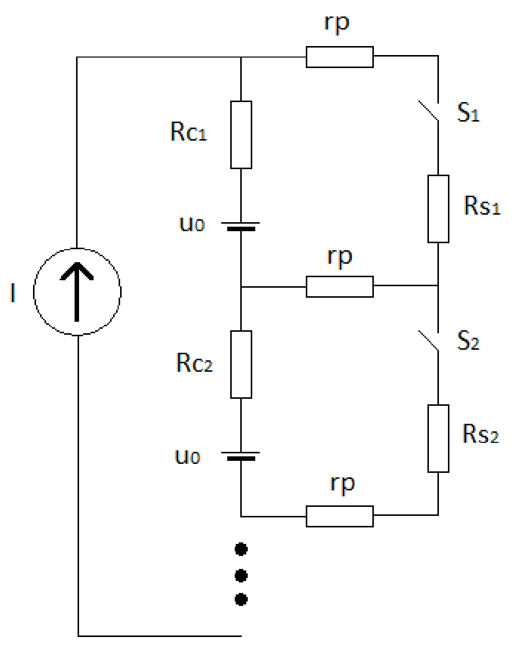

Figure 10: Electrochemical compressor modelled as an electrical circuit, Current source (I), Cell plate resistance (rp), Cell resistance (Rc$_x$), Shunt resistance (Rs$_x$), Shunt switch (S$_x$), pressurization boltage (u0)

The model in figure 10 represents an electrochemical compressor for which two cells are drawn. We represent the electrochemical cell resistance by Rc_i, the plate resistance by rp, the shunt resistance by Rs_i, and the actual shunt by S_i. We model the pressurization effect as a voltage source for which the voltage u_0 is calculated with the NERNST equation (equation 2). Lastly, the compressor is driven by a direct current source with current I.

When a cell is under pressure and shunted, we might create the current path as illustrated in the next figure. This current path is undesirable as it allows the process to reverse, i.e. for hydrogen to flow from the high-pressure to the low-pressure side.

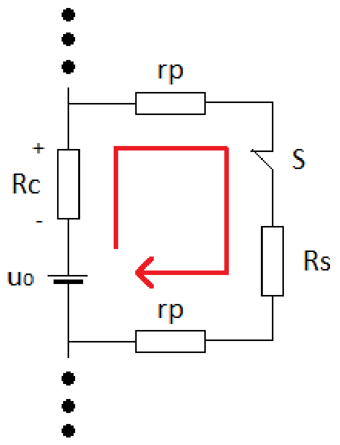

Figure 11: Current path due to the pressurization effect,
Cell plate resistance (rp), Cell resistance (Rc), Shunt resistance (Rs), Shunt switch (S), pressurization voltage (u0)

To avoid the situation as depicted in figure 11, we must ensure that the voltage over Rc is positive by either feeding a sufficiently high current through the compressor or by un-shunting the respective cell.

4.1. Solving the model

In order to calculate the effect of increasing/decreasing the compressor current for a certain shunt configuration, we first solve the model from figure 10 for the shunt configuration into an intermediate form that allows us to efficiently compute the cell voltages, flow rate and power consumption as a function to the compressor current. The intermediate form consists of a number of in series connected resistors and voltage sources, where each resistor/source pair represents a cell and the respective shunt. The intermediate form for two cells is depicted in the following figure.

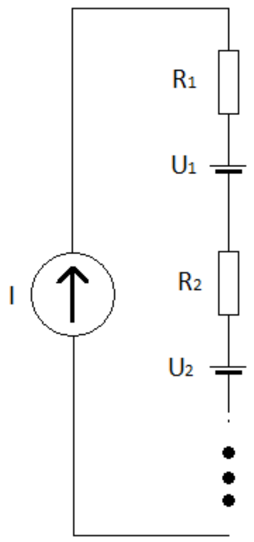

Figure 12: Intermediate form of a solved electrochemical compressor model. (Compressor current (I), Equivalent cell resistance (R_x), equivalent pressurization voltage (U_x))

From the intermediate form, we calculate the voltage over cell i for the compressor current I as.

$$E_i(I) = I * R_i + U_i$$

Equation 4: Voltage over cell i

And the current through cell i with cell resistance Rc_i and pressurization voltage $u0$ as.

$$A_i(I) = \frac{E_i(I) - u0}{Rc_i}$$

Equation 5: Current through cell i

We use equation 1 to calculate the flowrate through an electrochemical cell. However, this equation assumes that there is no pressure difference over the cell. To include the effect of pressurization we introduce a compensation current $icomp$ that compensates for the back-diffusion rate (equation 3). We calculate the mass-flow rate λ for a compressor with k cells and compressor current I as.

$$\lambda(I) = \frac{H}{2F}\sum_{i=1}^{k} A_i(I) - icomp$$

Equation 6: Hydrogen mass-flow rate for a compressor with k cells

Where:
H is the molar mass of *H2* $[g/mol]$.
F is the Faraday constant $[A * s/mol]$.

The power consumption is the electric power that is dissipated in the compressor, which we calculate as.

$$P(I) = I\sum_{i=1}^{k} E_i(I)$$

Equation 7: Power consumption of a compressor with k cells

Solving the model from figure 10 with all the shunts disabled is trivial as this would result in solving a network for which we have that $R_x = Rc_x$ and $U_x = u0$. However, if multiple cells are shunted consecutively, it becomes more difficult. Solving such a shunted section is done using a combination of Ohm's law, Kirchhoff's laws, superposition theorem and wye-delta transformations. To explain the cell voltage calculation when multiple cells are shunted consecutively, we will solve a shunted section consisting of three cells. We use the superposition theorem which implies that we solve the circuit for every power source. We first solve the circuit for the current source in section 4.1.1 after which we solve for the voltage sources in section 4.1.2. The algorithm for solving the electrochemical compressor is provided in Appendix A.2.

4.1.1. Solving for the current source

We solve for the current source by disabling all the voltage sources and apply two iterative steps. Firstly, we simplify the circuit using delta-wye transformations and calculate the equivalent section resistance. Secondly, we use the equivalent section resistance to determine the section voltage as a function of the current and walk back the previous iterations to calculate the cell voltages.

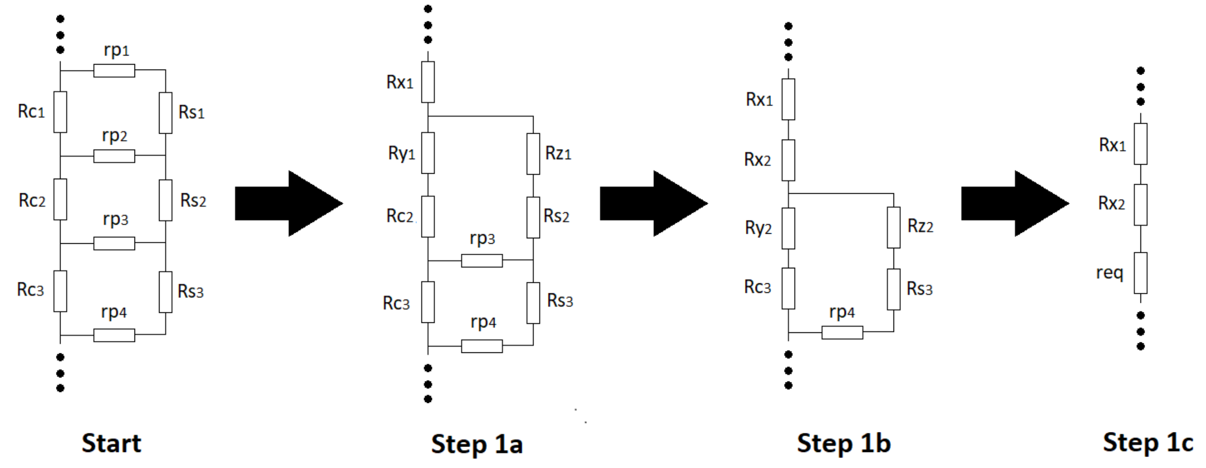

Figure 13: Calculating the equivalent section resistance for the current source

The first iterative step is depicted in figure 13. We start with the drawing all the way to the left and find that if we combine rp_1 and Rs_1, we create a triangle together with Rc_1 and rp_2. We can now do a delta-wye transformation and calculate Rx_1, Ry_1 and Rz_1. In step 1a, we find another triangle with rp_3 if we combine Ry_1 and Rc_2, and also combine Rz_1 and Rs_2. We do another delta-wye transformation and end up with the circuit in 1b. In this step we calculate the equivalent resistance that is formed by the parallel circuit consisting of the sum of Ry_2 and Rc_3, and the sum of Rz_2, Rp_3 and Rp_4. We end up the circuit shown in step 1c and calculate the equivalent section resistance as the sum of Rx_1, Rx_2 and req.

In the second step, we assume a compressor current of one ampere and calculate the cell voltages. We note that the cell voltages are proportional to the compressor current and that we are effectively calculating the cell voltage per ampere of compressor current. In other words, we calculate the equivalent cell resistances R_x. We will first calculate the voltage over the entire section $Vsection$. After which we walk back the iterations from the first step and calculate the node voltages $U1$, $U2$, $U3$ and $U4$, illustrated in the figure below.

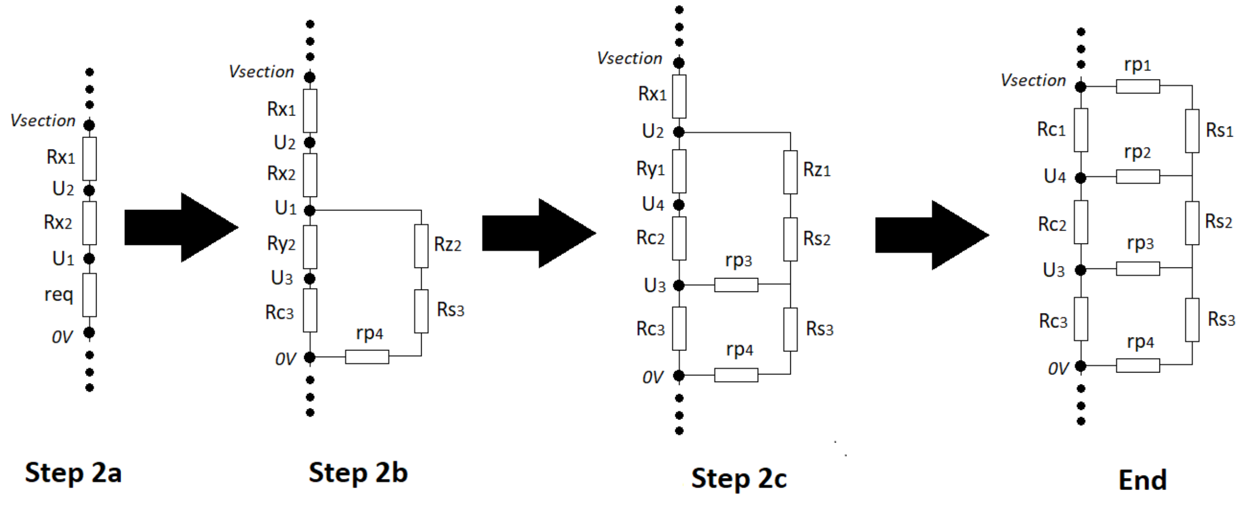

Figure 14: Calculating the cell voltages

We can calculate the cell voltages in the *End* drawing from figure 14 as the voltage difference between the two surrounding node voltages. Remember that we actually calculate the equivalent cell resistances R_x.

4.1.2. Solving for the voltage sources

We solve for the voltage sources by disconnecting the current source and iteratively enable a single voltage source for which the circuit is solved. We solve the circuit in two iterative steps. Firstly, we simplify the circuit such that we end up with all the resistors connected in series. We can now calculate the equivalent resistance for the voltage source and use this to calculate the sourced current. Secondly, we revert the previous iterations and calculate the cell currents and voltages.

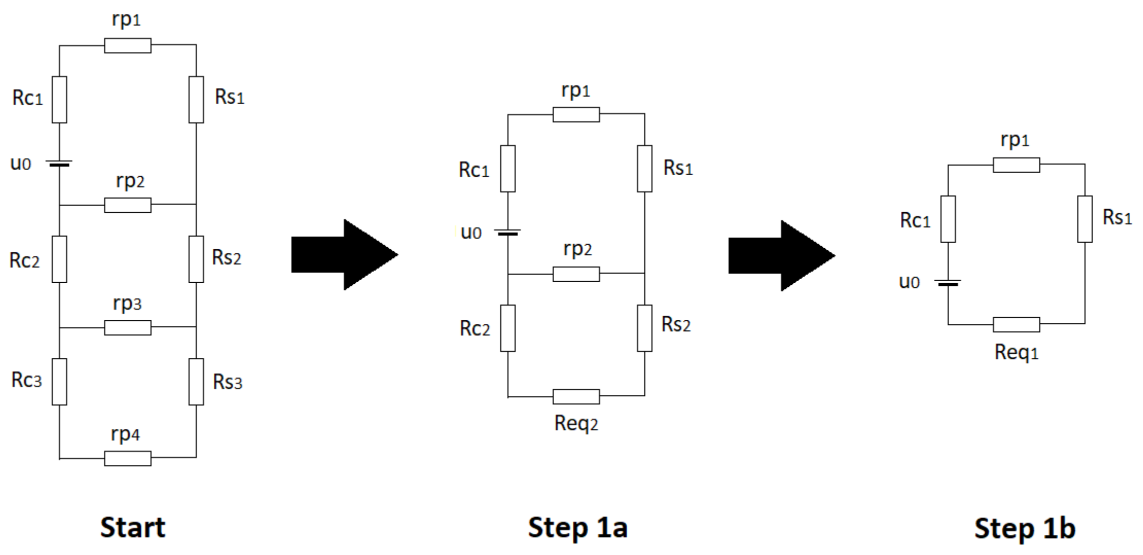

Figure 15: Calculating the equivalent section resistance for the voltage source

The first iterative step for a single voltage source is depicted in figure 15. We start with the drawing all the way to the left and find that if we sum Rc_3, Rs_3 and rp_4, we have a single resistor rp_3 in parallel. We calculate the equivalent resistance Req_2 and end up with the circuit in step 1a. We repeat the previous step and calculate the equivalent resistance Req_1 in step 1b by solving the parallel circuit with rp_2 and the sum of Rc_2, Rs_2 and Rb_2. We can now calculate the equivalent section resistance as the sum of Rc_1, rp_1 and Req_1.

With the second step, we walk back the iterations from the previous step and calculate the cell currents Ic_4, Ic_5, and Ic_6, illustrated in the figure below.

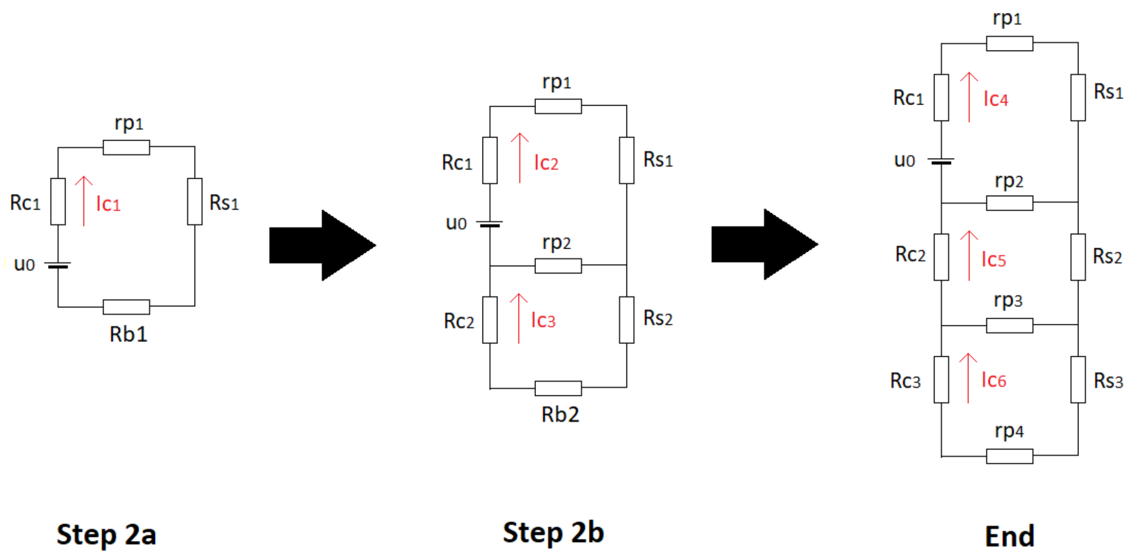

Figure 16: Calculating the cell currents

We can calculate the cell voltages in the *End* drawing in figure 16 by multiplying the cell current with the respective cell resistance. We repeat the previous steps for every voltage source and sum the calculated cell voltages such that we get the set with offset voltages U. Every U_i now represents the total offset voltage for cell i. In order to include u_0 and take care of the sign, we update every offset voltage U_i as follows.

$$U_i = u_0 - U_i$$

4.2. Modelling real-world constraints

To include the back-diffusion compensation current $icomp$, the maximum cell voltage $umax$, and the maximum compressor current $imax$ into our model, we convert them to minimum and maximum compressor currents, $imin$ and $imax$ respectively. We define $imin$ and $imax$ such that if we have a compressor current I and $imin \leq I \leq imax$, we will always have a cell current of at least $icomp$ a compressor current of at most $imax$, and a cell voltage of at most $umax$.

$$imin = \operatorname*{Max}_{1 \le i \le k}\left(\frac{icomp * Rc_i - U_i + u0}{R_i}\right)$$

Equation 8: Minimum compressor current

$$imax = \operatorname*{Min}_{1 \le i \le k}\left(imax, \frac{umax - U_i}{R_i}\right)$$

Equation 9: Maximum compressor current

Chapter 5

Load-sharing algorithm

We will now present the load-sharing algorithm that solves the problem as described in the problem statement, using the model we defined in chapter 4. We first solve problem P1 and present an algorithm that returns the relevant shunt configurations for a compressor. Secondly, for a system of N compressors, we solve problem P2 and present an algorithm that finds which shunt configurations to use and how much current to feed through each compressor. In the third and last section of this chapter, we include the skids and solve problem P3, we adapt the presented algorithm such that it can be implemented in the proposed hydrogen compression system defined in figure 4.

5.1. Problem P1 – find the relevant shunt configurations

In this section is described how we can find the relevant shunt configurations. A shunt configuration is a set of k Booleans, one Boolean for every cell in the compressor that is set when the corresponding shunt is enabled. We define the relevant shunt configurations as the configurations that are not outperformed by others considering flow rate and power consumption, such a set is also known as the Pareto-optimal set. The following image illustrates the Pareto-optimal set as the orange dots versus the poorer shunt configurations drawn as blue dots.

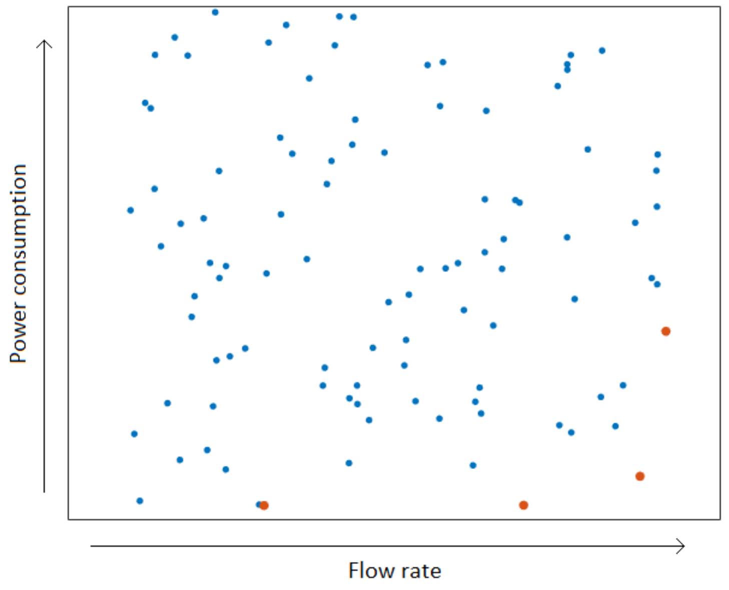

Figure 17: Best shunt configurations (orange dots), also known as the Pareto-optimal set, considering power consumption and flow rate.

A simple solution is to solve every combination of shunts and extract the Pareto-optimal set. However, this would lead to $O(2^k)$ combinations per compressor that need to be solved and examined. We can do this much smarter by realizing that if we shunt a cell, we divert part of the current through that cell. This causes a reduction in power and a reduction in flow rate. To counter the flow rate reduction, we can increase the compressor current and let the other cells work harder. Formulated like this, it becomes obvious that it is most effective to shunt the cell that is least efficient, i.e. dissipates the most power. Naturally, this is the cell with the highest resistance.

In order to select only the shunt configurations that belong to the Pareto-optimal set, we need to compare shunt configurations. One of the parameters of comparison is the maximum flow rate. The other parameter is the power consumption. Because the power consumption does not scale linearly with the flow rate, we must normalize the power consumption for a fair comparison. We define the normalized power consumption as the power consumption when we have a mass-flow rate of $1\ g/s$.

$$\hat{I} = icomp + \frac{2F/H + \sum_{i=1}^{k} (u0 - U_i)/Rc_i}{\sum_{i=1}^{k} R_i/Rc_i}$$ *Equation 10: Normalized current*

$$\hat{P} = \hat{I} \sum_{i=1}^{k} E_i(\hat{I})$$ *Equation 11: Normalized power*

Where:
$\hat{I}$ is the normalized current, i.e. current for $1\ g/s$ $[A]$.
$\hat{P}$ is the normalized power, i.e. power for1 g/s $[W]$.
k is the number of cells in the compressor.
$icomp$ is the back-diffusion compensation current $[A]$.
$u0$ is the pressurization voltage $[V]$.
U is the modelled offset voltage $[V]$.
R is the modelled equivalent resistance $[\Omega]$.
Rc is the cell resistance $[\Omega]$.
H is the molar mass of *H2* $[g/mol]$.
F is the Faraday constant $[A * s/mol]$.
$E_i(I)$ is cell voltage i for current I $[\mathrm{V}]$.

We define the algorithm that finds the Pareto-optimal shunt configurations for a compressor with k cells in polynomial time next.

Algorithm 1: Find the Pareto-optimal shunt configurations

```
$FindOptimalShuntConfigs(Rc, Rs, rp, u0, , umax, icomp, imax, k)$
1:    ▻ Input: A set $Rc = \{Rc_1, \dots, Rc_k\}$, with cell resistances, a set
        $Rs = \{Rs_1, \dots, Rs_k\}$ with shunt resistances, a cell plate resistance $rp$, a
        offset voltage $u0$, a maximum cell voltage $umax$, a compensation current
        $icomp$, a maximum compressor current $imax$, and a number of cells $k$.
2:    ▻ Output: A set $X2$ with up to $k + 1$ compressor models, where every
        model contains as set $Rc = \{Rc_1, \dots, Rc_k\}$ with cell resistances, a set
        $S = \{S_1, \dots, S_k\}$ with bits that represent the shunts, a set
        $R = \{R_1, \dots, R_k\}$ with equivalent cell resistances as specified in section
        4.1.1, a set $U = \{U_1, \dots, U_k\}$ with offset voltages as specified in section
        4.1.2, a minimum compressor current $imin$, and a maximum
        compressor current $imax$.
3:    $X1 \leftarrow \emptyset, X2 \leftarrow \emptyset$, Initialize $S_i \leftarrow 0$ for all $1 \le i \le k$.
4:
5:    ▻ Get the sorted indices for $Rc$ and solve models
6:    $J \leftarrow SortDescend(\{1, \dots, k\}, Rc)$
7:    $X1 \leftarrow SolveModel(S, Rc, Rs, rp, u0, umax, icomp, imax, k)$
8:    **for** $i = 1$ **to** $k$ **do**
9:      $j \leftarrow J_i,\ S_j \leftarrow 1$
10:     $X1 \leftarrow X1 \cup\ SolveModel(S, Rc, Rs, rp, u0, umax, icomp, imax, k)$
11:   **end for**
12:
13:   ▻ Only keep models that belong to the Pareto-optimal set
14:   **for** $i = 1$ **to** $k + 1$ **do**
15:     $f \leftarrow flow_{max}(X1_i), p \leftarrow power_{norm}(X1_i), add \leftarrow True$
16:     **for** $j = 1$ **to** $k + 1$ **do**
17:       **if** $i = j$ **then continue**
17:       **if** $flow_{max}(X1_j) \ge f$ **and** $power_{norm}(X1_j) \le p$ **then**
18:         $add \leftarrow False$, **break**
19:       **end if**
20:     **end for**
21:     **if** $add$ **then** $X2 \leftarrow X2 \cup X1_i$
22:   **end for**
23:   **return** $X2$
```

Algorithm 1 starts with sorting the cell resistances in descending cell resistance order. Next, we consider the situation where no cells are shunted and solve this configuration in $SolveModel$ as described in section 4.1 and implemented in Appendix A.2. In the following iterations are cells shunted one by one until all cells are shunted. With every newly shunted cell we solve and store the resulting model. In the second loop are only the models that belong to the Pareto-optimal set added to the output $X2$.

Let us consider the time complexity of the $FindOptimalShuntConfigs$ algorithm. If we apply the Heapsort algorithm from (Williams, 1964), we can do the sorting step in $O(k \log_2 k)$ time. Solving a model takes $O(k^2)$ time, since we solve k models in the first for-loop, we find a time complexity of $O(k^3)$ for the first loop. If we consider the second for-loop, we find that we can execute this loop in $O(k^2)$ time. We conclude that the computational complexity is determined by the first for-loop which gives us a resulting time complexity of $O(k^3)$.

5.2. Problem P2 – select shunt configurations and decide how much current to feed through each compressor

In the previous section are the shunt configurations modelled and the Pareto-optimal configurations returned, in this section we decide which shunt configuration to use and how much current to feed through each compressor such that we meet the hydrogen flow demand. We will use a dynamic programming approach as discussed in section 3.2.1. To this end, we develop an algorithm for when all power consumptions are integers. Let us assume that we have a system with a total of N compressors. For every compressor we define a set X that contains modelled shunt configurations. For every model $x \in X$, we define $power_{low}(x)$, $power_{high}(x)$, and $flow(x, p)$ as the minimum, maximum power consumption and flow rate for model x and power p, respectively. For a subset $Y \subseteq \{x_1 \in X_1 \dots x_N \in X_N\}$, and a set Z with the respective power consumption for every element in Y. We define $flow(Y, Z) = \sum_{i=1}^{|Y|} flow(Y_i, Z_i)$ and $power(Z) = \sum_{i=1}^{|Z|} Z_i$. Moreover, we define $power_{tot}$ as the maximum total power consumption for the entire system. For every $1 \leq i \leq N$ and $1 \leq j \leq power_{tot}$ we define.

$$A[i,j] = Max(\,flow(Y,Z) \colon Y \subseteq \{x_1 \in X_1 \ldots\, x_i \in X_i\},$$
$$power_{low}(Y_x) \leq Z_x \leq power_{high}(Y_x)\ for\ every\ Y_x \in Y$$
$$and\ power(Z) = j\ )$$

In other words, $A[i,j]$ denotes the maximum flow of subset Y for the first i compressors with at most one model per compressor such that power Z_x for every model $Y_x \in Y$ stays within its lower and upper limits and $power(Z)$ is exactly j. When element $A[i,j]$ does not exist, we specify that $A[i,j] = 0$. In order to extract the solution, we also fill tables P and J. Table P stores the power such that $P_{i,j}$ contains the power of compressor i that is used in $A_{i,j}$, and table J stores the model index such that $J_{i,j}$ contains the model index for compressor i that is used in $A_{i,j}$. We present the integer algorithm for N compressors next.

Algorithm 2: Integer algorithm that fills tables A, P, and J

```
$IntegerFillTables(X, N, power_{tot})$
1:   ▻ Input: A set $X = \{X_1, \dots, X_N\}$, where every $X_i \in X$ contains at most $k$
       compressor models for compressor $i$, a number of compressors $N$,
       and a total system power $power_{tot}$.
2:   ▻ Output: A table $A$ with $N$ rows and $power_{tot}$ columns, for which every
       $A[i, j]$ contains the highest flow for the first $i$ compressors and power $j$,
       a table $P$ that is similar to table $A$ except that it contains the power of
       compressor $i$ for total power $j$, and a table $J$ that is also similar to table
       $A$ except that this one contains the index of the shunt configuration of
       compressor $i$ for power $j$.
3:   Initialize $A_{i,j} \leftarrow 0$ for all $1 \leq i \leq N$ and $1 \leq j \leq power_{tot}$
4:   Initialize $P_{i,j} \leftarrow 0$ and $J_{i,j} \leftarrow 0$ for all $1 \leq i \leq N$ and $1 \leq j \leq power_{tot}$
5:
6:   for $i \leftarrow 1$ to $Length(X_1)$ do
7:     for $j \leftarrow power_{low}(X_{1,i})$ to $power_{high}(X_{1,i})$ do
8:       if $flow(X_{1,i}, j) > A_{1,j}$ then $A_{1,j} \leftarrow flow(X_{1,i}, j),\ J_{1,j} \leftarrow i,\ P_{1,j} \leftarrow j$
9:   end for
10:  end for
11:  for $i \leftarrow 2$ to $N$ do
12:    $A_i \leftarrow A_{i-1}$
13:    for $j \leftarrow 1$ to $Length(X_i)$ do
14:      for $k \leftarrow power_{low}(X_{i,j})$ to $power_{high}(X_{i,j})$ do
15:        $f \leftarrow flow(X_{i,j}, k)$
16:        if $f > A_{i,k}$ then $A_{i,k} \leftarrow f,\ J_{i,k} \leftarrow j,\ P_{i,k} \leftarrow k$
17:        for $l \leftarrow k + 1$ to $power_{tot}$ do
18:          if $A_{i-1,l-k} = 0$ then continue
19:          $a \leftarrow A_{i-1,l-k} + f$
20:          if $a > A_{i,l}$ then $A_{i,l} \leftarrow a,\ J_{i,l} \leftarrow j,\ P_{i,l} \leftarrow k$
21:        end for
22:      end for
23:    end for
24:  end for
25:  return $A, P, J$
```

The integer algorithm returns the sets A, P and J that make it possible to select the maximum flow rate for a power consumption efficiently. We execute the integer algorithm and fill the tables with a time complexity of.

$$O\left(\sum_{i=1}^{N}\sum_{x\in X_i}\left(power_{high}(x) - power_{low}(x)\right) * power_{tot}\right)$$

Let us define $power_{max}$ as the maximum power consumption of a compressor. Now, because we always have that $|X_i| \leq k+1$, $power_{high}(x) - power_{low}(x) \leq power_{max}$, and $power_{tot} \leq N * power_{max}$ we can simplify the time complexity to.

$$O(N^2 * k * {power_{max}}^2)$$

In order to implement the algorithm with real power consumptions, we must first convert them to integers. Unfortunately, this introduces rounding errors. In order to manage the rounding errors, we implement a *polynomial-time approximations scheme* as explained in section 3.2, which implies that we develop an algorithm for which we have a total power consumption of at most $(1+\varepsilon) * OPT$, with OPT being the optimal power consumption, and $\varepsilon > 0$ being the tuning parameter that defines the trade-off between accuracy and speed.

To the end of converting the real powers to integer powers, let us define $power_{LB}$ as a lower bound on the optimum power consumption such that we always have that $OPT \geq power_{LB}$. We calculate the lower bound power consumption by assuming that we have just one cell with the minimum cell voltage $umin$ that meets the flow-rate demand. We first calculate the required current for demand d, which we then multiply with the minimum cell voltage $umin$ to get the lower bound power consumption. The minimum cell voltage is calculated as the pressurization voltage $u0$, plus the minimum voltage over the resistive element in the cell. Because the minimum cell current is the back-diffusion compensation current $icomp$, we take the product of $icomp$ and the minimum cell resistance $rmin$ to calculate the minimum cell voltage in the equation below.

$$power_{LB} = \frac{d}{H/2F} * umin$$

Equation 12: Lower bound power consumption

$$umin = icomp * rmin + u0$$

We further simplify the lower bound power consumption and convert the real powers to integer powers, with the following equation.

$$power^{*}(x) = \left\lceil \frac{power(x)}{\Delta} \right\rceil$$

$$\Delta = \varepsilon * \frac{power_{LB}}{N}$$

Equation 13: Converting to integer power

$$power_{LB} = \frac{2F * d * umin}{H}$$

We tune the scaling with parameter $\varepsilon > 0$, where ε defines the maximum rounding error as $error \leq \ \varepsilon * power_{LB}\ /\ N$.

We can now convert the real powers to integer powers and execute the following algorithm.

Algorithm 3: Fill system tables

```
$FillTables(X, N, K, d, umin, \varepsilon)$
1:    ▻ Input: A set $X = \{X_1, \dots, X_N\}$, where every $X_i \in X$ contains at most $k$
        compressor models for compressor $i$, a number of compressors $N$, a
        total number of cells $K$, a flow demand $d$, a minimum cell voltage $umin$,
        and a tuning parameter $\varepsilon$.
2:    ▻ Output: A table $A$ with $N$ rows and $power_{tot}$ columns, for which every
        $A[i, j]$ contains the highest flow for the first $i$ compressors and power $j$,
        a table $P$ that is similar to table $A$ except that it contains the power of
        compressor $i$ for total power $j$, a table $J$ that is also similar to table $A$
        except that this one contains the index of the shunt configuration of
        compressor $i$ for power $j$, and a conversion factor $\Delta$.
3:
4:    ▻ Convert to integers
5:    $power_{tot} \leftarrow 0$
6:    **for** $i = 1$ **to** $N$ **do** $power_{tot} \leftarrow power_{tot} + power_{high}(X_{i,last})$
7:   $power_{LB} \leftarrow 2F * d * umin / H$
8:   $\Delta \leftarrow \varepsilon * power_{LB} / N, \quad {power_{tot}}^* \leftarrow \lceil power_{tot}/\Delta \rceil$
9:   ${power_{low}}^*(x) \leftarrow \lceil power_{low}(x)/\Delta \rceil$ for every $x \in X_{i,j}$
10:  ${power_{high}}^*(x) \leftarrow \lfloor power_{high}(x)/\Delta \rfloor$ for every $x \in X_{i,j}$
11:  $flow^*(x, p) \leftarrow flow(x, p * \Delta)$ for every $x \in X_{i,j}$ and
                                          ${power_{low}}^*(x) \le p \le {power_{high}}^*(x)$
12:
13:  Compute tables $A$, $P$ and $J$ with algorithm $IntegerFillTables$ for the
     number of compressors $N$, total power ${power_{tot}}^*$, power consumptions
     ${power_{low}}^*(x)$, ${power_{high}}^*(x)$ instead of $power_{low}(x)$, $power_{high}(x)$
     respectively, and, $flow^*(x, p)$ instead of $flow(x, p)$.
14:  **return** $A, P, J, \Delta$
```

With the conversion to integers included, we find a resulting time complexity of.

$$O\left(\frac{N^3 * k * {power_{max}}^2}{\epsilon * power_{LB}}\right) = O\left(\frac{N^3 * k * {power_{max}}^2}{\epsilon * d * umin}\right)$$

We report the final solution with the following algorithm.

Algorithm 4: Select system configuration

```
$SelectSystemConfig(X, N, A, P, J, d, \Delta)$
1:   ▻ Input: A set $X = \{X_1, \ldots, X_N\}$, where every $X_i \in X$ contains at most $k$
       compressor models for compressor $i$, a number of compressors $N$, a
       table $A$ with $N$ rows containing the maximum flows, a table $P$
       containing the compressor power for the respective element in $A$, a
       table $J$ containing the shunt configuration index for the respective
       element in $A$, a flow rate demand $d$, and a conversion factor $\Delta$.
2:   ▻ Output: A set $Y$ that contains for every compressor the selected
       model, and a set $Z$ that contains the power for the respective
       compressor.
3:
4:   $Y \leftarrow \emptyset, Z \leftarrow \emptyset, fmax \leftarrow 0$
5:   **for** $i = 1$ **to** $|A_N|$ **do**
6:      **if** $A_{N,i} > fmax$ **then** $fmax \leftarrow A_{N,i}, j \leftarrow i$
7:      **if** $A_{N,i} \geq d$ **then** $j \leftarrow i$ **break**
8:   **end for**
9:   **for** $i = N$ **to** 1 **do**
10:     **if** $j = 0$ **or** $J_{i,j} = 0$ **then** $Y \leftarrow Y \cup X_{i,1}, Z \leftarrow Z \cup 0$ **continue**
11:     $k \leftarrow J_{i,j},\ Y \leftarrow Y \cup X_{i,k},\ Z \leftarrow Z \cup \Delta * P_{i,j}$
12:     $j \leftarrow j - P_{i,j}$
13:  **end for**
14:  **return** $Y, Z$
```

Algorithm 4 uses the previously constructed tables P and J to extract the system configuration that meets the demand and consumes the least integer power in $O(N^2 * power_{max} \,/\, \varepsilon * d * umin)$ time. When a compressor is not used, we assign a configuration that has no shunts enabled such that we avoid the reversion of hydrogen flow as explained in chapter 4. When the demand cannot be met, we output the configuration that gives the highest flow rate.

With the presented algorithms, we can find a solution in polynomial time. Unfortunately, we cannot guarantee an optimal solution anymore. Instead, we guarantee that we find the solution that meets demand d when possible, and for which we have that $power \leq (1+\epsilon) * OPT$, with OPT being the optimum power consumption.

Proof. To prove that the error is at most $\varepsilon * OPT$. Let the set $Sopt$ be the optimal subset for a given input with $power(Sopt) = OPT$. Let S denote the subset returned by the algorithm. Since we did not change the flow rates, subset Y has flow at least d. The computed solution is feasible. We must now show that $power(S) \leq (1+\varepsilon) * OPT$. Because S is optimal for the new values, we have that $power^*(S) \leq power^*(Sopt)$. Moreover, we have.

$$\frac{power(x)}{\Delta} \leq power^*(x) \leq \frac{power(x)}{\Delta} + 1$$

$$\begin{aligned} power(S) &= \textstyle\sum_{x \in S} power(x) \leq \sum_{x \in S} \Delta * power^*(x) \\ &= \Delta \textstyle\sum_{x \in S} power^*(x) \leq \Delta * \sum_{x \in Sopt} power^*(x) \\ &\leq \Delta * \textstyle\sum_{x \in Sopt} \frac{power(x)}{\Delta} + 1 \\ &= \textstyle\sum_{x \in Sopt} power(x) + \Delta * |Sopt| \\ &\leq \textstyle\sum_{x \in Sopt} power(x) + \Delta * N \\ &= OPT + \varepsilon * power_{LB} \leq OPT + \varepsilon * OPT \end{aligned}$$

We can conclude that $power(S) \leq (1+\varepsilon) * OPT$.

We have found which shunt configurations to pick, which compressors to use and how much power to give to the compressors. To calculate how much current we need per compressor, we rewrite equation 7 to the following quadratic formula that solves for compressor current I, using the offset voltages U, equivalent resistances R and power P.

$$I = \frac{-\sum U + \sqrt{(\sum U)^2 + 4\,P\,\sum R}}{2\,\sum R}$$

Equation 14: Convert power to current

5.3. Problem P3 – adapt the algorithm such that it can be implemented in the proposed compression system

It remains us to adapt the algorithm such that it can be implemented in the proposed hydrogen compression system. This includes n skid nodes that are connected to each other in the outer network, and at most m compressors per skid that are connected to each other in the inner network. With the approach proposed next, we implement the algorithm in a distributed manner both to decrease runtime and to decrease bus utilization.

We propose to execute the algorithm in four sequential steps.

1. Each compressor first executes the $FindOptimalShuntConfigs$ algorithm and transmits the Pareto-optimal shunt configurations to the respective skid node as illustrated with the red arrows below.

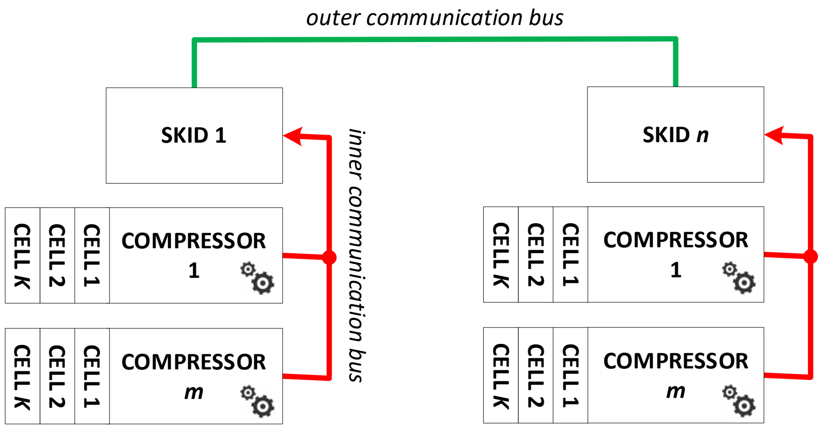

Figure 18: Communicating the Pareto-optimal shunt configurations

2. When the models are received by the skid, it executes the $FillTables$ algorithm for the compressors within the skid but with $\Delta \leftarrow \varepsilon * power_{LB} / N_{SYS}$, for the total number of compressors in the system

N_{SYS}. When done, every skid broadcasts the maximum flow rates per power consumption, i.e. the last row from table A, to the other skids as illustrated by the red arrows below.

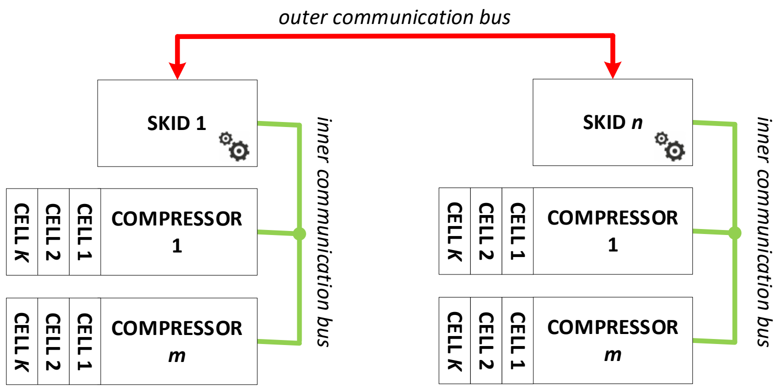

Figure 19: Broadcasting the maximum flow rates per power consumption values

3. When all values are received, every skid executes a variation of the $IntegerFillTables$ algorithm with the broadcasted values as input. The key here is that every skid uses the same row ordering when filling the tables such that all skids select the same solution. Every skid can now find for the compressors within the skid, which shunt configurations to use and how much power to give, similar as implemented in $SelectSystemConfig$. The last step is for the skids to communicate the selected shunt configuration and requested power to the compressors, illustrated by the red arrows below.

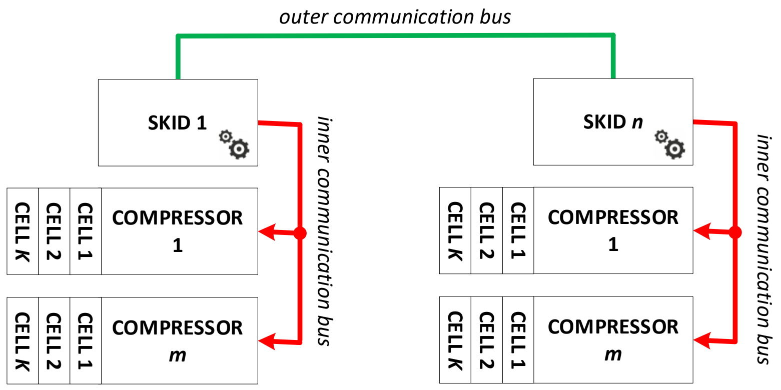

Figure 20: Communicating the selected shunt configurations and requested power consumptions

4. All that is left is for the compressors to enable the selected shunt configuration, convert the requested power to a compressor current with equation 14, and set the newly calculated current.

Since we used the total number of compressors in the system N_{SYS} when converting to integers, we can still guarantee a power consumption that is at most $(1+\varepsilon) * OPT$. The complete algorithms for the compressor and skid are provided in Appendix A and Appendix B respectively.

Chapter 6

Results

In this chapter are the computational, space, and communicational complexities presented for the adaption described in section 5.3 and implemented in Appendix A and Appendix B. To the end of quantizing these complexities, let us define a system with a variable skid size of $1 \leq n \leq 100$ skids, with $m = 10$ compressors per skid and $k = 120$ cells per compressor. We define the pressurization voltage $u0 = 60\ mV$, and the compensation current $icomp = 5\ A$. We have the limits $umax = 600\ mV$ and $imax = 200\ A$. Let us also define cell resistance $R_c = 2\ m\Omega$, shunt resistance $R_s = 0.6m\Omega$, and cell plate resistance $R_p = 1m\Omega$. Such a system would be able to achieve a mass-flow rate of over $21\ Mg/day$ when $n = 100$. In this system, we will use a demand d that is 80% of the maximum flow rate and consider a maximum deviation of 1%, 2%, and 5% from the optimum power consumption, i.e. $\varepsilon = 0.01$, $\varepsilon = 0.02$ and $\varepsilon = 0.05$.

6.1. Computational complexity

We will first consider the time complexity and find that after implementing the algorithm, we have a computational complexity of $O(k^3)$ for the compressor node. For the skid node we use the minimum cell voltage $umin$ and the maximum compressor power $power_{max}$ to define the computational complexity as.

$$O\left(\frac{n * m^3 * k * {power_{max}}^2}{\varepsilon * d * umin} + \frac{n^4 * m^4 * {power_{max}}^2}{(\varepsilon * d * umin)^2}\right)$$

To give some contrast, let us consider a simple exhaustive implementation in which for every combination of compressors and shunt configurations the optimal compressor currents are calculated. Let us assume that calculating the optimum compressor currents is done in constant time, we would still need to solve every combination. This gives us a computational complexity of $O(2^k * k^2)$ for the compressor and $O(2^{k*n*m})$ for the skid. Or in other words, around 10^{40} computations for the compressor and 10^{40000} computations for the skid. If we compare this to the presented algorithm, we have just 10^6 computations for the compressor and at most 10^{21} computations for the skid when $\varepsilon = 0.01$.

To find out how the algorithm performs in an embedded environment, we implement the compressor algorithm on a cortex-M3 processor running at $60\ MHz$, and the skid algorithm on a cortex-A9 processor running at $1\ GHz$. We find an execution time of $110\ ms$ for the compressor node and we plot the execution time for the skid below.

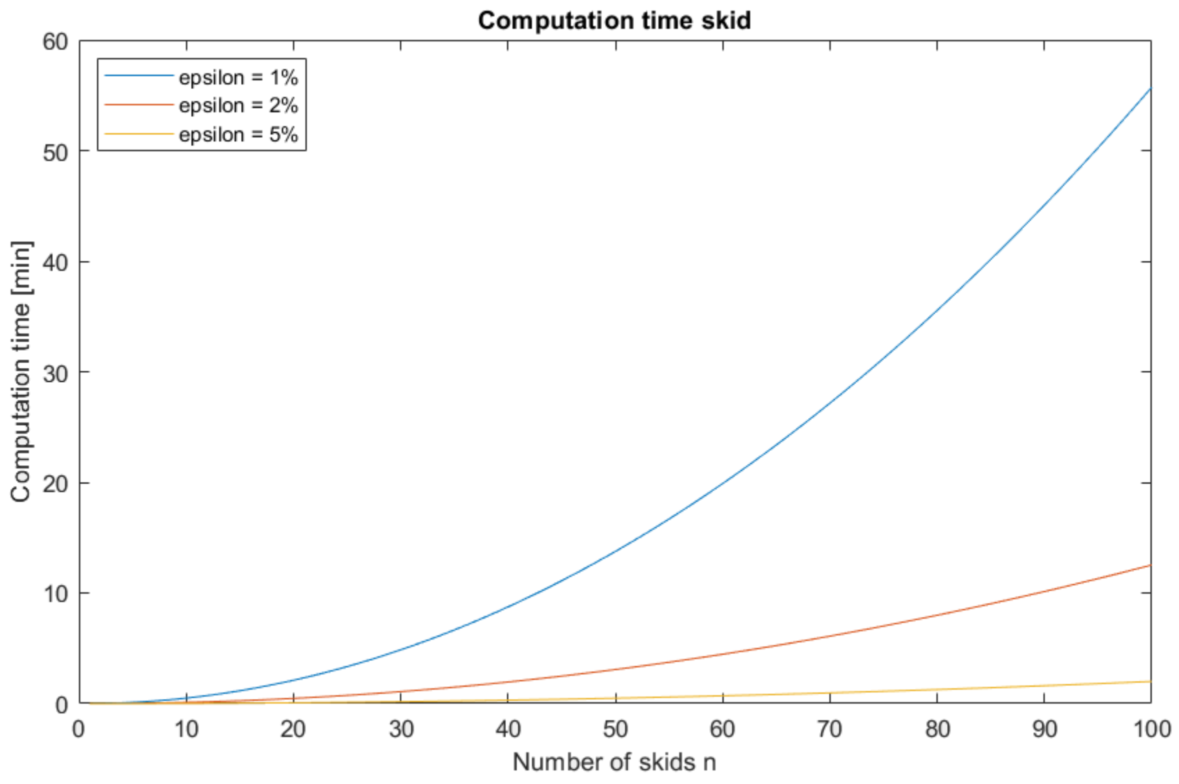

Figure 21: Skid computation time

6.2. Space complexity

The compressor node must be able to store $k + 1$ models, a single model consists of $O(k)$ elements, therefore we need to store $O(k^2)$ elements in the compressor node. The skid node needs much more memory as it stores first the models of the compressors. Secondly, the tables A, AP and AJ for the compressors within the skid. Thirdly, the last row of table A for all skids in the system. And lastly, a system table that consists of tables BA and BJ. For a minimum cell voltage $umin$ and a maximum compressor power $power_{max}$, we find a space complexity of.

$$O\left(m * k^2 + \frac{n * m^3 * power_{max} + n^2 * m^2 * power_{max}}{\varepsilon * d * umin}\right)$$

For the system in question, we find that we need just over 230 kB of memory for the compressor. For the skid, we find that the memory usage scales linearly with the number of skids in the system, see the plot below.

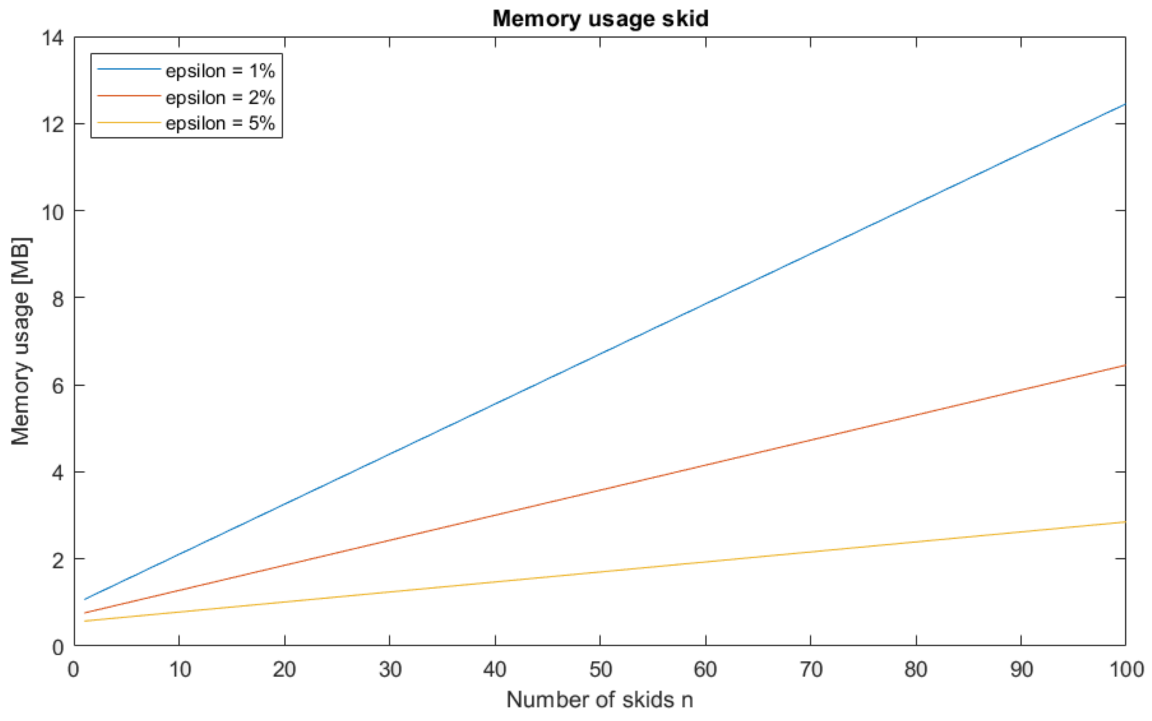

Figure 22: Memory usage skid

6.3. communicational complexity

Because we implement the algorithm in a distributed manner, with multiple nodes connected through a communication bus, we must also consider the communicational complexity. We find that for the inner network, we communicate at most $O(m * k)$ models. With each model consisting of $O(k)$ elements, we communicate a total of $O(m * k^2)$ elements in the inner network. In the outer network, we broadcast for each skid the last row of table A. For a minimum cell voltage $umin$ and a maximum compressor power $power_{max}$, we find a communicational complexity of.

$$O\left(\frac{n^2 * m^2 * power_{max}}{\varepsilon * d * umin}\right)$$

For the system in question, let us assume for both the inner and outer networks a bus with a bit rate of $1\,Mbit/s$. In the worst-case, we require just over $15s$ of communication time in the inner network. For the outer network we find a linear relationship between communication time and the number of skids, see the plot below.

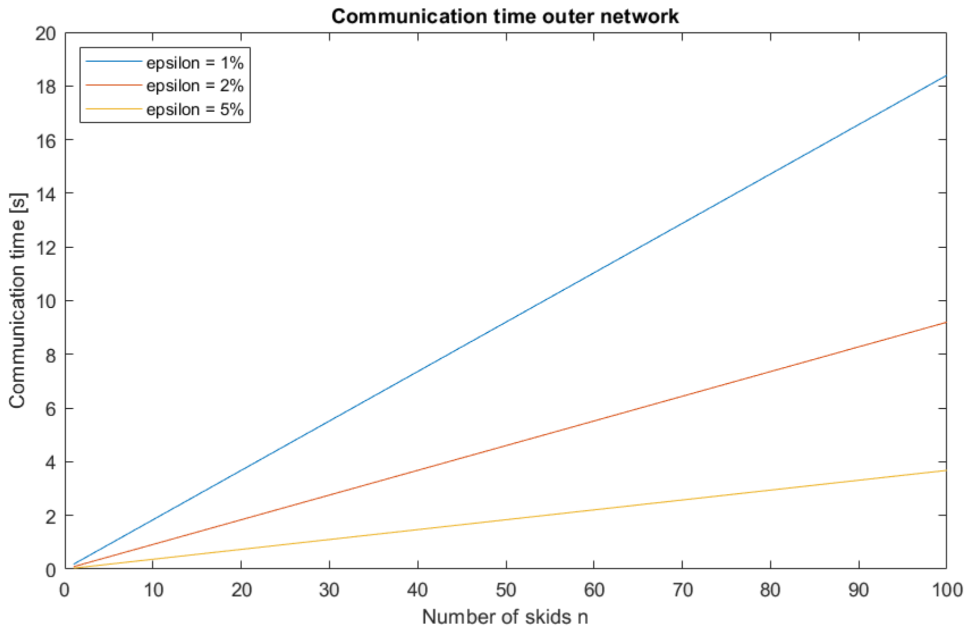

Figure 23: Outer network communication time

Chapter 7

Conclusion

In this thesis is a load-sharing algorithm presented for an electrochemical hydrogen compression system. The presented algorithm is able to decide which cells to shunt and how much current to feed through the compressors such that we meet a certain hydrogen flow rate demand, respect the cell voltage and compressor current limitations, and achieve near-optimal power consumption. More precisely, we guarantee a power consumption of at most $(1+\varepsilon) * OPT$ for some tuning parameter $\varepsilon > 0$, and the optimum power consumption OPT. Moreover, we achieve this result in polynomial time and with polynomial space requirements. Furthermore, for the proposed compression system, we have adapted the algorithm to a distributed implementation with a polynomial communicational complexity. We conclude that we have successfully solved the problem statement and completed the related milestones.

Bibliography

Bellman, R. (1957). *Dynamic programming.* Princeton: Princeton university press.

Blanchet, S., Yoon, W., & Quet, P.-F. (2014). *United States Patent No. US9915004B2.*

Bretthauer, K., & Shetty, B. (1995). The nonlinear resource allocation problem. *Operations Research*, 670-683.

Bretthauer, K., & Shetty, B. (2002). A pegging algorithm for the nonlinear resource allocation problem. *Computers and Operations Research*, 505–527.

Bretthauer, K., & Shetty, B. (2002). The nonlinear knapsack problem – algorithms and applications. *European Journal of Operational Research*, 459-472.

Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research*, 266-277.

de Berg, M. (2019). *Advanced Algorithms (2IMA10).* Eindhoven.

Faraday, M. (1834). On Electrical Decomposition. *Philosophical Transactions of the Royal Society*, 77-122.

Grigoriev, S., Shtatniy, I., Millet, P., Porembsky, V., & Fateev, V. (2011). Description and characterization of an electrochemical hydrogen compressor/concentrator based on solid polymer electrolyte technology. *Hydrogen energy*, 4148-4155.

Macdonald, J. (1971). Electrical response of materials containing space charge with discharge at the electrodes. *The Journal of Chemical Physics*, 2026.

Mathews, G. (1897). On the partitioning of numbers. *Proceedings of the London Mathematical*, 486–490.

McAllister, W. (2020, April 3). *Delta-Wye resistor networks*. Retrieved from Khan Academy: https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-delta-wye-resistor-networks

Randles, J. (1947). Kinetics of rapid electrode reactions. *Discussions of the Faraday Society*, 11-19.

Rohland, B., Eberle, K., Ströbel, R., Scholta, J., & Garche, J. (1998). Electrochemical hydrogen compressor. *Electrochimica Acta, Volume 43, Issue 24*, 3842.

Scheepers, F., Stähler, M., Stähler, A., Rauls, E., Müller, M., Carmo, M., & Lehnert, W. (2019). Improving the Efficiency of PEM Electrolyzers through Membrane-Specific Pressure Optimization. *Energies*, 612.

Sedlak, J., Austin, J., & LaConti, A. (1981). Hydrogen recovery and purification using the solid polymer electrolyte electrolysis cell, Volume 6, Issue 1. *International Journal of Hydrogen Energy*, 45-51.

Ströbel, R., Oszcipok, M., Fasil, M., Rohland, B., Jorissen, L., & Garche, G. (2002). The compression of hydrogen in an electrochemical cell based on a PE fuel cell design. *Power Sources*, 208-215.

Suermann, M., Kiupel, T., Schmidt, T., & Buchi, F. (2017). Electrochemical Hydrogen Compression: Efficient Pressurization Concept Derived from an Energetic Evaluation. *Journal of The Electrochemical Society*, 1187-1195.

Williams, J. W. (1964). Algorithm 232 - Heapsort. *Communications of the ACM*, 347-348.

Appendix A.

Compressor algorithm

```
$ProcessCompressor(id, Rc, Rs, rp, u0, umax, icomp, imax, k)$
1:  ▻ Input: A compressor identifier $id$, a set $Rc = \{Rc_1, \dots, Rc_k\}$, with cell
       resistances, a set $Rs = \{Rs_1, \dots, Rs_k\}$ with shunt resistances, a cell plate
       resistance $rp$, an offset voltage $u0$, a compensation current $icomp$, a
       maximum cell voltage $umax$, a maximum compressor current $imax$, and
       a number of cells $k$.
2:  ▻ Output: A set $S = \{S_1, \dots, S_k\}$, with bits that indicate that the respected
       shunt is enabled when the bit is set, and a compressor current $I$.
3:
4:  ▻ Find, solve and transmit compressor models
5:  $X \leftarrow TransmitModels(Rc, Rs, rp, u0,, umax, icomp, imax, k)$
6:
7:  ▻ Receive model index $ix$ with power $p$ for compressor with $id$
8:  $[ix, p] \leftarrow ReceiveFromSkid(id)$
9:  $x \leftarrow X_{ix}$
10:
11: ▻ Calculate current and return results
12: $rsum \leftarrow \sum x.R,\ \ usum \leftarrow \sum x.U$
13: $I \leftarrow \left(\sqrt{usum^2 + 4 * p * rsum} - usum\right) / 2 * rsum$
14: **return** $x.S,\ I$
```

Appendix A.1. Find, solve and transmit compressor models

```
$TransmitModels(Rc, Rs, rp, u0, umax, icomp, imax, k)$
1:   ⊳ Input: A set $Rc = \{Rc_1, \dots, Rc_k\}$, with cell resistances, a set
       $Rs = \{Rs_1, \dots, Rs_k\}$ with shunt resistances, a cell plate resistance $rp$, a
       offset voltage $u0$, a maximum cell voltage $umax$, a compensation current
       $icomp$, a maximum compressor current $imax$, and a number of cells $k$.
2:   ⊳ Output: A set $X$ with at most $k + 1$ compressor models, where every
       model contains a set $Rc = \{Rc_1, \dots, Rc_k\}$ with cell resistances, a set
       $S = \{S_1, \dots, S_k\}$ with bits that represent the shunts, a set $R = \{R_1, \dots, R_k\}$
       with equivalent cell resistances, a set $U = \{U_1, \dots, U_k\}$ with offset
       voltages, a minimum compressor current $imin$, and a maximum
       compressor current $imax$.
3:   Initialize $S_i \leftarrow 0$ for all $1 \leq i \leq k$.
4:   $X \leftarrow \emptyset, fmax \leftarrow \emptyset, pnorm \leftarrow \emptyset$
5:
6:    ⊳ Get the sorted indices for $Rc$ and solve models
7:   $J \leftarrow SortDescend(\{0, \dots, k\}, Rc)$
8:   **for** $i = 1$ **to** $k + 1$ **do**
9:     **if** $i > 1$ **then** $j \leftarrow J_{i-1},\ S_j \leftarrow 1$
10:    $X_i \leftarrow SolveModel(S, Rc, Rs, rp, u0, umax, icomp, imax, k)$
11:    $pnorm_i \leftarrow powernorm(X_i, icomp, u0)$
12:    $fmax_i \leftarrow flowmax(X_i, icomp, u0)$
13:  **end for**
14:
15:  ⊳ Only transmit models that belong to the Pareto-optimal set
16:  **for** $i = 1$ **to** $k + 1$ **do**
17:    $add \leftarrow True$
18:    **for** $j = 1$ **to** $k + 1$ **do**
19:      **if** $i = j$ **then continue**
20:      **if** $fmax_j \geq fmax_i$ **and** $pnorm_j \leq pnorm_i$ **then** $add \leftarrow False$, **break**
21:    **end for**
22:    **if** $add$ **then** $TransmitToSkid(i, x)$
23:  **end for**
24:  **return** $X$
```

Appendix A.2. Solve model

```
$SolveModel(S, Rc, Rs, rp, u0, umax, icomp, imax, k)$
1:   ▻ Input: A set $S = \{S_1, \dots, S_k\}$, with bits that indicate that the respected
       shunt is enabled when the bit is set, a set $Rc = \{Rc_1, \dots, Rc_k\}$, with cell
       resistances, a set $Rs = \{Rs_1, \dots, Rs_k\}$ with shunt resistances, a cell plate
       resistance $rp$, an offset voltage $u0$, a maximum cell voltage $umax$, a
       compensation current $icomp$, a maximum compressor current $imax$,
       and a number of cells $k$.
2:   ▻ Output: A compressor model $X$ that contains as set $Rc = \{Rc_1, \dots, Rc_k\}$
       with cell resistances, a set $S = \{S_1, \dots, S_k\}$ with bits that represent the
       shunts, a set $R = \{R_1, \dots, R_k\}$ with equivalent cell resistances, a set
       $U = \{U_1, \dots, U_k\}$ with offset voltages a minimum compressor current
       $imin$, and a maximum compressor current $imax$.
3:   Initialize $X.Rc_i \leftarrow Rc_i$ for all $1 \leq i \leq k$
4:   Initialize $X.R_i \leftarrow Rc_i$ for all $1 \leq i \leq k$
5:   Initialize $X.U_i \leftarrow u0$ for all $1 \leq i \leq k$
6:
7:   ▻ Create sections
8:   $istart_1 \leftarrow 1, s \leftarrow S_1, n \leftarrow 1$
9:   **if** $k > 1$ **then**
10:    **for** $i \leftarrow 2$ to $k$ **do**
11:      **if** $s \neq S_i$ **then**
12:        $iend_n \leftarrow i - 1, s \leftarrow S_i, n \leftarrow n + 1$
13:        $istart_n \leftarrow i$
14:      **end if**
15:    **end for**
16: **end if**
17: $iend_n \leftarrow k$
18:
19: ▻ Solve sections
20: **for** $i = 1$ **to** $n$ **do**
21:    $is \leftarrow istart_i, ie \leftarrow iend_i$
22:    **if** $S_{is}$ **then**
23:      $[R, U] \leftarrow SolveShuntedSection(Rc_{is:ie}, Rs_{is:ie}, rp, u0, ie - is + 1)$
24:      $X.R_{is:ie} \leftarrow R,\ X.U_{is:ie} \leftarrow U$
25:    **end if**
```

```
26: **end for**
27:
28: ▻ Set current limits and return model
29: $X.imin \leftarrow \underset{1 \leq i \leq k}{\mathrm{Max}}((icomp * X.Rc_i - X.U_i + u0) \ / \ X.R_i)$
30: $X.imax \leftarrow \underset{1 \leq i \leq k}{\mathrm{Min}}(imax, \ (umax - X.U_i) \ / \ X.R_i)$
31: **return** $X$
```

Appendix A.3. Solve shunted section

```
$SolveShuntedSection(Rc, Rs, rp, u0, k)$
1:  ⊳ Input: A set $Rc = \{Rc_1, \dots, Rc_k\}$, with cell resistances, a set
       $Rs = \{Rs_1, \dots, Rs_k\}$ with shunt resistances, a cell plate resistance $rp$, an
       offset voltage $u0$, and a number of cells $k$.
2:  ⊳ Output: A set $R = \{R_1, \dots, R_k\}$ with equivalent cell resistances, and a set
       $U = \{U_1, \dots, U_k\}$ with offset voltages.
3:
4:  ⊳ Solve for current source
5:  Initialize $Rx_i \leftarrow 0$ for all $1 \le i \le k$
6:  Initialize $Ry_i \leftarrow 0$ for all $1 \le i \le k$
7:  Initialize $Rz_i \leftarrow rp$ for all $1 \le i \le k$
8:  **if** $k > 1$ **then**
9:     **for** $i = 2$ **to** $k$ **do**
10:        $Rz_i \leftarrow \frac{rp*(Rs_{i-1}+Rz_{i-1})}{Rc_{i-1}+Rz_{i-1}+Ry_{i-1}+Rs_{i-1}+rp}$
11:        $Ry_i \leftarrow \frac{rp*(Rc_{i-1}+Ry_{i-1})}{Rc_{i-1}+Rz_{i-1}+Ry_{i-1}+Rs_{i-1}+rp}$
12:        $Rx_i \leftarrow \frac{(Rc_{i-1}+Ry_{i-1})*(Rs_{i-1}+Rz_{i-1})}{Rc_{i-1}+Rz_{i-1}+Ry_{i-1}+Rs_{i-1}+rp}$
13:    **end for**
14: **end if**
15: $rpar \leftarrow \frac{1}{(1/(Ry_k+Rc_k))+(1/(Rz_k+Rs_k+rp))}$
16: $r \leftarrow \sum Rx + rpar$
17:
18: ⊳ Calculate equivalent cell resistances
19: $R \leftarrow \emptyset, Un \leftarrow \emptyset$
20: $Un_k \leftarrow rpar, ic \leftarrow Un_k/(Rc_k + Ry_k),\ R_k \leftarrow Rc_k * ic$
21: **if** $k > 1$ **then**
22:    **if** $k > 2$ **then**
23:       **for** $i = k - 1$ **to** $2$ **do**
24:          $Un_i \leftarrow Un_{i+1} + Rx_{i+1}$
```

```
            $ic \leftarrow \left(Un_i - \sum_{j=i+1}^{k} R_i\right)/(Rc_i + Ry_i)$
            $R_i \leftarrow Rc_i * ic$
        **end for**
    **end if**
    $R_1 \leftarrow r - \sum_{i=2}^{k} R_i$
**end if**

▻ Solve for voltage sources
$Rt \leftarrow \emptyset, Rb \leftarrow \emptyset$
Initialize $U_i \leftarrow u0$ for all $1 \leq i \leq k$
**for** $i = 1$ **to** $k$ **do**

    ▻ Simplify
    **if** $i > 1$ **then**
        $Rt_1 \leftarrow \frac{1}{(1/(Rc_1+Rs_1+rp))+(1/rp)}$
        **if** $i > 2$ **then**
            **for** $j = 2$ **to** $i - 1$ **do** $Rt_j \leftarrow \frac{1}{\left(1/\left(Rc_j+Rs_j+Rt_{j-1}\right)\right)+(1/rp)}$
        **end if**
        $rtt \leftarrow Rt_{i-1} + Rc_i + Rs_i$
    **else** $rtt \leftarrow Rc_1 + Rs_1 + rp$
    **end if**
    **if** $k > i$ **then**
        $Rb_k \leftarrow \frac{1}{(1/(Rc_k+Rs_k+rp))+(1/rp)}$
        **if** $k > i + 1$ **then**
            **for** $j = k - 1$ **to** $i + 1$ **do** $Rb_j \leftarrow \frac{1}{\left(1/\left(Rc_j+Rs_j+Rb_{j+1}\right)\right)+(1/rp)}$
        **end if**
        $rbb \leftarrow Rb_{i+1}$
    **else** $rbb = rp$
    **end if**
    $ic \leftarrow u0/(rtt + rbb)$

    ▻ Calculate offset voltages
    $U_i \leftarrow U_i - ic * Rc_i$
    **if** $i > 1$ **then**
```

```
59:       $icc \leftarrow ic$
60:       **if** $i > 2$ **then**
61:          **for** $j = i - 1$ **to** $2$ **do**
62:             $icc \leftarrow (icc * Rt_j)/(Rt_{j-1} + Rc_j + Rs_j)$
63:             $U_j \leftarrow U_j - (icc * Rc_j)$
64:          **end for**
65:       **end if**
66:       $icc \leftarrow (icc * Rt_1)/(Rc_1 + Rs_1 + rp)$
67:       $U_1 \leftarrow U_1 - (icc * Rc_1)$
68:    **end if**
69:    **if** $k > i$ **then**
70:       $icc \leftarrow ic$
71:       **if** $k > i + 1$ **then**
72:          **for** $j = i + 1$ **to** $k - 1$ **do**
73:             $icc \leftarrow (icc * Rb_j)/(Rb_{j-1} + Rc_j + Rs_j)$
74:             $U_j \leftarrow U_j - (icc * Rc_j)$
75:          **end for**
76:       **end if**
77:       $icc \leftarrow (icc * Rb_k)/(Rc_k + Rs_k + rp)$
78:       $U_k \leftarrow U_k - (icc * Rc_k)$
79:    **end if**
80: **end for**
81: **return** $R, U$
```

Appendix A.4. Functions

Appendix A.4.1. Calculate maximum flow

$$flowmax(x, icomp, u0) = \frac{H}{2F} \sum \frac{(x.imax - icomp) * x.R + x.U - u0}{x.Rc}$$

Where:
x is the compressor model.
$icomp$ is the back-diffusion compensation current $[A]$.
$u0$ is the pressurization voltage $[V]$.
H is the molar mass of *H2* $[g/mol]$.
F is the Faraday constant $[A * s/mol]$.

Appendix A.5. Calculate normalized current

$$currentnorm(x, icomp, u0) = icomp + \frac{2F/H + \sum (u0 - x.U)/x.Rc}{\sum x.R/x.Rc}$$

Where:
x is the compressor model.
$icomp$ is the back-diffusion compensation current $[\mathrm{A}]$.
$u0$ is the pressurization voltage $[\mathrm{V}]$.
H is the molar mass of *H2* $[g/mol]$.
F is the Faraday constant $[A * s/mol]$.

Appendix A.6. Calculate normalized power

$$\begin{aligned} powernorm(x, icomp, u0) \\ &= currentnorm(x, icomp, u0) \\ &* \sum currentnorm(x, icomp, u0) * x.R + x.U \end{aligned}$$

Appendix B.

Skid algorithm

```
$ProcessSkid(id, n, m, N, K, rmin, d, icomp, u0)$
1:  ▻ Input: A skid identifier $id$ that defines the system table row index, a
      number of skids in the system $n$, a number of compressors in the skid
      $m$, the total number of compressors $N$, the total number of cells $K$, the
      minimum cell resistance $rmin$, a flow-rate demand $d$, a compensation
      current $icomp$, and an offset voltage $u0$.
2:  $X \leftarrow \emptyset, IX \leftarrow \emptyset, AA \leftarrow \emptyset, AN \leftarrow \emptyset$
3:
4:  ▻ Calculate conversion factor
5:  $umin \leftarrow icomp * rmin + u0$
5:  $power_{LB} \leftarrow 2F * d * umin / H$
6:  $\Delta \leftarrow \varepsilon * power_{LB} / n * m$
7:
8:  ▻ Receive compressor models
9:  **for** $i = 1$ **to** $m$ **do**
10:     $[IX_i, X_i] \leftarrow ReceiveFromCompressor(i)$
11: **end for**
12:
13: ▻ Build tables and communicate with other skids
14: $[A, AP, AJ] \leftarrow BuildSkidTable(X, \Delta, m, d, icomp, u0)$
15: $BroadcastToSkids(A)$
16: **for** $i = 1$ **to** $n$ **do**
17:     **if** $i = id$ **then** $AA_i \leftarrow A, AN_i \leftarrow |A|$ **continue**
```

```
17:     $[AA_i, AN_i] \leftarrow ReceiveFromSkid(i)$
18: **end for**
19:
20: ⊳ Build the system table and select the best configuration
21: $[BA, BJ] \leftarrow BuildSystemTable(id, AA, AN, n, d)$
22: $[Y, Z] \leftarrow SelectSkidConfig(X, IX, BA, BJ, AP, AJ, \Delta, m, d)$
23:
24: ⊳ Transmit model index and power to compressors
25: **for** $i = 1$ **to** $m$ **do**
26:     $TransmitToCompressor(i, Y_i, Z_i)$
27: **end for**
28: **return**
```

Appendix B.1. Build skid table

```
$BuildSkidTable(X, \Delta, m, d, icomp, u0)$
1:  ⊳ Input: A set $X = \{X_1, \dots, X_N\}$ for which every $X_i \in X$ contains the
      compressor model for compressor $i$, a conversion value $\Delta$, a number of
      compressors in the skid $m$, a flow-rate demand $d$, a compensation
      current $icomp$, and an offset voltage $u0$.
2:  ⊳ Output: A row from table $A$ that contains the skid flows per power
      consumption, a table $P$ that is later used to find how much power each
      compressor needs to consume, and a table $J$ that is later used to find
      which shunt configuration to use.
3:
4:  ⊳ Calculate total skid power
5:  $power_{tot} \leftarrow 0$
6:  for $i = 1$ to $m$ do $power_{tot} \leftarrow power_{tot} + powermax(X_{i,last})$
7:  $power_{tot}^* \leftarrow \lceil power_{tot}/\Delta \rceil$
8:
9:  ⊳ Fill first row
10: Initialize $A_{i,j} \leftarrow 0$ for all $1 \leq i \leq 2$ and $1 \leq j \leq power_{tot}^*$
11: Initialize $P_{i,j} \leftarrow 0$ for all $1 \leq i \leq m$ and $1 \leq j \leq power_{tot}^*$
12: Initialize $J_{i,j} \leftarrow 0$ for all $1 \leq i \leq m$ and $1 \leq j \leq power_{tot}^*$
13: for $i \leftarrow 1$ to $Length(X_1)$ do
14:     $plow \leftarrow powerlow\_int(X_{1,i}, \Delta)$
15:     $phigh \leftarrow powerhigh\_int(X_{1,i}, \Delta)$
16:     for $j \leftarrow plow$ to $phigh$ do
17:         $f \leftarrow flowp(X_{1,i}, j * \Delta, icomp, u0)$
18:         if $f > A_{1,j}$ then $A_{1,j} \leftarrow f,\ P_{1,j} \leftarrow j,\ J_{1,j} \leftarrow i$
19:         if $f > d$ then break
20:     end for
21: end for
22: if $m = 1$ then return $A_1, P, J$
23:
24: ⊳ Fill remaining rows
25: for $i \leftarrow 2$ to $m$ do
26:     $ipre \leftarrow (i\ modulo\ 2) + 1$
```

```
27:     $icur \leftarrow \big((i-1)\ modulo\ 2\big) + 1$
28:     $A_{icur} \leftarrow A_{ipre}$
29:     **for** $j \leftarrow 1$ **to** $Length(X_i)$ **do**
30:       $plow \leftarrow powerlow\_int(X_{i,j}, \Delta)$
31:       $phigh \leftarrow powerhigh\_int(X_{i,j}, \Delta)$
32:       **for** $k \leftarrow plow$ **to** $phigh$ **do**
33:         $f \leftarrow flowp(X_{i,j}, k * \Delta, \mathrm{icomp}, u0)$
34:         **if** $f > A_{icur,k}$ **then** $A_{icur,k} \leftarrow f,\ J_{i,k} \leftarrow j,\ P_{i,k} \leftarrow k$
35:         **for** $l \leftarrow k + 1$ **to** ${power_{tot}}^{*}$ **do**
36:           **if** $A_{iprev,l-k} = 0$ **then continue**
37:           $a \leftarrow A_{iprev,l-k} + f$
38:           **if** $a > A_{icur,l}$ **then** $A_{icur,l} \leftarrow a,\ J_{i,l} \leftarrow j,\ P_{i,l} \leftarrow k$
39:           **if** $a > d$ **then break**
40:         **end for**
41:       **end for**
42:     **end for**
43:   **end for**
44:   **return** $A_{icur}, P, J$
```

Appendix B.2. Build system table

```
$BuildSystemTable(id, AA, AN, n, d)$
1:  ▻ Input: A skid identifier $id$ that defines the system table row index, a
       table $AA$ for which every row represents the maximum flow rates for
       a skid, a table $AN$ that contains for every row in $AA$ the data length, a
       number of skids in the system $n$, and a flow rate demand $d$.
2:  ▻ Output: A row from table $BA$ that contains the system flows per power
       consumption, and a row from table $BJ$ that is later used to find which
       skid configuration to use.
3:
4:  ▻ Calculate total system power
5:  $power_{tot} \leftarrow \sum_{i=1}^{n} AN_i$
6:
7:  ▻ Process first row
8:  Initialize $BA_{i,j} \leftarrow 0$ for all $1 \leq i \leq 2$ and $1 \leq j \leq power_{tot}$
9:  Initialize $BJ_{i,j} \leftarrow 0$ for all $1 \leq i \leq 2$ and $1 \leq j \leq power_{tot}$
10: **for** $j \leftarrow 1$ **to** $AN_1$ **do**
11:    $BA_{1,j} \leftarrow AA_{1,j}$
12:    **if** $id = 1$ **and** $BA_{1,j}$ **then** $BJ_{1,j} \leftarrow j$
13: **end for**
14: **if** $n = 1$ **then return** $BA_1, BJ_1$
15:
16: ▻ Process remaining rows
17: $nsum \leftarrow AN_1$
18: **for** $i \leftarrow 2$ **to** $n$ **do**
19:    $ipre \leftarrow (i \ modulo \ 2) + 1$
20:    $icur \leftarrow ((i - 1) \ modulo \ 2) + 1$
21:    $BA_{icur} \leftarrow BA_{ipre}$
22:    **for** $j \leftarrow 1$ **to** $AN_i$ **do**
23:       $f \leftarrow AA_{i,j}$
24:       **if** $f = 0$ **then continue**
25:       **if** $f > BA_{icur,j}$ **then**
26:          $BA_{icur,j} \leftarrow f$
27:          **if** $i = id$ **then** $BJ_{icur,j} \leftarrow j$
28:       **end**
```

```
29:       for $k \leftarrow 1$ to $nsum$ do
30:         if $BA_{iprev,k} = 0$ then continue
31:         $a \leftarrow BA_{iprev,k} + f$
32:         if $a > BA_{icur,k+j}$ then
33:           $BA_{icur,k+j} \leftarrow a$
34:           if $i = id$ then $BJ_{icur,k+j} \leftarrow j$
35:           else $BJ_{icur,k+j} \leftarrow BJ_{iprev,k}$
36:         end
37:         if $a > d$ then break
38:       end for
39:     end for
40:     $nsum \leftarrow nsum + AN_i$
41:   end for
42:   return $BA_{icur}, BJ_{icur}$
```

Appendix B.3. Select skid config

```
$SelectSkidConfig(IX, BA, BJ, AP, AJ, \Delta, m, d)$
1:  ⊳ Input: A set $IX = \{IX_1, \dots, IX_m\}$ for which every $IX_i \in IX$ contains the
      respective compressor model indices, a row $BA$ that contains the
      system flows per power consumption, a row $BJ$ that contains indices
      that match row $BA$ to rows $AP$ and $AJ$, a table $AP$ that contains the
      powers for the compressors in the skid, a table $AJ$ that contains the
      shunt configuration for the compressors in the skid, a conversion factor
      $\Delta$, a number of compressors in the skid $m$, and a flow rate demand $d$.
2:  ⊳ Output: A set $Y = \{Y_1, \dots, Y_m\}$ for which every $Y_i \in Y$ contains the
      compressor model index for compressor $i$, and a set $Z = \{Z_1, \dots, Z_m\}$ for
      which every $Z_i \in Z$ contains the power for compressor $i$.
3:
4:  $j \leftarrow 0, Y \leftarrow \emptyset, Z \leftarrow \emptyset, fmax \leftarrow 0$
5:  **for** $i = 1$ **to** $|BA|$ **do**
6:      **if** $BA_i > fmax$ **then** $fmax \leftarrow BA_i, j \leftarrow BJ_i$
7:      **if** $BA_i \geq d$ **then** $j \leftarrow BJ_i$ **break**
8:  **end for**
9:  **for** $i = m$ **to** $1$ **do**
10:     **if** $j = 0$ **or** $AJ_{i,j} = 0$ **then** $Y_i \leftarrow X_{i,1},\ Z_i \leftarrow 0$ **continue**
11:     $k \leftarrow AJ_{i,j}, Y_i \leftarrow IX_{i,k},\ Z_i \leftarrow \Delta * AP_{i,j}$
12:     $j \leftarrow j - AP_{i,j}$
13: **end for**
14: **return** $Y, Z$
```

Appendix B.4. Functions

Appendix B.4.1. Calculate maximum power

$$powermax(x) = x.imax\ \left(x.imax * \sum x.R + \sum x.U\right)$$

Appendix B.4.2. Calculate minimum integer power

$$powerlow\_int(x, \Delta) = \left\lceil \frac{x.imin\ (x.imin * \sum x.R + \sum x.U)}{\Delta} \right\rceil$$

Appendix B.4.3. Calculate maximum integer power

$$powerhigh\_int(x, \Delta) = \left\lfloor \frac{x.imax\ (x.imax * \sum x.R + \sum x.U)}{\Delta} \right\rfloor$$

Appendix B.4.4. Calculate mass-flow rate for current

$$flowi(x, I, icomp, u0) = \frac{H}{2F} \sum \frac{(I - icomp) * x.R + x.U - u0}{x.Rc}$$

Where:
x is compressor model.
I is the compressor current $[A]$.
$icomp$ is the compensation current $[A]$.
$u0$ is the pressurization voltage $[V]$.
H is the molar mass of *H2* $[g/mol]$.
F is the Faraday constant $[A * s/mol]$.

Appendix B.4.5. Calculate mass-flow rate for power

$$flowp(x, p, icomp, u0) = flowi\left(x, \frac{\sqrt{(\sum x.U)^2 + 4 * p * \sum x.R} - \sum x.U}{2 * \sum x.R}, icomp, u0\right)$$