

Algebraic Approaches for Gas Networks

Olmo Chiara Llanos

Born February 11th, 1995 in Sevilla, Spain

March 2nd, 2020

Master's Thesis Mathematics

Advisor: Prof. Dr. Jochen Garcke

Second Advisor: Dr. Bastian Bohn

INSTITUT FÜR NUMERISCHE SIMULATION

MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT DER
RHEINISCHEN FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

Acknowledgements

I would like to start this thesis by thanking Prof. Jochen Garcke for giving me the opportunity to work in a topic that bridges the two usually disconnected areas of algebra and applied mathematics, as well as for his comments and questions at all stages. I would also like to thank Dr. Bernhard Klaassen at Fraunhofer SCAI for his insight into the problems in modelling gas networks, as well as for his questions and remarks that were key in making this work.

I would also like to extend my gratitude to the professors at the INS and the Mathematical Institute, as well as the administration staff and the entirety of the Universität Bonn.

Of course, none of this would have been possible without the support of my family and friends, to whom I am deeply indebted.



Ludwig van Beethoven: 'Bagatelle No. 25 in A minor' (WoO 59, Bia 515).
Neue Briefe Beethovens. Ed.: Ludwig Nohl. Stuttgart: J.G. Cotta, 1867.

Contents

1	Introduction	1
2	Gas networks	3
2.1	Network description	3
2.2	Flow modelling through a pipe	5
2.3	Flow throughout the network	6
2.4	Obtaining a polynomial system	12
3	Gröbner basis	13
3.1	The ideal membership problem	13
3.1.1	Monomial ordering	15
3.1.2	Division algorithms	17
3.1.3	Leading terms and ideal membership	19
3.1.4	Elimination theory	20
3.2	Gröbner basis algorithms	23
3.3	Triangular sets	25
3.4	The Multivariate Quadratic problem	28
4	Solving gas networks	29
4.1	Limits of the Gröbner solver	30
4.2	Stability of the computations	34
4.2.1	Floating point errors	35
4.2.2	Approximation errors	39
4.3	Finding the flow directions	42
4.3.1	Parametrized flow directions	45
5	Network reduction	51
5.1	Topological reductions	51
5.1.1	Pruning	52
5.1.2	Cycle collapse	53
5.1.3	Path simplification	54
5.1.4	Dealing with unknown gas inputs	56
5.2	Node pressures	58
5.3	Improving the restrictions	60
5.4	The Irish gas network	63
6	Conclusions	73
	Bibliography	75

1 Introduction

Natural gas is currently one of the most important sources of energy in the world. It is used for electricity generation, but also to provide heating or to power vehicles. According to Eurostat data, natural gas accounted for 22.6 % of the total energy consumed by end users across the European Union.

Due to this, it is very important to guarantee the supply of natural gas from its production sites to the end users. As natural gas is not suited for transport in small vessels (like trucks or tank ships, for example), this supply takes place through large networks of gas pipelines. In these networks gas is injected at nodes and flows at a high pressure through some pipes, and at a lower pressure at others to supply consumers. It is therefore very important to have tools that can model adequately this type of networks.

One question we pose ourselves is the following: given a network and a load (a set of injections and withdrawals of gas at some of the nodes), is it possible to move the gas throughout the network in such a way that enough gas injected at supply points can be withdrawn to satisfy the demands of the nodes representing end users? If the answer to this question is positive, the load is said to be feasible. Because of European regulations (see Regulation (EC) No 715/2009), the operators of the network (known as Transmission System Operators, or TSOs) are required to be different and independent from the energy companies that use the network to transport the gas. The procedure used to move gas through the network is based around *nominations*: a gas supplier nominates ahead of time a series of nodes alongside a balanced injection and extraction of gas at some of them (meaning that the amount of gas injected into the system equals the amount of gas extracted). On the specified day, however, the supplier can choose to decrease that nomination. It is the duty of the TSO to ensure that the final load is still feasible so that the contract can be fulfilled.

We will study this nomination problem. Our goal will be, starting from a given gas network, to determine whether a given load is feasible, and how would the gas flow through the network in that case. A comprehensive overview of possible approaches to solve this problem can be found in Pfetsch et al. 2015.

After an adequate modeling of the gas flow, the problem can be reduced to finding a solution to a system of multivariate equations. To solve this system we study the theory of Gröbner basis, which will allow us to triangularize the system so that a solution can be found by finding roots of univariate polynomials.

Note that we will work with networks for gas transportation, but the general idea of using Gröbner basis to treat graphs with cycles can be extended further to other types of problems, for instance in electrical circuits (where this idea was already being used in

the 1990s) or networks for Hydrogen transportation. More generally, the techniques of Gröbner basis and triangular sets are useful tools to find accurate solutions of systems of polynomial equations in several variables, regardless of where the specific equations appear.

Objectives

The main objective of this thesis is to study the theoretical and practical limits of the Gröbner basis approach, in terms of the size and the topology of the network but also depending on the choice of coefficients. This allows us to determine bounds in the network in order for our algorithm to be competitive.

In order to study the sources of error in the algorithm, we use Gröbner systems to compute condition numbers of systems, and give arguments to support that the shape of the ideals we work with is optimal. We also introduce the concept of parametric ideals to work with networks with unknown flow directions while avoiding the need to loop over all possible directions. To solve these parametric ideals faster we make use of the theory of triangular sets.

Our main contribution is a topological reduction scheme designed to find a solution of a large problem by reducing it, solving several subproblems and then merging the different solutions together to obtain a solution for the larger problem.

Outline

Let us summarize the structure of the rest of this work. In chapter 2 we show how to derive a system of equations for the gas flow depending only on the flow through the fundamental cycles of the network, which allows us to reduce the dimension of the problem. The algebraic tools are developed in chapter 3, where the concept of Gröbner basis is introduced and studied. In chapter 4 we test the limits of the method by looking at the performance and the potential sources of error for the algorithm. Finally, in chapter 5, a multi-step reduction process is proposed and studied.

2 Gas networks

We start by introducing the basic theory of gas networks, obtaining the equations that govern the flow through pipes as well as the pressure at different points of the network. We will always limit ourselves to the particular case of *balanced, stationary, isothermal* gas networks. Balanced means that the sum of the inputs equals the sum of the outputs, so we limit ourselves to moving gas and do not store it. Stationary (also known as *steady state*) means that the inputs will be fixed numbers, and not be dependent on time, and that we are at a point of time where the flows through the network have stabilized and are also constant. Isothermal means that the gas will be at a constant temperature (in particular, at the standard condition of 273.15 K).

2.1 Network description

We will first explain which elements constitute a gas network. While in real life there are many components serving different functions, we will limit ourselves to just a few of them.

- The most basic elements are *nodes* and *pipes*. Pipes are the basic structure through which the gas flows from one place to another, nodes are points at which one can access the pipe and therefore either introduce gas into the network or remove it, as well as junctions between two or more pipes.
- *Valves* control the activation state of the pipe on which they are installed. If the valve corresponding to a pipe is open, the pipe will function normally, as if the valve did not exist. If, on the other hand, the valve is closed, then there will be no flow through that pipe.
- *Control valves* act like normal valves, but with additional security measures. First of all, it forces the gas to flow through the pipe in one direction, otherwise it closes. Whenever the flow exceeds a preset value (which could be a safety limit), the valve closes too. Finally, it forces the pressure to be within given bounds: in particular, it guarantees that the outgoing pressure will always be above a certain value, thus limiting the pressure loss.
- *Compressors* increase the pressure of the gas, to counter the pressure lost when the gas is moved around. Unlike control valves, they can increase the pressure to a value above the incoming pressure of the gas, and are therefore used to ensure that along a long pipe the pressure is always kept above certain levels. They will also force a direction on the flow of the gas. See Hiller and Walther 2017 for a large

survey on compressor stations.

We will work with models that consist only of nodes and pipes. For valves, we will suppose that they are always open, and therefore they can be reduced to a standard pipe. Otherwise, we will simply remove the pipe on which they are located from the model. For control valves and compressors, we will consider them as if they were simple pipes. In order to describe our networks we use the tools of graph theory. A network instance becomes a graph $G = (V^*, E)$ where the vertices V^* are the nodes of the network and the edges E are the pipes. We will use those names interchangeably. We will also assume that our graph G is connected; otherwise, one could simply treat each component separately. Since we are dealing with flow through pipes, it makes sense to also consider an associated directed graph $\vec{G} = (V^*, A)$ with the same vertices but a set A of arcs, with $|A| = |E|$. Ideally one would define this digraph by orienting every edge in E according to the direction of the gas flow. However, this direction is not known to us beforehand, so we have to find another way.

To get a directed graph, we start by building a tree \mathcal{T} by performing a depth-first search on the graph, starting at a root node $r \in V^*$ that we fix. This gives us a predecessor function

$$\begin{aligned} p : V^* - \{r\} &\longrightarrow V^*, \\ u &\longmapsto p(u), \end{aligned}$$

that associates to each vertex u the previous vertex in the order defined by the depth-first search (so, the previous vertex in the path linking r to u in \mathcal{T}).

Lemma 2.1. For any edge $a = (u, v) \in E(G)$, we must have that there is some natural number n such that either $p^n(u) = v$ or $p^n(v) = u$.

Proof. Assume u appears first in the depth-first search, and let us place ourselves in the moment where the algorithm has chosen u . Then, since we are going for depth-first and u has an edge to v , we know that before the algorithm leaves u it will have added to \mathcal{T} either the edge (u, v) or a path P linking the two vertices. But then either $p(v) = u$ or, if P is formed by n edges, $p^n(v) = u$. \square

The previous lemma says that we can orient any edge (u, v) to get the arc

$$\vec{a} = (p^n(v), v) \in A(\vec{G}),$$

if u appears first, or with the inverted direction if v appears first.

Lemma 2.2. The graph \vec{G} constructed this way is acyclic.

Proof. Assume $u_1 \rightarrow u_2 \rightarrow \dots \rightarrow u_k \rightarrow u_1$ is a cycle. Then

$$u_1 = p^{n_1}(u_2) = p^{n_1}(p^{n_2}(u_3)) = \dots = p^{n_1}(p^{n_2}(\dots p^{n_k}(u_1))) = p^{n_1 + \dots + n_k}(u_1),$$

meaning u_1 is a predecessor of itself, which is impossible. \square

To any graph representing a network we associate a series of quantities representing the flow of gas, as well as its pressure. Any vertex and any arc has an associated volumetric gas flow, measured in $\text{m}^3 \text{s}^{-1}$ (other units of volume per time are of course possible). As mentioned before, for vertices this is called the load (since it refers to the amount of gas we add or remove from the system), and is denoted by q_v^{nom} for the junction v . For edges, this is the volumetric flow rate, and we denote it by $Q_{0,a}$ for arc a . We can also measure the pressure at every node v , which we denote by p_v , and which we measure in Pa. Note that the pressure of the gas along a pipe varies, which is crucial for the model.

Of course the flows are in general real numbers, and the pressure is always positive. This means that we get vectors

$$\begin{aligned} p^* &= (p_{u_1}, \dots, p_{u_{|V|}}) \in \mathbb{R}_{\geq 0}^{|V|}, \\ q^{\text{nom}*} &= (q_{u_1}^{\text{nom}}, \dots, q_{u_{|V|}}^{\text{nom}}) \in \mathbb{R}^{|V|}, \\ Q_0 &= (Q_{0,a_1}, \dots, Q_{0,a_{|A|}}) \in \mathbb{R}^{|A|} \end{aligned}$$

that define the state of the network. If the flow $Q_{0,a}$ at arc a is positive, it means that the gas flows in the direction of the arc \vec{G} ; if it is negative, the flow goes against the direction. If the load at a node is positive it means that we remove that volume of gas from the system at that point; if it is negative it means we insert it. If the load is zero, then the vertex acts as a simple junction between several pipes, and there is no gas inserted or removed.

Since we wish for our network to be stationary, we must require additionally that

$$\sum_{u \in V} q_u^{\text{nom}} = 0 \iff \mathbf{1}^T q^{\text{nom}} = 0. \quad (2.1)$$

This means that one of the loads is redundant. To simplify, we will choose the node r that we have taken as the root for \mathcal{T} and ignore its load as it can be derived from the rest of the system.

2.2 Flow modelling through a pipe

As it could be expected, it is not easy to model how the gas flows through a pipe. This depends on many variables related to the pipe geometry (diameter, slope) or the pipe construction (friction), as well as to considerations about the gas itself (its viscosity).

Broadly, for a pipe $a = (u, v)$ along the x axis one would have a system

$$A \frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial x} = 0, \quad (2.2)$$

$$\frac{\partial p}{\partial x} + \lambda \frac{|v|v}{2D} \rho + \frac{1}{A} \frac{\partial q}{\partial t} + g \rho s + \frac{1}{A} \frac{\partial \rho v^2}{\partial x} = 0. \quad (2.3)$$

where A and D are the area of the section of the pipe and its diameter, respectively, ρ is the density of the gas, q is the mass flow, λ is the friction factor, v is the velocity of the

gas through the pipe, g is the acceleration due to gravity, and s is the slope of the pipe. Since our network is stationary, all the time derivatives vanish, and therefore eq. (2.2) gives

$$\frac{\partial q}{\partial x} = 0.$$

This means that the mass flow through the entire length of the pipe (given by x) is constant (this justifies *a fortiori* the definition of $Q_{0,a}$ as a real number and not a function of the position along the pipe, because $Q_{0,a} = q/\rho_0$ where ρ_0 is the density of the gas under normal conditions, of 273.15 K and 101 325 Pa). We assume that our model is isothermal (meaning that the gas is always at the same temperature at every point of the network) and planar (meaning that there is no height difference at any point, so $s = 0$). We ignore the last term of the second equation since it gives a very small contribution, and arrive to

$$\frac{\partial p}{\partial x} + \lambda \frac{|v|v}{2D} \rho = 0$$

The flow satisfies $A\rho v = q$, as Av is the volumetric flow through a section of area A of the pipe and ρ translates the volume into mass. Since $q = \rho_0 Q_{0,a}$, we get

$$\frac{\partial p}{\partial x} = -\lambda \frac{Q_{0,a}|Q_{0,a}|}{2D\rho}.$$

Further simplifications of the remaining ρ term lead to

$$2p \frac{\partial p}{\partial x} = -\psi |Q_{0,a}| Q_{0,a}$$

where ψ depends on many parameters of the gas and the pipes. Integrating, this gives

$$p_v^2 - p_u^2 = -\phi |Q_{0,a}| Q_{0,a}, \quad (2.4)$$

where u and v are the start and end, respectively, nodes of the pipe. This means that, while the flow remains constant, the pipe causes a pressure drop between the start and end nodes. We call ϕ the *pressure drop coefficient*.

2.3 Flow throughout the network

The former equation relates pressures to flow. We still need to relate flows through pipes and through junctions, and we do so by looking at mass conservation. At each junction v one has

$$q_v = \sum_{\text{head}(a)=v} Q_{0,a} - \sum_{\text{tail}(a)=v} Q_{0,a}, \quad (2.5)$$

that is, in order for the network to be balanced, the net sum of gas flow to junction v must be exactly what we remove/insert from it. This in practice is equivalent to the role of Kirchhoff's law of currents in the analysis of electrical circuits.

Remark 2.3. Kirchhoff's loop rule, stating that the sum of voltages along a cycle adds up to zero, would be equivalent to saying that the pressure drop of the gas between two nodes is independent of the path taken by the gas.

To model the entire network, we use the incidence matrix \mathcal{A}^* of the graph \vec{G} . This matrix is defined as

$$\mathcal{A}^* = (a_{i,j}) \in \mathbb{R}^{|V^*| \times |E|}, \quad a_{i,j} = \begin{cases} +1 & \text{if head}(a_j) = u_i, \\ -1 & \text{if tail}(a_j) = u_i, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 2.4. The rank of \mathcal{A}^* is $|V^*| - 1$.

Proof. Consider the $|V^*| - 1$ columns representing the arcs of the tree \mathcal{T} . For any other column associated to an arc \vec{a} , we know this arc will be part of an (unoriented) cycle. Therefore, if $\vec{a} = (u, v)$, it is possible to find a path joining u and v by picking the smallest values of n and m such that $p^n(u) = p^m(v) = w$, and joining the path $w \rightarrow v$ with the inverted path $u \rightarrow w$. This allows us to write the column associated to \vec{a} as the sum of the columns in the $w \rightarrow v$ path minus the columns in the $w \rightarrow u$, giving that the arcs of \mathcal{T} are a generating set, and hence $\text{rank}(\mathcal{A}^*) \leq n - 1$.

What remains is showing that this set is independent, which is clear: let $\mathcal{A}_1, \dots, \mathcal{A}_r$ be the columns corresponding to the arcs of \mathcal{T} , and assume

$$0 = \sum_{i=1}^r \lambda_i \mathcal{A}_i.$$

That relation in particular has to hold component-wise. If v_j is a leaf in the tree (and there must be at least one leaf in any tree), and \mathcal{A}_k is the column corresponding to the only arc going to that leaf, we obtain by limiting the equation above to row j the equation

$$\sum_{i=1}^r \lambda_i \mathcal{A}_{j,i} = \lambda_k \mathcal{A}_{j,k} = 0 \implies \lambda_k = 0.$$

Assuming without loss of generality that $k = r$, we arrive to

$$0 = \sum_{i=1}^{r-1} \lambda_i \mathcal{A}_i,$$

and repeating the process for new leaves of the tree after deleting v_j we obtain

$$\lambda_1 = \lambda_2 = \dots = \lambda_r = 0,$$

therefore proving the independence. \square

Since the rank of \mathcal{A}^* is $|V^*| - 1$, we define a new matrix \mathcal{A} to be equal to \mathcal{A}^* but with the row related to r deleted. If $V = V^* - \{r\}$, then $\text{rank}(\mathcal{A}) = |V|$. We arrange the matrix as

$$\mathcal{A} = [\mathcal{A}_B \mid \mathcal{A}_N],$$

with the subindex B (standing for *basis*) meaning that we take the arcs stemming from the edges of \mathcal{T} . Since it was a tree on a graph with $|V^*|$ vertices, there are $|V|$ such edges and hence $\mathcal{A}_B \in \mathbb{R}^{|V| \times |V|}$.

Corollary 2.5. The matrix \mathcal{A}_B is nonsingular.

Proof. Simply note that in lemma 2.4 we already showed that the columns of \mathcal{A}_B form a basis of the space that \mathcal{A}^* spans, and the arguments hold the same for \mathcal{A} . \square

Keeping with this convention, we define p and q^{nom} from its starred siblings by removing the value corresponding to the root node, either p_r or q_r^{nom} . We also define $\phi \in \mathbb{R}^{|A|}$ to be the vector of pressure drops at every arc, and $\Phi = \text{diag}(\phi)$.

If we look at the construction of the incidence matrix and eq. (2.5) it is clear that

$$\mathcal{A}^*Q_0 = q^{\text{nom}*}.$$

This system is overdetermined since \mathcal{A}^* does not have full rank, therefore it is equivalent to the system

$$\mathcal{A}Q_0 = q^{\text{nom}} \tag{2.6}$$

since we can lift any solution due to eq. (2.1).

We have similarly that eq. (2.4) leads to

$$(\mathcal{A}^*)^\top (p^*)^2 = -\Phi|Q_0|Q_0. \tag{2.7}$$

Here, $(p^*)^2$ is the vector of the squared pressures, and similarly for $|Q_0|Q_0$, meaning that we perform the operations element-wise. If we consider the system

$$\begin{aligned} \mathcal{A}Q_0 &= q^{\text{nom}}, \\ (\mathcal{A}^*)^\top (p^*)^2 &= -\Phi|Q_0|Q_0, \end{aligned}$$

we have $|V| + |E|$ equations and $|V^*| + |E|$ variables. This difference is because we have removed the equation for q_r^{nom} , that could be deduced from the others. To solve the issue of the system being underdefined, we also set the pressure p_r at the root. Then the previous equation is equivalent to the equations

$$\mathcal{A}_{r,B}^\top p_r^2 + \mathcal{A}_B^\top p^2 = -\Phi_B|Q_{0,B}|Q_{0,B} \tag{2.8}$$

and

$$\mathcal{A}_{r,N}^\top p_r^2 + \mathcal{A}_N^\top p^2 = -\Phi_N|Q_{0,N}|Q_{0,N} \tag{2.9}$$

by considering for instance that r is the first node and hence the first row \mathcal{A}_r is removed from \mathcal{A}^* to get \mathcal{A} , giving

$$\mathcal{A}^* = \begin{bmatrix} \mathcal{A}_r \\ \mathcal{A}_B \ \mathcal{A}_N \end{bmatrix} \implies (\mathcal{A}^*)^\top = \begin{bmatrix} \mathcal{A}_{r,B}^\top & \mathcal{A}_B^\top \\ \mathcal{A}_{r,N}^\top & \mathcal{A}_N^\top \end{bmatrix},$$

and performing the block-matrix multiplication.

Rearranging eq. (2.8) then leads to

$$p^2 = -(\mathcal{A}_B^\top)^{-1} \mathcal{A}_{r,B}^\top p_r^2 - (\mathcal{A}_B^\top)^{-1} \Phi_B |Q_{0,B} | Q_{0,B}.$$

Since \mathcal{A}^* is an incidence matrix, there is only one +1 and one -1 in each column, hence

$$\mathbb{1}^\top \mathcal{A}^* = 0 \implies \mathbb{1}^\top \begin{bmatrix} \mathcal{A}_{r,B} \\ \mathcal{A}_B \end{bmatrix} = 0 \implies \mathcal{A}_{r,B} = -\mathbb{1}^\top \mathcal{A}_B, \quad \mathcal{A}_{r,N} = -\mathbb{1}^\top \mathcal{A}_N.$$

We therefore get the simplified equation

$$p^2 = \mathbb{1} p_r^2 - (\mathcal{A}_B^\top)^{-1} \Phi_B |Q_{0,B} | Q_{0,B}. \quad (2.10)$$

If we substitute back into eq. (2.9) we obtain

$$\mathcal{A}_{r,N}^\top p_r^2 + \mathcal{A}_N^\top \mathbb{1}^\top p_r^2 - \mathcal{A}_N^\top (\mathcal{A}_B^\top)^{-1} \Phi_B |Q_{0,B} | Q_{0,B} = -\Phi_N |Q_{0,N} | Q_{0,N}.$$

The first two terms cancel each other by rewriting $\mathcal{A}_{r,N}^\top$, and we are left with

$$\mathcal{A}_N^\top (\mathcal{A}_B^\top)^{-1} \Phi_B |Q_{0,B} | Q_{0,B} = \Phi_N |Q_{0,N} | Q_{0,N}. \quad (2.11)$$

Going back to eq. (2.6), we have

$$\mathcal{A} Q_0 = q^{\text{nom}} \implies \mathcal{A}_B Q_{0,B} + \mathcal{A}_N Q_{0,N} = q^{\text{nom}} \implies Q_{0,B} = \mathcal{A}_B^{-1} (q^{\text{nom}} - \mathcal{A}_N Q_{0,N}).$$

Note that \mathcal{A}_B is invertible due to corollary 2.5.

If we now define

$$g : \mathbb{R}^{|V|} \times \mathbb{R}^{|A|-|\mathcal{J}|} \longrightarrow \mathbb{R}^{|V|}, \\ (s, t) \longmapsto (\mathcal{A}_B^{-1})^\top \Phi_B | \mathcal{A}_B^{-1} (s - \mathcal{A}_N t) | \mathcal{A}_B^{-1} (s - \mathcal{A}_N t),$$

we get that the eq. (2.11) is equivalent to

$$\mathcal{A}_N^\top g(q^{\text{nom}}, Q_{0,N}) = \Phi_N |Q_{0,N} | Q_{0,N}.$$

We can therefore state the following theorem.

Theorem 2.6. For a gas network with load vector q^{nom} , let

$$F(z) = \mathcal{A}_N^\top g(q^{\text{nom}}, z) - \Phi_N |z | z.$$

If \hat{z} is root of F , then the vector Q_0 with

$$Q_{0,N} = \hat{z}, \\ Q_{0,B} = \mathcal{A}_B^{-1} (q^{\text{nom}} - \mathcal{A}_N \hat{z}).$$

is the vector of flows of the network.

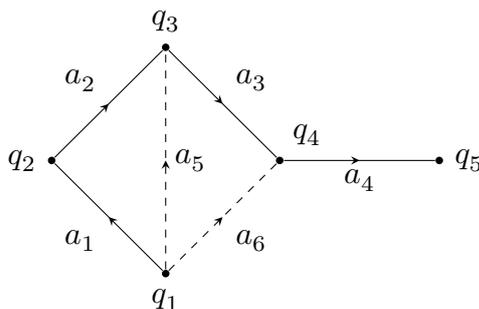
The proof of the theorem is immediate from the previous discussion.

Remark 2.7. The equation $F(z) = 0$ has a unique solution over the real numbers; a proof can be found in Ríos Mercado et al. 2002, § 4.

Remark 2.8. From the construction of \mathcal{A} one gets that every column of \mathcal{A}_N corresponds to an arc added to the tree, and therefore an arc that creates an undirected cycle in the graph G . We call any such cycle a fundamental cycle. Therefore from the previous theorem we obtain that in order to get a solution for the entire network it is enough to look at the fundamental cycles, and then finding the rest of the flows can be done by simple linear algebra.

Note that, while a solution for $F(z)$ leads to a set of flows and pressures that satisfy eqs. (2.2) and (2.3), this is not enough to guarantee that it's an actual solution for the network. Indeed, the network will always have some additional constraints on the flow through the pipes and the pressure at nodes associated with the physical capabilities of the pipes and the junctions. In particular, one should expect that the pipes have a maximum volumetric flow rate, and that the pressure at nodes is restricted to a certain interval for safety reasons. However, we will not mention these conditions, and instead focus in finding the solution of $F(z)$.

Example 2.9. Let us consider a simple network given by the graph



The tree \mathcal{T} is formed by the vertices q_1, q_2, q_3, q_4, q_5 visited in that order, therefore the two fundamental cycles are formed by a_5 and a_6 , that will become z_1 and z_2 respectively. The arcs in the graph follow the orientation induced by \mathcal{T} . The matrix \mathcal{A} will be given by

$$\mathcal{A} = \begin{pmatrix} +1 & -1 & 0 & 0 & 0 & 0 \\ 0 & +1 & -1 & 0 & +1 & 0 \\ 0 & 0 & +1 & -1 & 0 & +1 \\ 0 & 0 & 0 & +1 & 0 & 0 \end{pmatrix}.$$

The right hand side of the system of equations will simply consist of $\phi_5|z_1|z_1$ and $\phi_6|z_2|z_2$. For the right hand side, note that

$$q - \mathcal{A}_N z = \begin{pmatrix} q_2 \\ q_3 - z_1 \\ q_4 - z_2 \\ q_5 \end{pmatrix} \implies \mathcal{A}_B^{-1}(q - \mathcal{A}_N z) = \begin{pmatrix} q_2 + q_3 + q_4 + q_5 - z_1 - z_2 \\ q_3 + q_4 + q_5 - z_1 - z_2 \\ q_4 + q_5 - z_2 \\ q_5 \end{pmatrix}.$$

We also have that

$$\mathcal{A}_N^\top (\mathcal{A}_B^{-1})^\top \Phi_B = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix} = \begin{pmatrix} \phi_1 & \phi_2 & 0 & 0 \\ \phi_1 & \phi_2 & \phi_3 & 0 \end{pmatrix}.$$

This leads to the overall system

$$\begin{aligned} \phi_5 |z_1| z_1 &= \phi_1 |q_2 + q_3 + q_4 + q_5 - z_1 - z_2| (q_2 + q_3 + q_4 + q_5 - z_1 - z_2) \\ &\quad + \phi_2 |q_3 + q_4 + q_5 - z_1 - z_2| (q_3 + q_4 + q_5 - z_1 - z_2), \\ \phi_6 |z_2| z_2 &= \phi_1 |q_2 + q_3 + q_4 + q_5 - z_1 - z_2| (q_2 + q_3 + q_4 + q_5 - z_1 - z_2) \\ &\quad + \phi_2 |q_3 + q_4 + q_5 - z_1 - z_2| (q_3 + q_4 + q_5 - z_1 - z_2) \\ &\quad + \phi_3 |q_4 + q_5 - z_2| (q_4 + q_5 - z_2). \end{aligned}$$

Note that this system is independent of ϕ_4 , as expected since it is not part of any cycle, and moreover q_5 appears only with q_4 and not alone. In the construction of $\mathcal{A}_B^{-1}(q - \mathcal{A}_N z)$, which gives the flows $Q_{0,N}$, the last component is simply the load at q_5 , as that will always be the flow independently of the values in the rest of the network. The elements of $Q_{0,N}$ can also be described simply: for instance, for Q_{0,a_3} representing the flow through a_3 , one sees that this flow equals the loads at q_4 and q_5 that the pipe has to supply, minus the flow through z_2 that is also supplying those nodes. So the formula simply reads $Q_{0,a_3} + Q_{0,a_6} = q_4 + q_5 = q_4 + Q_{0,a_4}$, equivalent to

$$q_4 = Q_{0,a_3} + Q_{0,a_6} - Q_{0,a_4}.$$

This is simply a reformulation of eq. (2.5).

Remark 2.10. It is possible to arrive to different models for the gas flow, depending on the treatment of the system in eqs. (2.2) and (2.3). For instance, in Ekhtiari et al. 2019, they work with nodes at different heights (meaning that $s \neq 0$), and arrive to

$$p_u^2 - p_v^2 = \phi_a |Q_{0,a}| Q_{0,a} + \psi_a (p_u + p_v)^2.$$

The issue is the fact that one gets a cross product $p_u p_v$ when squaring out the sum, which means that one cannot have an equation linear in p^2 . In Pfetsch et al. 2015, some pipes are considered to be compressors. In those cases, one lets $\phi_a = 0$ and defines $\Delta_a = p_v^2 - p_u^2$. Then one gets

$$\mathcal{A}_N^\top (\mathcal{A}_B^\top)^{-1} (\Phi_B | \mathcal{A}_B^{-1}(q - \mathcal{A}_N z) | (\mathcal{A}_B^{-1}(q - \mathcal{A}_N z)) - \Delta_B) = \Phi_N |z| z - \Delta_N.$$

But then the choice of Δ is unclear: one could also solve for Δ by considering the equations of a compressor station, but the system also becomes non-polynomial.

2.4 Obtaining a polynomial system

We are tasked with solving a system of equations of the form

$$\mathcal{A}_N^\top (\mathcal{A}_B^{-1})^\top \Phi_B |\mathcal{A}_B^{-1}(q^{\text{nom}} - \mathcal{A}_N z)| (\mathcal{A}_B^{-1}(q^{\text{nom}} - \mathcal{A}_N z)) = \Phi_N |z|z.$$

The problem is that our idea is solving the equation using polynomial solvers, therefore we have to remove the absolute values.

To do so, we consider a vector $s \in \{+1, -1\}^{|A|}$ that represents the sign of the associated flow, so that

$$s_a Q_{0,a}^2 = |Q_{0,a}| Q_{0,a}.$$

If we divide s as (s_B, s_N) as before, it is clear that $s_N z^2 = |z|z$ (where again the operation is considered component-wise). Moreover, note that

$$Q_{0,B} = \mathcal{A}_B^{-1}(q^{\text{nom}} - \mathcal{A}_N z),$$

therefore we can remove the absolute value in the left hand side by multiplying with s_B . We then get

$$\mathcal{A}_N^\top (\mathcal{A}_B^{-1})^\top \Phi_B \text{diag}(s_B) (\mathcal{A}_B^{-1}(q^{\text{nom}} - \mathcal{A}_N z))^2 = \Phi_N \text{diag}(s_N) z^2. \quad (2.12)$$

The issue is that eq. (2.12) has, in general, more solutions than the system we want to solve. The reason for this is that, although if we set $(s_N)_i = 1$ it means we are solving for $z_i \geq 0$, it is entirely possible that a solver will find a solution where $z_i < 0$. To get around this problem, one can simply check that, for a solution \hat{z} , the signs are consistent, meaning that

$$\text{diag}(s_N) \hat{z} \geq 0, \quad \text{diag}(s_B) \mathcal{A}_B^{-1}(q^{\text{nom}} - \mathcal{A}_N \hat{z}) \geq 0.$$

Although many solutions are possible in eq. (2.12), only one of them will satisfy the inequalities above, since that solution will also be a valid solution in the system defined in theorem 2.6. And as that system can only have one equation, so will ours.

Example 2.11. Repeating the previous example, one would get the system

$$\begin{aligned} \phi_5 s_5 z_1^2 &= \phi_1 s_1 (q_2 + q_3 + q_4 + q_5 - z_1 - z_2)^2 + \phi_2 s_2 (q_3 + q_4 + q_5 - z_1 - z_2)^2, \\ \phi_6 s_6 z_2^2 &= \phi_1 s_1 (q_2 + q_3 + q_4 + q_5 - z_1 - z_2)^2 + \phi_2 s_2 (q_3 + q_4 + q_5 - z_1 - z_2)^2 \\ &\quad + \phi_3 s_3 (q_4 + q_5 - z_2)^2, \end{aligned}$$

alongside the restrictions $s_5 z_1 \geq 0, s_6 z_2 \geq 0, s_4 q_5 \geq 0$ and

$$\begin{aligned} s_1 (q_2 + q_3 + q_4 + q_5 - z_1 - z_2) &\geq 0, \\ s_2 (q_3 + q_4 + q_5 - z_1 - z_2) &\geq 0, \\ s_3 (q_4 + q_5 - z_2) &\geq 0. \end{aligned}$$

By choosing values for s , the system becomes polynomial, and the restrictions can be checked in order to find if the solutions are valid. Again, once the load q_5 is fixed, the direction s_4 is fixed.

3 Gröbner basis

From the previous chapter we know that computing a solution (meaning a flow distribution through the network) for a given gas network boils down to finding a solution of a set of quadratic equations given by eq. (2.12).

To find such solutions we will use Gröbner bases, which are one of the most important tools in computational algebra. The main idea is that a Gröbner basis, with respect to a given ordering, will give an equivalent but more useful representation of the system that allows us to find global solutions by considering equations with a smaller number of variables first.

This of course comes at a price: although the description and the algorithms for computing Gröbner bases are straightforward, they are provably EXPSPACE (so they require both exponential time and memory, see Bardet 2002), which means they scale badly when the size of the network increases.

3.1 The ideal membership problem

We will now broadly follow Cox, Little, and O’Shea 2015 in the presentation of the theory of Gröbner bases. Let us start by recalling some definitions.

Definition 3.1 (*Rings*). A ring $(R, +, \cdot)$ is an Abelian group $(R, +)$ with an additional associative binary operation $\cdot : R \times R \rightarrow R$ with identity 1 such that $1 \cdot a = a \cdot 1 = a$, and where \cdot distributes over $+$ as

$$a \cdot (b + c) = a \cdot b + a \cdot c, \quad (a + b) \cdot c = a \cdot c + b \cdot c.$$

Definition 3.2 (*Fields*). A field K is a ring where the operation \times has an inverse.

We will consider three fields at the same time. First, we would like to be working over \mathbb{R} , since the values of loads, flows and pressures will be real numbers. However, as we are working with computers, it’s more practical to work over \mathbb{Q} , so that if we have $r \in \mathbb{R}$ written as

$$r = z \times 10^e$$

with z and e both integers (due to the limitations of working with a computer with limited precision) we can consider instead

$$q = \frac{z}{10^{-e}} \in \mathbb{Q}.$$

3 Gröbner basis

Finally, since for many questions of algebraic geometry it is more convenient to work over an algebraically closed field, we will sometimes consider our problem to be in \mathbb{C} .

The ring of choice for us will be the ring of polynomials in n variables, denoted by

$$R = K[x_1, \dots, x_n].$$

Definition 3.3 (*Ideals*). An ideal $I \subseteq R$ of a ring R is a subset that is itself a ring, and such that $R \times I \subseteq I$. Given elements r_1, r_2, \dots, r_k of the ring, we can define the ideal they generate as

$$I = \langle r_1, \dots, r_k \rangle = \left\{ \sum_{i=1}^k a_i r_i \mid a_i \in R \right\}$$

Now we want to consider ideals I in $R = K[x_1, \dots, x_n]$. We can in general write them as

$$I = \langle f_1, \dots, f_k \rangle$$

where the f_i are polynomials in n variables. This is a consequence of the following theorem.

Theorem 3.4 (*Hilbert's basis theorem*). Any ideal I of $R = K[x_1, \dots, x_n]$ can be written as

$$I = \langle f_1, \dots, f_k \rangle$$

for some $f_i \in R$.

Note that in general (for other rings) this is not true, since some ideals may have infinitely many generators. For rings of polynomials over a field, however, every ideal has a finite set of generators.

Despite the simplicity of the description of I , some questions are really hard to answer. A clear example is, given I and J two ideals over the same field, to find whether $I = J$. Other is the one we will focus on: given an element f and an ideal I of a ring R , to establish whether $f \in I$. The theory of Gröbner basis can provide a solution to both of these problems.

To get a picture of the path that we are going to follow it is interesting to consider a one-dimensional case, so work with ideals in $\mathbb{Q}[x]$. Since this is a principal ideal domain, all ideals are of the form

$$I = \langle f(x) \rangle \subset \mathbb{Q}[x].$$

If we're given a set of generators f_1, \dots, f_k of the ideal, then

$$\langle f_1, \dots, f_k \rangle = \langle \gcd(f_1, \dots, f_k) \rangle.$$

Bézout's identity implies that the greatest common divisor $d(x)$ is a member of the ideal, since it gives

$$d(x) = \sum_{i=1}^k \alpha_i(x) f_i(x).$$

In the other direction, if $f_i(x) = d(x)g_i(x)$, then

$$\sum_{i=1}^k a_i(x)f_i(x) = d(x) \sum_{i=1}^k a_i(x)g_i(x).$$

Then, in order to test whether $f(x)$ is in the ideal generated by the f_i , it is enough to check whether $d(x)$ divides $f(x)$. Our goal is to find a similar characterization for the multivariate case.

3.1.1 Monomial ordering

We want to establish some analogue of the Euclidean division of univariate polynomials. However, in the one dimensional case it is easy to compare polynomials by looking at their degree: when more monomials are involved, this becomes more complicated. We therefore have to define monomial orderings. This amounts to establishing an order in \mathbb{N}^n , since we can write

$$x_1^{a_1}x_2^{a_2}\cdots x_n^{a_n} = x^\alpha, \quad \alpha = (a_1, \dots, a_n) \in \mathbb{N}^n.$$

and then compare monomials by looking at α (and forgetting about the coefficients c_α). Then

$$\alpha > \beta \text{ in } \mathbb{N}^n \iff x^\alpha > x^\beta \text{ in } K[x_1, \dots, x_n].$$

Of course, our ordering α must have some sensible properties: it has to be a total ordering (meaning that any two monomials can be compared) and it has to be transitive. Moreover, we must have

$$x^\alpha > x^\beta \implies x^\alpha x^\gamma > x^\beta x^\gamma.$$

It also has to be a well-ordering (so every set has a smallest element).

Definition 3.5 (*Lexicographic ordering*). Given α and β in \mathbb{N}^n , we say $\alpha >_{\text{lex}} \beta$ if the first component of $\alpha - \beta$ is positive.

It is easy to show that this is indeed an ordering: the only thing one has to check is

$$x^\alpha >_{\text{lex}} x^\beta \implies x^\alpha x^\gamma >_{\text{lex}} x^\beta x^\gamma.$$

But indeed,

$$\begin{aligned} x^\alpha >_{\text{lex}} x^\beta &\implies \alpha - \beta > 0 \\ &\implies (\alpha + \gamma) - (\beta + \gamma) > 0 \\ &\implies x^\alpha x^\gamma >_{\text{lex}} x^\beta x^\gamma, \end{aligned}$$

since the product of monomials is equivalent to the sum of their degrees.

The lexicographic ordering works by always giving more weight to having more higher degrees of the initial variables. So, in $K[x_1, x_2, x_3]$,

$$x_1^2 >_{\text{lex}} x_1 >_{\text{lex}} x_2^n x_3^m.$$

There are of course other choices of ordering.

3 Gröbner basis

- The *graded lexicographic ordering* first looks at the total degree of the monomials and orders them accordingly, and then resolves ties by looking at the lexicographic ordering of monomials with equal total degree, so

$$x_1^2 x_2 >_{\text{grlex}} x_1 x_2^2 >_{\text{grlex}} x_1^2 >_{\text{grlex}} x_2^2.$$

- The *graded reverse lexicographic order* takes a different approach. The first thing to do is ranking monomials by their degree. Then, to compare them, we look at the last variable x_n and take as higher the one with lowest exponent. In case of a tie, we look at x_{n-1} , and so on. This means that $x_1 x_2^2 x_3 >_{\text{grevlex}} x_1^2 x_3^2$. For example, this gives

$$x_1^2 >_{\text{grevlex}} x_1 x_2 >_{\text{grevlex}} x_2^2 >_{\text{grevlex}} x_1 x_3 >_{\text{grevlex}} x_2 x_3 >_{\text{grevlex}} x_3^2,$$

the main difference being that $x_2^2 >_{\text{grevlex}} x_1 x_3$ despite the first term not having any x_1 term.

We will however use mostly the lexicographic ordering.

Remark 3.6. The choice of ordering is actually crucial for the computation of a Gröbner basis, and some orderings (in particular $>_{\text{grevlex}}$) are much faster than the others. In general one could say that the lexicographic ordering is the worse for computation, since it is the one that gives more information about the inner structure of the ideal. Unfortunately, it is the one we need to use.

Once we have a choice of ordering, we can define some analogues of the univariate polynomials.

Definition 3.7. Let $f = \sum a_\alpha x^\alpha$ be a polynomial, and choose some monomial order $>$. We define:

1. The multidegree of f as

$$\text{multideg}(f) = \max_{>} \{ \alpha \in \mathbb{N}^n \mid a_\alpha \neq 0 \}.$$

2. The leading coefficient of f as

$$\text{LC}(f) = a_{\text{multideg}(f)}.$$

3. The leading monomial of f as

$$\text{LM}(f) = x^{\text{multideg}(f)}.$$

4. The leading term of f as

$$\text{LT}(f) = \text{LC}(f) \text{LM}(f).$$

We have the following properties for multideg.

Lemma 3.8. For $f, g \in K[x_1, \dots, x_n]$, we have

1. $\text{multideg}(fg) = \text{multideg}(f) + \text{multideg}(g)$.

$$2. \text{multideg}(f + g) \leq \max(\text{multideg}(f), \text{multideg}(g)).$$

Proof. First note that

$$fg = \left(\sum_{\alpha} a_{\alpha} x^{\alpha} \right) \left(\sum_{\beta} b_{\beta} x^{\beta} \right) = \sum_{\alpha, \beta} a_{\alpha} b_{\beta} x^{\alpha + \beta}.$$

Then one clearly has

$$\text{multideg}(fg) \geq \text{multideg}(f) + \text{multideg}(g)$$

by taking the term $\text{LT}(f)\text{LT}(g)$. Moreover, for any monomial, one has

$$\text{multideg}(x^{\alpha + \beta}) = \alpha + \beta = \text{multideg}(x^{\alpha}) + \text{multideg}(x^{\beta}),$$

meaning that, for any monomial of the product, its multidegree will be bounded by the sum of the multidegrees of the original polynomials, giving the opposite inequality. For the inequality with the sum, we write

$$f + g = \sum_{\gamma} (a_{\gamma} + b_{\gamma}) x^{\gamma}.$$

Then if $a_{\gamma} + b_{\gamma} \neq 0$ it means either of the two terms is nonzero, so assuming it is a_{γ} without loss of generality gives

$$\text{multideg}(f + g) = \gamma \implies \text{multideg}(f) \geq \gamma,$$

and adding the inequality for $b_{\gamma} \neq 0$ gives the result. \square

Note that the inequality for the sum arises from the fact that the leading terms in f and g might cancel each other.

Remark 3.9. In particular, lemma 3.8 implies that multideg is a non-Archimedean norm in $K[x_1, \dots, x_n]$.

3.1.2 Division algorithms

As mentioned before, the key idea for the one-dimensional case rests on the idea of divisibility and the greatest common divisor. In one dimension it is easy to simply use Euclid's algorithm: by the recursion

$$\text{gcd}(f_1, \dots, f_k) = \text{gcd}(f_1, \text{gcd}(f_2, \dots, f_k))$$

it suffices to calculate the gcd of two polynomials, which is easy to do in a way completely analogous to the case of integers. For the case of many variables, there is an analogue to the division algorithm.

Theorem 3.10. Given a monomial order $>$ in $K[x_1, \dots, x_n]$, and a set f_1, \dots, f_s of polynomials, one can write every other polynomial f as

$$f = q_1 f_1 + \dots + q_s f_s + r$$

where the q_i and r are all polynomials and r is either 0 or all its monomials are not divisible by $\text{LT}(f_1), \dots, \text{LT}(f_s)$. Then r is the remainder of the division, and

$$\text{multideg}(f) \geq \text{multideg}(q_i f_i).$$

Just like in the Euclidean case, the proof here is actually constructive: one can give an algorithm to calculate the q_i and r . Note that r is not uniquely determined.

Proof. We have the procedure described in algorithm 1.

Algorithm 1 Division algorithm in $K[x_1, \dots, x_n]$

Require: $f \in k[x_1, \dots, x_n]$
Require: $\{f_1, \dots, f_s\} \subset k[x_1, \dots, x_n]$

- 1: $q_1, \dots, q_s, r \leftarrow 0$
- 2: $p \leftarrow f$
- 3: **while** $p \neq 0$ **do**
- 4: $i \leftarrow 1$
- 5: $\text{division} \leftarrow \text{false}$
- 6: **while** $i \leq s$ **and** $\text{division} = \text{false}$ **do**
- 7: **if** $\text{LT}(f_i) \mid \text{LT}(p)$ **then**
- 8: $q_i \leftarrow q_i + \text{LT}(p) / \text{LT}(f_i)$
- 9: $p \leftarrow p - (\text{LT}(p) / \text{LT}(f_i)) f_i$
- 10: $\text{division} \leftarrow \text{true}$
- 11: **else**
- 12: $i \leftarrow i + 1$
- 13: **if** $\text{division} = \text{false}$ **then**
- 14: $r \leftarrow r + \text{LT}(p)$
- 15: $p \leftarrow p - \text{LT}(p)$

First, note that at every step one has

$$f = q_1 f_1 + \dots + q_s f_s + p + r.$$

In the beginning this is because $p = f$ and the rest are zeroes. After every step, either we add and subtract $f_i(\text{LT}(p) / \text{LT}(f_i))$, if the division occurs, or we add and subtract $\text{LT}(p)$ otherwise.

Note that the WHILE loop goes on until p vanishes, which does in finitely many steps. Indeed, every pass of the loop takes care of the leading term of p , and either adds it to a

quotient q_i or sends it to the remainder r . This is clear in the second case; in the first note that we are doing

$$p' = p - \frac{\text{LT}(p)}{\text{LT}(f_i)} f_i$$

and we have

$$\text{LT}\left(\frac{\text{LT}(p)}{\text{LT}(f_i)} f_i\right) = \text{LT}(p)$$

This means that the algorithm terminates, and the number of steps is the number of monomials of the input f . Hence, when $p = 0$ is reached, the algorithm breaks and

$$f = q_1 f_1 + \cdots + q_s f_s + r$$

holds.

Now, clearly every monomial in r is not divisible by any of the $\text{LT}(f_i)$, otherwise it would have been added to q_i in the step before. Finally, note that

$$\begin{aligned} \text{multideg}(q_i f_i) &= \text{multideg}(q_i) + \text{multideg}(f_i) \\ &= \text{multideg}(\text{LT}(p') / \text{LT}(f_i)) + \text{multideg}(f_i) \\ &= \text{multideg}(p') \end{aligned}$$

where p' is one of the temporary values of p . And, since we are always removing the leading terms of p ,

$$\text{multideg}(f) = \text{multideg}(p) \geq \text{multideg}(p') = \text{multideg}(q_i f_i). \quad \square$$

This may give hope for an answer of the ideal membership problem, in a similar way to what happens in the one-dimensional case: given f and $I = \langle f_1, \dots, f_s \rangle$, perform the division and, if $r = 0$, $f \in I$. While this holds, it is also possible to have elements of the ideal that leave nonzero remainder, so a more precise analysis is needed.

3.1.3 Leading terms and ideal membership

Let us start working towards our goal: solving the ideal membership problem.

Definition 3.11. Let I be an ideal and $>$ an ordering.

1. We define

$$\text{LT}(I) = \{ cx^\alpha \mid \text{there is some } f \in I - \{0\} \text{ with } \text{LT}(f) = cx^\alpha \},$$

the set of leading terms of I .

2. We define $\langle \text{LT}(I) \rangle$ to be the ideal generated by the previous set.

Recall that the division algorithm is influenced essentially by the leading terms of the polynomials (as in the conditions for the remainder). Note that

$$\langle \text{LT}(f_1), \dots, \text{LT}(f_k) \rangle \subseteq \langle \text{LT}(\langle f_1, \dots, f_k \rangle) \rangle,$$

but in general we do not have equality.

We have the following interesting result.

Proposition 3.12. For an ideal I , there are $g_1, \dots, g_t \in I$ such that

$$\langle \text{LT}(I) \rangle = \langle \text{LT}(g_1), \dots, \text{LT}(g_t) \rangle.$$

Moreover, $I = \langle g_1, \dots, g_t \rangle$.

The proof relies on Dickson's Lemma, which is quite technical, so we omit it. This proposition gives us exactly what we need.

Definition 3.13. A subset $G = \{g_1, \dots, g_t\}$ of an ideal $I \subset K[x_1, \dots, x_n]$ is called a Gröbner basis if

$$\langle \text{LT}(I) \rangle = \langle \text{LT}(g_1), \dots, \text{LT}(g_t) \rangle.$$

By the previous proposition, every ideal has a Gröbner basis.

Remark 3.14. Despite their name, Gröbner bases were first defined by Bruno Buchberger in his Ph.D. thesis, and were named after his advisor Wolfgang Gröbner. Note that they appeared previously in the work of the Russian mathematician Nikolai Maximovich Gjunter, but went unnoticed – see also Renshukh, Roloff, and Rasputin 1987.

Why is this what we want? Take an element $f \in K[x_1, \dots, x_n]$ and divide it by G to get

$$f = q_1g_1 + q_2g_2 + \dots + q_tg_t + r.$$

We already know that $r = 0$ implies that $f \in I$. On the other direction, assume $f \in I$. Then

$$r = f - q_1g_1 - \dots - q_tg_t \in I,$$

since ideals are closed under addition. But then, if $r \neq 0$, we have that

$$\text{LT}(r) \in \langle \text{LT}(I) \rangle = \langle \text{LT}(g_1), \dots, \text{LT}(g_t) \rangle,$$

so that $\text{LT}(r)$ is divisible by some $\text{LT}(g_i)$. However, this contradicts our division algorithm, hence $r = 0$. This means that

$$f \in I \iff r = 0,$$

which solves in a simple way the ideal membership problem.

While not explicit, note that the choice of an ordering is present throughout all the steps since it is required to define the leading term.

Overall, this means that the problem of ideal membership can be solved by computing a Gröbner basis of the ideal.

3.1.4 Elimination theory

Let us now take a detour towards algebraic geometry. Given an ideal I , we define

$$V(I) = \{ (a_1, \dots, a_n) \in K^n \mid f(a_1, \dots, a_n) = 0 \forall f \in I \} \subset K^n.$$

3.1 The ideal membership problem

This is called the *variety* generated by I , and it is a fundamental tool. If $I = \langle f_1, \dots, f_k \rangle$, to describe the variety it is enough to check for roots of the system

$$\begin{cases} f_1 = 0, \\ f_2 = 0, \\ \vdots \\ f_k = 0. \end{cases}$$

Clearly all elements of $V(I)$ are solutions to this system, as the f_i are in particular a set of elements of I . And on the other direction, if a is a solution of the system, then for $f \in I$ one has

$$f = \sum_{i=1}^k q_i f_i \implies f(a) = \sum_{i=1}^k q_i(a) f_i(a) = \sum_{i=1}^k q_i(a) \cdot 0 = 0$$

This means that solving the system of equations is equivalent to describing the associated variety that the ideal generates. To see why this matters, let us look at an example.

Example 3.15. Consider the system of equations

$$\begin{aligned} x_1^2 + x_2 + x_3 &= 1, \\ x_1 + x_2^2 + x_3 &= 1, \\ x_1 + x_2 + x_3^2 &= 1. \end{aligned}$$

To the system we associate the ideal

$$I = \langle x_1^2 + x_2 + x_3 - 1, x_1 + x_2^2 + x_3 - 1, x_1 + x_2 + x_3^2 - 1 \rangle,$$

and compute its Gröbner basis with respect to $>_{\text{lex}}$ to get

$$\begin{aligned} g_1 &= x_1 + x_2 + x_3^2 - 1, \\ g_2 &= x_2^2 - x_2 - x_3^2 + x_3, \\ g_3 &= 2x_2x_3^2 + x_3^4 - x_3^2, \\ g_4 &= x_3^6 - 4x_3^4 + 4x_3^3 - x_3^2. \end{aligned}$$

Now note that any element of $V(I)$, which is any solution of the original system, will also be a solution of the system

$$g_1 = 0, g_2 = 0, g_3 = 0, g_4 = 0$$

since the Gröbner basis also generates the ideal. But now note that the fourth equation is actually an univariate polynomial,

$$g_4 = x_3^2(x_3 - 1)^2(x_3^2 + 2x_3 - 1) = 0$$

3 Gröbner basis

and can be easily solved (in this case factorized, in others, by using some one-dimensional polynomial solver). Then we can substitute the possible solutions $a_3 \in K$ back to g_3 to find possible values a_2 of x_2 , and work upwards to find global solutions

$$(a_1, a_2, a_3) \in V(I) \subset K^3.$$

This means that the original system is reduced to finding solutions of univariate polynomials.

The goal now is showing that this behaviour is not just luck. To do so we need the elimination and extension theorems.

Definition 3.16. For an ideal $I = \langle f_1, \dots, f_k \rangle$, we define the ℓ -th elimination ideal to be

$$I_\ell = I \cap K[x_{\ell+1}, \dots, x_n].$$

Under this definition, the elimination ideal I_ℓ consists of all those polynomials in the ideal that are independent of the first ℓ variables. For the previous example, one would have $I_1 = \langle g_2, g_3, g_4 \rangle$ and $I_2 = \langle g_4 \rangle$.

Theorem 3.17 (Elimination Theorem). Let I be an ideal and G a Gröbner basis with respect to $>_{\text{lex}}$. Then

$$G_\ell = G \cap K[x_{\ell+1}, \dots, x_n]$$

is a Gröbner basis of I_ℓ .

This theorem tells us that a (Gröbner) basis of the ℓ -th elimination ideal can be found by simply taking the elements of G that don't contain the first ℓ variables. Note that using $>_{\text{lex}}$ is not really fundamental here: all we need is an order that satisfies $x_1 > x_2 > \dots > x_n$.

Imagine the variety $V(I)$ generated by some ideal I as a subset of the n -dimensional space. Then one could consider that what the above theorem is saying is that G_ℓ describes the variety V_ℓ defined as the projection of the original variety. This would mean, in particular, that any point of $V(I_\ell)$ is the projection of some point in $V(I)$. Unfortunately, this is not true, but it's really close.

Theorem 3.18 (Extension Theorem). Let $I = \langle f_1, \dots, f_k \rangle$ be an ideal in $\mathbb{C}[x_1, \dots, x_n]$ and let I_1 be its first elimination ideal. Assume

$$f_i = c_i(x_2, \dots, x_n)x_1^{n_i} + \text{other terms with smaller } x_1\text{-degree.}$$

Then, if a partial solution $(a_2, \dots, a_n) \in V(I_1)$ is not in $V(\langle c_1, \dots, c_k \rangle)$ (meaning that is not a common zero of all coefficients c_i), it can be extended to a solution $(a_1, \dots, a_n) \in V(I)$.

This means that we can almost always extend solutions of elimination ideals to get full solutions. This extension does not have to be a bijection (imagine, for instance, a line collapsing into a point when taking its projection), and we also might get complex solutions instead of just real (since neither \mathbb{Q} nor \mathbb{R} are algebraically closed). For this theorem to hold we explicitly need to be working over an algebraically closed field.

Remark 3.19. The condition of the partial solution not vanishing at all coefficients could be removed if we were working in the projective space, but this is not interesting for us.

The above theorems show that the theory of Gröbner basis provides an effective way of solving polynomial systems of equations by reducing them into alternative systems in fewer dimensions.

3.2 Gröbner basis algorithms

So far we have shown how Gröbner bases allow us to translate potentially hard algebraic problems into much simpler ones. Of course, this is only of use if the computation of a Gröbner basis is faster than solving these problems directly. This turns out to be the case.

The first algorithm for the computation of the Gröbner basis of an ideal was published by Buchberger himself in his dissertation. To start, given a set S of polynomials, we denote by

$$\overline{f^S}$$

the remainder of the division of the polynomial f by the elements of S . We then define the S -polynomial.

Definition 3.20. Let f, g be nonzero polynomials.

1. If $\text{multideg}(f) = \alpha$ and $\text{multideg}(g) = \beta$, then let

$$\gamma_i = \max(\alpha_i, \beta_i), \quad \gamma = (\gamma_1, \dots, \gamma_n).$$

We say $x^\gamma = \text{lcm}(\text{LM}(f), \text{LM}(g))$ is the least common multiple of $\text{LM}(f)$ and $\text{LM}(g)$.

2. The S -polynomial of f and g is defined as

$$S(f, g) = \frac{x^\gamma}{\text{LT}(f)}f - \frac{x^\gamma}{\text{LT}(g)}g.$$

Note that the quotients are indeed polynomials, as the leading term divides the least common multiple by its construction. One can see from the definition that the S -polynomial will remove the leading terms of both f and g . The key reason for this definition is the following.

Theorem 3.21 (*Buchberger's Criterion*). A basis $G = \{g_1, \dots, g_k\}$ of an ideal is a Gröbner basis if and only if

$$\overline{S(f_i, f_j)}^G = 0 \quad \forall i \neq j.$$

Since it is easy to both calculate the S -polynomials and the remainder of the divisions algorithmically, this gives us a good option to check whether a given basis is or not a Gröbner basis. The idea is then to forcefully create a set G that satisfies this criterion.

3 Gröbner basis

For a set G and two elements f_i and f_j , compute

$$g = \overline{S(f_i, f_j)}^G.$$

If g is zero there is nothing to do since we are in the path towards a Gröbner basis already. Otherwise, consider $G' = G \cup \{g\}$. Clearly they generate the same ideal, as

$$g = S(f_i, f_j) - \sum_{r=1}^k q_r f_r,$$

so g was in the ideal already. Moreover, $\overline{S(f_i, f_j)}^{G'}$ is now zero (since we have added the remainder). This leads to Buchberger's Algorithm, in algorithm 2.

Algorithm 2 Buchberger's Algorithm

Require: $F = \{f_1, \dots, f_k\}$

```

1:  $G \leftarrow F$ 
2: repeat
3:    $G' \leftarrow G$ 
4:   for  $\{p, q\} \subset G', p \neq q$  do
5:      $r \leftarrow \overline{S(p, q)}^{G'}$ 
6:     if  $r = 0$  then
7:        $G \leftarrow G \cup \{r\}$ 
8: until  $G = G'$ 
9: return  $G$ 

```

Theorem 3.22. Buchberger's Algorithm computes the Gröbner basis of an ideal $\langle f_1, \dots, f_k \rangle$.

Of course this algorithm is far from optimal. The main reason is that we are repeating many divisions of S -polynomials that should be zero for a Gröbner basis. This is actually a lengthy calculation, and there are two main ways to reduce the number of times we do it:

1. Improving our choice of S so that it can be computed more efficiently.
2. Adding *memory* to avoid comparing the same couple of polynomials over and over.

Improvements in these parts give rise to more efficient Buchberger-like algorithms.

However, there are other methods. The most well-known are those by Jean-Charles Faugère, known as F_4 (J.-C. Faugère 1999) and F_5 (J.-C. Faugère 2002). The strategy is, very broadly, to start by computing Gröbner basis of a smaller set of generators, and then when adding a new one to use sparse matrices to work in parallel.

As is always the case, the performance of any algorithm depends on the ordering we choose. In particular, the lexicographic ordering is usually the slowest, while the $>_{\text{grevlex}}$ order is the fastest. One common option when computing a lexicographic Gröbner basis is then to start by computing a $>_{\text{grevlex}}$ basis and then use a *conversion algorithm* to transform it into a $>_{\text{lex}}$ basis (an example is the FGLM algorithm, see J. C. Faugère et al.

1993; Cox, Little, and O’Shea 1998). This two-step process can usually be at worst in the same order of complexity of computing the lexicographic ordering from the beginning.

Finally, let us note that the Gröbner basis associated to an ideal is not unique. Indeed, one could keep adding polynomials g such that $\text{LT}(g) \in \langle \text{LT}(I) \rangle$ to the basis. In practice this means that an algorithm like Buchberger’s, based on adding remainders of many S -polynomials, will lead to a Gröbner basis with too many elements. A solution is considering *reduced* Gröbner basis.

Definition 3.23. A Gröbner basis G of an ideal I is said to be reduced if

1. All polynomials $p \in G$ are monic (so $\text{LC}(p) = 1$), and
2. For all $p \in G$, no monomial m of p satisfies $m \in \langle \text{LT}(G - \{p\}) \rangle$.

The first condition is there for the sake of uniqueness, since we can just divide by the leading term. The second means that all elements of G that do not add anything (because they are already in the ideal generated by the other elements) can be removed, and for the elements p that are important, the monomials that are in the ideal generated by the other leading terms can be removed and substituted with another expression of lower degree and not in the ideal. And indeed this leads to compact representation, as well as to the following uniqueness theorem.

Theorem 3.24. Every ideal $I \neq \{0\}$ has a unique reduced Gröbner basis.

This means that the problem of comparing two ideals I and J to see if they are equal can be reduced to finding a Gröbner basis for each and then checking if they are equal.

While the computation of a Gröbner basis allows us to solve systems of polynomial equations faster than with other methods, it is still very slow: depending on the algorithm and the ordering it can go from exponential at best up to doubly exponential. For the lexicographic ordering we have a lower bound of $n^3 2^{\mathcal{O}(n^3)}$ operations. In Bardet, J.-C. Faugère, and Salvy 2015, a lower bound of at least $2^{4 \cdot 29n}$ is given for the F_5 algorithm with $>_{\text{grevlex}}$ ordering, which is the best situation one could expect. Although translating such a basis into a lexicographic can be expensive (even exponential, depending on the ideal), it is still faster than the direct computation (in J. C. Faugère et al. 1993 one can find an analysis of the performance of the algorithm when the growth of the coefficients is taken into consideration).

3.3 Triangular sets

Another approach to the solution of systems of polynomial equations is triangular sets. First note that the theory of elimination is really similar to the way of solving systems of linear equations by building a triangular matrix, and in fact the latter can be considered a special case of the former.

The idea behind triangular sets is, starting from a set F of equations, to build a family T_1, \dots, T_k of sets of equations satisfying

$$V(\langle F \rangle) = V(\langle T_1 \rangle) \cup \dots \cup V(\langle T_k \rangle).$$

3 Gröbner basis

We present now a simple outlook on the theory of triangular sets, following Aubry, Lazard, and Moreno Maza 1999. We start from a ring

$$R_n = K[x_1, \dots, x_n]$$

and, for $p \in R_n$, we define $\text{MVAR}(p)$ to be the greatest variable appearing in p with respect to the lexicographic ordering. If $\text{MVAR}(p) = x_i$, then

$$p \in K[x_n, \dots, x_{i+1}][x_i].$$

Remark 3.25. We will use the lexicographic ordering with $x_1 > x_2 > \dots > x_n$ to follow the theory of Gröbner basis. In most of the literature about triangular sets, however, this order is inverted.

The definition of what constitutes a triangular set is really broad.

Definition 3.26. A subset T of R_n is called a *triangular set* if no element of T is a complex number, and if for all different pairs $p, q \in T$ we have $\text{MVAR}(p) = \text{MVAR}(q)$.

Therefore a set is triangular if a variable x_i appears as the main variable only in one of the equations, and in the rest either it doesn't appear at all or it is part of the coefficients of the main variable. This gives it a "triangular shape", similar to the case of matrices.

Under this definition, a Gröbner basis G could constitute an example of a triangular set, but this is not always the case. For instance, example 3.15 shows a computed Gröbner basis where two of the elements g_2 and g_3 share $\text{MVAR}(g_2) = \text{MVAR}(g_3) = x_2$. The definition being so broad means that different algorithms to compute triangular sets will arrive to different (albeit all valid) sets.

Example 3.27. Consider the polynomials

$$\{ c_1 c_2 - s_1 s_2 + c_1 - a, s_1 c_2 + c_1 s_2 + s_1 - b, c_1^2 + s_1^2 - 1, c_2^2 + s_2^2 - 1 \},$$

for the variable ordering $c_2 > s_2 > c_1 > s_1 > b > a$. A possible computed decomposition into triangular sets gives

$$\begin{aligned} T_1 &= \{(4b^2 + 4a^2)s_1^2 + (-4b^3 - 4a^2b)s_1 + b^4 + 2a^2b^2 + a^4 - 4a^2, \\ &\quad 2ac_1 + 2bs_1 - b^2 - a^2, \\ &\quad 2as_2 + (2b^2 + 2a^2)s_1 - b^3 - a^2b, \\ &\quad 2c_2 - b^2 - a^2 + 2\}, \\ T_2 &= \{a, 2s_1 - b, 4c_1^2 + b^2 - 4, s_2 - bc_1, 2c_2 - b^2 + 2\}, \\ T_3 &= \{a, b, c_1^2 + s_1^2 - 1, s_2, c_2 + 1\}. \end{aligned}$$

The last set T_3 gives $a = b = 0$, and from there one can easily find the values of the other variables. The set T_2 gives $a = 0$, with b being a free parameter. And for T_1 the values of a and b can be fixed to find a solution.

There are many approaches to finding sets T_i . In Aubry and Moreno Maza 1999, several methods are compared (including Lazard's method, which creates the decomposition in

the example above). For our systems, however, the lexicographic Gröbner basis methods will work faster than triangular sets algorithms.

Do note however that it's possible to use triangular sets to extend the usefulness of Gröbner basis, by computing the sets starting from a Gröbner basis using Möller's algorithm (see Möller 1993). This decomposition will lead to a number of systems of shape

$$\begin{aligned}
 t_n(x_n) &= x_n^{d_n} + \sum_{i=1}^{d_n-1} g_{n,i} x_n^i && \in K[x_n] \\
 t_{n-1}(x_{n-1}, x_n) &= x_{n-1}^{d_{n-1}} + \sum_{i=1}^{d_{n-1}-1} g_{n-1,i}(x_n) x_{n-1}^i && \in K[x_n, x_{n-1}] \\
 &\vdots \\
 t_1(x_1, \dots, x_n) &= x_1^{d_1} + \sum_{i=1}^{d_1-1} g_{1,i}(x_2, \dots, x_n) x_1^i && \in K[x_n, x_{n-1}, \dots, x_1].
 \end{aligned}$$

where $\text{MVAR}(t_i) = x_i$, which form a triangular set. Of course it will now be very simple to solve this type of systems starting from the first equation. This achieves a full triangularization of our ideal. In comparison with simply taking a Gröbner basis, note that elimination ideals may be generated by adding several polynomials with the same main variable. This means that sometimes to pass from one elimination ideal to the next we may have to solve a system of polynomial equations in one variable, which is not desirable. Möller's algorithm solves this problem by breaking the Gröbner basis into several triangular sets.

Example 3.28. Going back to example 3.15, we had reached a system

$$\begin{aligned}
 g_1 &= x_1 + x_2 + x_3^2 - 1, \\
 g_2 &= x_2^2 - x_2 - x_3^2 + x_3, \\
 g_3 &= 2x_2x_3^2 + x_3^4 - x_3^2, \\
 g_4 &= x_3^6 - 4x_3^4 + 4x_3^3 - x_3^2.
 \end{aligned}$$

Unfortunately, once we fix a value for x_3 , we have to solve equations g_2 and g_3 for x_2 simultaneously. If we instead compute now a triangular set over that basis (using Hillebrand 2018) we obtain the two triangular sets

$$\begin{aligned}
 T_1 &= \begin{cases} x_3^2, \\ x_2^2 - x_2 + x_3, \\ x_1 + x_2 - 1. \end{cases} \\
 T_2 &= \begin{cases} x_3^4 - 4x_3^2 + 4x_3 + 1, \\ 2x_2 + x_3^2 - 1, \\ 2x_1 + x_3^2 - 1. \end{cases}
 \end{aligned}$$

The sets are fully triangularized and we now only have to solve univariate polynomials at each step. Note that T_1 and T_2 differ mainly in the choice of which elements of the decomposition of g_4 one takes.

This type of decomposition will be useful for us in section 4.3.1.

3.4 The Multivariate Quadratic problem

The goal of solving the system defined by the gas network is in particular an instance of the more general *multivariate quadratic (MQ) problem* (finding all solutions of a system of m quadratic polynomials in n over some field K), as from the definition it can be seen that all equations in the system have degree at most two. Although it is simpler than the general case of solving a system of polynomial equations, due to the degree being bounded, this problem is still NP-complete.

This type of problems has received attention in the last years during the search for post-quantum cryptography cryptosystems (although they were first defined as far back as the 1980s, see for instance Matsumoto and Imai 1988). As a result of the efforts to break this type of schemes, some methods have appeared. The problem for us is that, since cryptographic protocols are based on finite fields (either \mathbb{F}_p for some large prime p or $(\mathbb{F}_2)^n$ for a medium value of n) they cannot be applied directly to our instance.

Moreover, the fact that we have a system of equations with the same number of equations m as the number of variables n is problematic. For cases where $m \geq \varepsilon n^2$ one can apply the linearization technique, by defining new variables $y_{ij} = z_i z_j$ and solving the original system, which becomes linear. If the number of equations is $m \geq n(n+1)/2 + n$, the above system can be solved exactly. Otherwise, the method of relinearization (see Kipnis and Shamir 1999) allows us to go down to $\varepsilon \approx 1/10$. Even further improvements are possible, as long as $m > n$ (see Courtois et al. 2000).

However, it can be shown that many methods for solving the MQ problem are actually instances of the F_4 and F_5 algorithm, so there is no improvement by using them as compared to simply opting for the Gröbner-based method (an overview of this can be found in Thomae and Wolf 2010). Therefore by constraining ourselves to the use of Faugère-based algorithms we are not losing much efficiency (and the bigger problem is the fact that $n = m$ in our system).

4 Solving gas networks

We will now focus on the main topic of interest. Recall that we have to solve a system of equations given by

$$\mathcal{A}_N^\top (\mathcal{A}_B^{-1})^\top \Phi_B \text{diag}(s_B) (\mathcal{A}_B^{-1} (q^{\text{nom}} - \mathcal{A}_N z))^2 = \Phi_N \text{diag}(s_N) z^2, \quad (4.1)$$

where z is the flow through the fundamental cycles of the system, and Φ and s represent the pressure drop coefficients and the pipe flow signs, respectively.

Once we have the values of z it is possible to calculate the rest of the gas flows as

$$Q_{0,B} = \mathcal{A}_B^{-1} (q^{\text{nom}} - \mathcal{A}_N z).$$

And since s is the vector of signs of the flows Q_0 , our solution will only be valid if the signs of s and Q_0 are equal.

However, this comparison happens *a fortiori*, because to calculate the values of z we need to start from a given vector of signs. The simplest option is simply to iterate over all possible signs, that is, through all elements of the set $\{ +1, -1 \}^{|A|}$.

Algorithm 3 Network solver.

```

1: for  $s \in \{ +1, -1 \}^{|A|}$  do
2:    $Z \leftarrow$  solutions of eq. (4.1) using Gröbner basis
3:   for  $z \in Z$  do
4:      $Q_{0,B} \leftarrow \mathcal{A}_B^{-1} (q^{\text{nom}} - \mathcal{A}_N z)$ 
5:      $Q_{0,N} \leftarrow z$ 
6:      $Q_0 \leftarrow [Q_{0,B} \mid Q_{0,N}]$ 
7:     if  $\text{diag}(s)Q_0 \geq 0$  then
8:       return  $Q_0$ 
9: return no solution found

```

Remark 4.1. The calculation of the Gröbner basis of an ideal is always sequential, therefore it can not be paralellized. However, the inner loop depends only on the value of s , and not on anything else. This means that it is possible to implement the algorithm with the outer loop in parallel, so that it can benefit from multicore processors.

Moreover, note that there are no benefits in using GPUs. While Faugère’s algorithm does involve some matrix operations (in particular with potentially sparse matrices), the improvement one could achieve fades when compared to the possibility of using modern CPUs with potentially dozens of (much faster) cores.

Remark 4.2. The main step in algorithm 3 is finding the set of solutions of eq. (4.1). The advantage of using a Gröbner approach is that, whenever the system has no solution, the solver will tell us so even faster than what it would take for it to find a solution. Indeed, for an ideal $I \subset \mathbb{C}[x_1, \dots, x_n]$ we have

$$V(I) = \emptyset \implies 1 \in I.$$

Note that we need to be working over the complex numbers (or any other algebraically closed field). This is essentially the statement of the *weak Nullstellensatz*. Therefore, if eq. (4.1) has no solutions, the Gröbner basis of its associated ideal will simply be $\{1\}$, and we will detect this really fast.

We have stated (and will state) many results that are useful only for varieties of dimension zero. In particular, this means that the variety will have a finite number of elements (which we need, as our algorithm does not work if there are infinitely many solutions). Note that we can write our variety as

$$V = V(I) = V(F_1) \cap \dots \cap V(F_n) \subset K^n,$$

where the F_i are the components of the system of equations we have to solve (see Shafarevich 2013). Since, for each F_i , we know that $F_i \neq 0$, we get that $\dim(V(F_i)) = n - 1$, so we are working with the intersection of n hypersurfaces. Moreover, we know that

$$\dim(V(F_i) \cap V(F_j)) = \dim(V(F_i)) - 1 \quad \text{if } F_j|_{V(F_i)} \neq 0,$$

so as long as no variety is contained in another, the dimension of the intersection will be

$$\begin{aligned} \dim(V(F_1) \cap \dots \cap V(F_n)) &= \dim(V(F_1) \cap \dots \cap V(F_{n-1})) - 1 \\ &\vdots \\ &= \dim(V(F_1)) - (n - 1) = 0. \end{aligned}$$

The condition we have is essentially that the F_i are independent from each other, meaning that there is no F_j such that $V(F_i) \subset V(F_j)$. This should always be the case in our equations, since otherwise it would mean that one of the equations depends on the others. If, for instance, F_1 depends on the others, then we could remove it and the system would not change, but that would imply that the behaviour of the gas network is independent of the coefficient $(\phi_N)_1$, which does not make sense. Of course there can be isolated cases where this reasoning does not apply (for instance by making all loads and all pressure drop coefficients be zero), but we simply discard those pathological cases.

4.1 Limits of the Gröbner solver

Let us start by studying to what extent can the previous algorithm be applied, in terms of size of the network (so dimension of q) and number of cycles (dimension of z). Note

Table 4.1. Timing comparison for the solution of different networks with different sets of parameters: $\mathbb{1}$ (all ones), \mathbb{Z} (integer loads), \mathbb{Q} (rational loads), U (loads uniform in the unit interval) and \mathcal{N} (normal loads). The values above are the average of 100 samples (only 10 for †, only 1 for ‡), the values below in (brackets) are the standard deviation.

Network shape	Nodes	Cycles	Timing (s)				
			$\mathbb{1}$	\mathbb{Z}	\mathbb{Q}	U	\mathcal{N}
	3	1	0.0170 (0.0018)	0.0172 (0.0013)	0.0161 (0.0020)	0.0199 (0.0012)	0.0198 (0.0010)
	5	1	0.0180 (0.0009)	0.0182 (0.0011)	0.0180 (0.0012)	0.0239 (0.0014)	0.0228 (0.0016)
	11	1	0.0208 (0.0127)	0.0210 (0.0012)	0.0211 (0.0024)	0.0359 (0.0102)	0.0345 (0.0011)
	4	2	0.0296 (0.0020)	0.0304 (0.0015)	0.0292 (0.0039)	0.0434 (0.0042)	0.0433 (0.0015)
	5	2	0.0313 (0.0023)	0.0314 (0.0018)	0.0303 (0.0041)	0.0492 (0.0035)	0.0490 (0.0018)
	4	3	0.0538 (0.0015)	0.0749 (0.0016)	0.0655 (0.0216)	0.1105 (0.0021)	0.1094 (0.0020)
	5	3	0.0701 (0.0022)	0.0765 (0.0023)	0.0736 (0.0127)	0.1256 (0.0049)	0.1268 (0.0088)
	9	3	0.0715 (0.0016)	0.0782 (0.0016)	0.0781 (0.0019)	0.1723 (0.0045)	0.1707 (0.0023)
	5	4	0.1743 (0.0040)	0.6611 (0.0262)	0.6186 (0.1371)	1.0050 (0.0366)	1.0260 (0.0493)
	8	4	0.1858 (0.0054)	0.6267 (0.0215)	0.6058 (0.0229)	0.9648 (0.0381)	0.9969 (0.0556)
	5	5	0.6404 (0.0114)	40.018 [†] (1.6983)	34.243 [†] (11.529)	55.352 [†] (1.6037)	57.716 [†] (3.6540)
	6	5	0.8259 (0.0057)	38.085 [†] (1.3471)	36.303 [†] (2.1699)	50.399 [†] (2.6935)	50.401 [†] (3.4210)
	5	6	7.9876 [†] (0.0313)	2520.3 [‡]	2294.6 [‡]	3384.3 [‡]	3384.8 [‡]

that we will always count only the number of fundamental cycles of the network (that is, cycles produced by adding an edge to the tree \mathcal{T}).

In all of the cases we simply took $s = 1$, regardless of the geometry of the network or how many flow directions were possible. A comparison of the timings for several different networks can be found in table 4.1. The different parameters are as follows:

- For $\mathbb{1}$, both the loads q^{nom} and the pressure drop coefficients ϕ are set to 1.
- For \mathbb{Z} , the loads are chosen to be integers in the set $\{-10, -9, \dots, 9, 10\}$, while the pressure drops are random natural numbers in the set $\{10^3, \dots, 10^6\}$.
- For \mathbb{Q} , the loads are random rational numbers p/q with p and q taken from the interval $\{10^4, 10^5\}$, the pressure drops are as in \mathbb{Z} .
- For U , the loads are taken from a uniform distribution $U(0, 1)$, the pressure drops are as in \mathbb{Z} .
- For \mathcal{N} , the loads are taken from a normal distribution $\mathcal{N}(0, 1)$, the pressure drops are as in \mathbb{Z} .

For the last network, the times shown correspond to only one sample.

Remark 4.3. All computations (here and on the rest of this thesis) were done using SageMath (see The Sage Developers 2020) as the backbone and interfacing to SINGULAR (see Decker et al. 2020). The timings were calculated in a computer running Linux (Ubuntu 18.04 on a 4.15.0 kernel), with an Intel® Xeon® E5-2620 v2 processor and 8 GB of RAM. To compute the solutions of a system of equations we used the SINGULAR function `lex_solve` (see Wenk and Pohl 2019).

There are several conclusions that can be drawn from table 4.1. The first is that the biggest factor in the runtime is the number of cycles. Once it is fixed, increasing the number of nodes of the network leads only to a moderate increase in the time it takes to find a solution. This could be expected beforehand, as the size of the system we have to solve remains the same: what the addition of new nodes means is mostly that the coefficients of the polynomials will become more convoluted (the other consequence, the increase in the number of flow directions, is not reflected here).

For a fixed network, the fastest runtime was of course achieved for the simplest choice of setting all parameters to one. For the other choices, they seem to be coupled: both \mathbb{Z} and \mathbb{Q} showed similar results, and similarly for U and \mathcal{N} . The relationship between \mathbb{Z} and \mathbb{Q} could be simply because whatever the choice of parameters it is likely that one will find large fractions in the Gröbner basis. Therefore taking either small integers or large fractions does not lead to an increase in the ‘complexity’ of the resulting basis. For U and \mathcal{N} , however, the fact that they are transformed into large rationals (meaning that, if written as p/q , both the numerator and the denominator will be very large) to account for their precision as floating point numbers does have a significant effect on the construction of the Gröbner basis. This difference is more clear as one considers larger examples: for the network with 6 nodes and 5 cycles, using normal loads leads to a 6000 % increase in running time when compared to the simple version.

It may be interesting to highlight one difference: in \mathbb{Z} and \mathcal{N} both positive and negative

loads are added, while in \mathbb{Q} and U all loads are positive, meaning that the network has only one entry point. However, this does not seem to lead to any significant difference. Finally, the results show clearly the issue with exponential behaviour. Although the smaller networks are able to perform really fast, for the slightly bigger ones an extra cycle can lead to a stark increase in running time. The series of networks with five nodes and an increasing number of fundamental cycles shows a good example, with a $10\times$ increase going from three to five cycles and another $10\times$ increase by adding an extra cycle in the simplest case with all parameters set to 1. When the loads and pressure drops are changed, the time required to obtain a solution explodes after the fourth cycle. This means that, in a realistic setting where the pressure drop coefficients and the loads are floating point number, networks with more than four fundamental cycles should not be solved using Gröbner-based methods.

There are other issues arising from the computation of Gröbner basis. The first is related to the degree of the resulting Gröbner basis. The resulting Gröbner basis of an ideal $I \subset K[x_1, \dots, x_n]$ generated by polynomials of degrees d is bounded by

$$\left((n+1)(d+1)\right)^{(n+1)2^{V(I)+1}}.$$

In our particular case, this would lead to a $(3(n+1))^{2(n+1)}$ upper bound for the degree of the Gröbner basis. This is just an upper bound, but in practice the degree of the resulting polynomials behave exponentially. Indeed, according to Lazard 1983, one should expect that, if $I = \langle f_1, \dots, f_k \rangle$ with degrees $\deg(f_i) = d_i$ sorted from biggest to smallest, then the degree $\deg(I)$ of the Gröbner basis satisfies

$$\deg(I) \leq d_1 \cdot \dots \cdot d_{n-\dim V(I)}.$$

But, in the particular case where $k = n - \dim V(I)$, the inequality becomes an equality, giving for our ideals that $\deg(I) = 2^n$ (although this comes with an *almost always* attached). In section 4.2.1 this behaviour will be seen.

The other problem comes from the size of the coefficients. The cancellation of leading terms produced by considering S -polynomials also introduces products and divisions by leading coefficients, which adds up to complicate the resulting polynomials. For instance, if we take as graph the complete graph K_4 on four vertices, with all parameters (pressure drops, loads and directions) being set to one, we obtain a system

$$\begin{aligned} F_1 &= 2z_1^2 + 4z_1z_2 + 4z_1z_3 - 12z_1 + 2z_2^2 + 2z_2z_3 - 10z_2 + 2z_3^2 - 6z_3 + 14, \\ F_2 &= 2z_1^2 + 4z_1z_2 + 2z_1z_3 - 10z_1 + z_2^2 + 2z_2z_3 - 10z_2 + z_3^2 - 4z_3 + 13, \\ F_3 &= 2z_1^2 + 2z_1z_2 + 4z_1z_3 - 6z_1 + z_2^2 + 2z_2z_3 - 4z_2 + z_3^2 - 6z_3 + 5, \end{aligned}$$

with Gröbner basis

$$\begin{aligned} g_1 &= z_1 - \frac{896}{22815}z_3^6 + \frac{368}{2535}z_3^5 - \frac{568}{22815}z_3^4 + \frac{376}{22815}z_3^3 + \frac{782}{7605}z_3^2 - 1, \\ g_2 &= z_2 + \frac{46}{22815}z_3^6 + \frac{392}{2535}z_3^5 - \frac{11347}{22815}z_3^4 - \frac{23519}{91260}z_3^3 + \frac{17567}{30420}z_3^2 - 1, \end{aligned}$$

$$g_3 = z_3^7 - 3z_3^6 - \frac{5}{2}z_3^5 + \frac{83}{8}z_3^4 - \frac{15}{4}z_3^3 - \frac{117}{8}z_3^2.$$

The coefficients of the g polynomials are much more convoluted than those of F , despite the relative simplicity of the equations. This last system of polynomials has $\text{length}(I) = 213$, where by length we simply mean the amount of characters needed to write it out in the most simplified way. If we calculate in the same way the length of the Gröbner basis associated to the ideal of the same network but with pressure drops

$$\phi = (532632, 488351, 966319, 102511, 724872, 456706)$$

chosen at random, we get $\text{length}(I) = 12541$. If we further complicate it by setting

$$q^{\text{nom}} = (0.90584491, 0.39104003, 0.67354593),$$

also chosen randomly, then we get $\text{length}(I) = 26134$. Therefore even for very small networks the coefficients explode. In particular, note that the Gröbner basis will have at most 26 monomials (due to the conditions on $\text{deg}(I)$ and the Shape Lemma which will be seen below), therefore the average coefficient size per monomial for the last basis will be of 1000 characters.

4.2 Stability of the computations

So far, we have an algorithm that we know works, meaning that it finds the correct solution in finite time. However, in practice there may be problems arising from the implementation of the algorithm. The first of them is time, as we know. The second is due to the inability of the computer to perform exact arithmetic.

There are two ways that this type of errors can appear.

1. The computation of the Gröbner basis is generally based on the computation of many S -polynomials and their remainders. This is effective because of the cancellation of the main terms (see Sasaki and Kako 2010, § 2 for an in-depth analysis of how different cancellations can happen). However, when the coefficients of the polynomials are very small (such that their difference may be in the order of magnitude of the floating-point accuracy of our computer) there may be unwanted cancellation of terms that leads to potentially large errors in further computations.
2. Once we have found the Gröbner basis of the ideal, what we have is a reformulation of the original system of equations that has the same solutions. The issue is that this system of equations will potentially be of a much higher degree compared to the original, and with more complex coefficients. While numerical solvers can reliably find all roots of a univariate polynomial up to a very high degree of precision, even the smallest error in the approximation of the solutions can create a cascade effect: the next polynomials to be solved (after the solved variable is substituted) will be incorrect (since the value of the substitution introduces an error), leading to an increased error.

4.2.1 Floating point errors

The first problem has been studied on many occasions. A common approach is setting an infinitesimal quantity ε such that $\varepsilon^m = 0$, which can be used by working in the ring

$$\hat{R} = R / \langle \varepsilon^m \rangle,$$

and then using it to take care of quantities that are under the floating point of the computer. This makes it possible to define an *extended Gröbner basis* that accounts for small perturbations, as in Kondratyev, Stetter, and Winkler 2004. Other methods can be found in Sasaki and Kako 2007, Sec. 3.

The easiest solution to this problem is refraining from using floating point numbers at all. In particular, we can assume that all of the coefficients are rational numbers, and perform exact arithmetic to avoid issues with an engine that is powerful enough to handle them. The price we pay is, as explained above, that the size of the coefficients (understood not as the value, but the space needed to store them) becomes bigger and bigger.

In practice, in the construction of the system of equations we have to solve, the problem does not come from the values of the pressure drop coefficients ϕ , since these tend to be large integers and therefore it is easier to work with them. Instead, the problem comes from the load vector q^{nom} that has the potentially small floating point numbers, that we may not want to turn into rational numbers. A possible solution can then be solving the equation with the unknown values as parameters.

Consider the ring

$$\hat{R} = \mathbb{Q}[q_1^{\text{nom}}, \dots, q_{|V|}^{\text{nom}}][z_1, \dots, z_n]$$

of polynomials in z with coefficients that can depend on q . For $f \in \hat{R}$, and a vector q of loads, let

$$\pi_q(f) = f(q, z) \in \mathbb{Q}[z_1, \dots, z_n]$$

be the result of the substitution of the loads in the system. One would hope that, if

$$I = \langle f_1, \dots, f_k \rangle \subset \hat{R}$$

is an ideal with Gröbner basis $\{g_1, \dots, g_t\}$, then a basis for the system

$$I_q = \langle \pi_q(f_1), \dots, \pi_q(f_k) \rangle \subset \mathbb{Q}[z_1, \dots, z_n]$$

would be given by $\{\pi_q(g_1), \dots, \pi_q(g_t)\}$. While that is not the case, there is a close enough result from Weispfenning 1992.

Definition 4.4. Let $\mathcal{G} = \{(V_i, G_i) \mid 1 \leq i \leq n\}$, where the V_i are algebraic sets that cover $\mathbb{Q}^{|V|}$ and the $G_i \subset \mathbb{Q}[z_1, \dots, z_n]$ are sets of polynomials, and let \prec be a term order with $q \prec z$, meaning that any term with coefficients in q is bigger than a term without them. We say that \mathcal{G} is a *comprehensive Gröbner system* (or simply a Gröbner system) if, for all values $q = (q_1, \dots, q_{|V|}) \in \mathbb{Q}^{|V|}$, we have that, if $q \in V_i$, then

$$G_q = \{\pi_q(g) \mid g \in G_i\}$$

is a Gröbner basis in $\mathbb{Q}[z_1, \dots, z_n]$ with respect to the order induced by \prec .

4 Solving gas networks

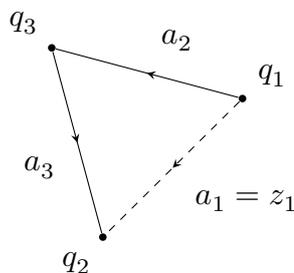
This means that, instead of having to compute the Gröbner basis of a system for every possible vector q of loads, one has to first compute the Gröbner system and then just find to which set V_i does q belong. Each tuple (V_i, G_i) is called a branch, and for the algorithms we will work with we will have $V_i = V(E_i) - V(N_i)$, with E_i and N_i ideals in $\mathbb{Q}[q_1, \dots, q_{|V|}]$

Of course, this is only useful insofar there are algorithms that allow one to compute the Gröbner system associated to an ideal. But that is the case: an algorithm can be found in Kapur, Sun, and Wang 2010, and it is also implemented in SINGULAR (see Montes and Schoenemann 2018).

Remark 4.5. In Nitsche 2018, the idea of Gröbner systems is used to obtain parametrizations $z(q)$ of the flows through the cycle, leading to a parametric representation of the set of feasible loads.

Having a Gröbner system allows us to study the conditioning of our system, as in Montes 1998; Montes and Castro 1995. Consider the following simple example.

Example 4.6. Consider the simple network formed only by a triangle, with the following variables:



We let all pressure drops be 1, and denote by the dashed line the pipe whose flow we will calculate (that is, $Q_{0,N}$). A Gröbner system in this case consists of only one couple (V, G) , with V being therefore the entire space \mathbb{Q}^2 (recall that $q_1 = -q_2 - q_3$ is omitted). The associated Gröbner basis is simply the polynomial

$$z_1^2 + (-4q_2 - 2q_3)z_1 + 2q_2^2 + 2q_2q_3 + q_3^2,$$

which is a quadratic polynomial on z_1 . If we let

$$A = 1, \quad B(q_2, q_3) = -4q_2 - 2q_3, \quad C(q_2, q_3) = 2q_2^2 + 2q_2q_3 + q_3^2$$

we obtain

$$\begin{aligned} z_1 &= \frac{-B(q_2, q_3) \pm \sqrt{B(q_2, q_3)^2 - 4C(q_2, q_3)}}{2} \\ &= 2q_2 + q_3 \pm \sqrt{2q_2(q_2 + q_3)}. \end{aligned}$$

We can now compute the condition number of the system, by computing the derivatives

$$\frac{\partial z_1}{\partial q_2} = 2 \pm \frac{2q_2 + q_3}{\sqrt{2q_2(q_2 + q_3)}},$$

$$\frac{\partial z_1}{\partial q_3} = 1 \pm \frac{q_2}{\sqrt{2q_2(q_2 + q_3)}}.$$

Then we get

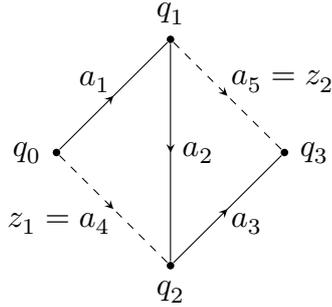
$$C(z_1, q_2) = \left| \frac{q_2}{z_1} \right| \left| \frac{\partial z_1}{\partial q_2} \right| = \frac{2 \pm \frac{2q_2 + q_3}{\sqrt{2q_2(q_2 + q_3)}}}{2 + \frac{q_3}{q_2} + \sqrt{2 + 2\frac{q_3}{q_2}}}.$$

If we assume that $q_2 \sim q_3$, we would have

$$C(z_1, q_2) \sim \frac{2 \pm 3/2}{2 + 1 + \sqrt{4}} = \frac{2}{5} \pm \frac{3}{10},$$

which is a relatively low number. A similar calculation can be made for $C(z_1, q_3)$. Therefore the system is *stable*, in the sense that the flow z_1 will not change drastically with changes in the loads, as long as the loads are similar.

Example 4.7. For a more challenging task, let us consider the example in Gotzes, Nitsche, and Schultz 2017, § 4. The network has the shape



Unlike in the previous example, this time we set $\phi = (1, 1, 100, 1, 1)$. In the notation of the previous paper, we will work on Orientation II (and, in the notation of Nitsche 2018, § 7, we will be in Direction 1). We take the first branch of the Gröbner system, and get the polynomials

$$\begin{aligned} g_1 &= (4q_1^2 + 8q_1q_2 + 16q_1q_3 + 4q_2^2 + 16q_2q_3 + 16q_3^2)z_2^2 \\ &\quad + (4q_1^3 + 4q_1^2q_2 - 4q_1q_2^2 - 24q_1q_2q_3 - 24q_1q_3^2 - 4q_2^3 - 24q_2^2q_3 - 40q_2q_3^2 - 24q_3^3)z_2 \\ &\quad + q_1^4 - 2q_1^2q_2^2 - 4q_1^2q_2q_3 + 8q_1q_2q_3^2 + 8q_1q_3^3 + q_2^4 + 4q_2^3q_3 + 12q_2^2q_3^2 + 16q_2q_3^3 + 8q_3^4, \\ g_2 &= (2q_1 + 2q_2 + 2q_3)z_1 - 2q_3z_2 - q_1^2 - 2q_1q_2 - 2q_1q_3 - q_2^2 - 2q_2q_3, \end{aligned}$$

corresponding to the system

$$\begin{aligned} 0 &= A(q_1, q_2, q_3)z_2^2 + B(q_1, q_2, q_3)z_2 + C(q_1, q_2, q_3), \\ 0 &= M(q_1, q_2, q_3)z_1 + N(q_1, q_2, q_3, z_2). \end{aligned}$$

Table 4.2. Timing comparison between the computation of a Gröbner system for different networks with different values for the pressure drop coefficients ϕ . The values above are the average of 1000 samples, the values below in (brackets) are the standard deviation.

Network shape	Number of nodes	Number of cycles	Timing (s)	
			$\phi = \mathbb{1}$	$\phi \in \mathbb{N}$
	3	1	0.01722 (0.00212)	0.01831 (0.00275)
	11	1	0.02112 (0.00206)	0.02118 (0.00248)
	4	2	0.03610 (0.01485)	0.04513 (0.00813)
	8	1	0.04631 (0.00356)	0.06377 (0.01021)

It is therefore trivial to solve for z_2 and z_1 . To compute the partial derivatives, there is not much to do when working with z_2 ; for z_1 , in particular with $N(q_1, q_2, q_3, z_2)$ we simply use the fact that

$$\frac{\partial f(z_2)}{\partial q_i} = \frac{\partial f(s(q_1, q_2, q_3))}{\partial q_i} = \frac{\partial f(s)}{\partial s} \frac{\partial s(q_1, q_2, q_3)}{\partial q_i} = \frac{\partial f(s)}{\partial s} \frac{\partial z_2}{\partial q_i}.$$

This allows us to compute $C = C(z_i, q_j)$, obtaining, for $q = (10, 100, 10)$, the matrix

$$C = \begin{pmatrix} 0.09430 & 1.05609 & 0.03821 \\ 0.06874 & 0.80760 & 0.12364 \end{pmatrix}.$$

In this case, for example, this tells us that if we increase the load q_3 of the rightmost node, we can expect the load to increase more on the z_2 pipe than on the z_1 (so that most of the flow would come from the (q_0, q_1, q_3) path).

Although we know that there are algorithms to compute a Gröbner system, it is important to know how long can this take. Of course, we can expect it to be more expensive than simply computing the Gröbner basis for a set of values of q . In table 4.2 a comparison of different networks is shown, alongside the time it took to find a Gröbner system (for the simplest problem with three fundamental cycles, the complete graph K_4 , no solution could be found even after several days). For all of the systems we simply solved for one direction; in a realistic scenario one would need to solve as many systems as possible flow directions there are.

There were two scenarios to find a Gröbner system. The first one, $\phi = \mathbb{1}$, involves simply setting all pressure drops to one. For the second, $\phi \in \mathbb{N}$, we choose each pressure drop from a uniform distribution over the set $\{1, 2, \dots, 10000\}$. For each of them table 4.2 shows the average and standard deviation of the time taken over 1000 samples.

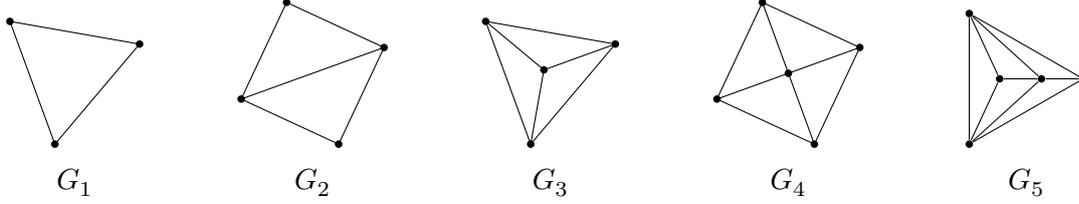


Figure 4.1. Graphs for the degree comparison.

4.2.2 Approximation errors

For this second problem, it may be worth looking at some practical examples. Consider the graphs G_1 , G_2 , G_3 , G_4 and G_5 , with 1 to 5 fundamental cycles, as in fig. 4.1. We let all coefficients (q^{nom} , ϕ and s) be simply one, for simplicity. If we compute the Gröbner basis and look at the last polynomial g_{-1} (the one eliminating the last variable) we get

$$\begin{aligned}
 g_{-1}(G_1) &= z_1^2 - 6z_1 + 5, \\
 g_{-1}(G_2) &= z_2^4 + 12z_2^3 + 18z_2^2 + 9, \\
 g_{-1}(G_3) &= z_3^7 - 3z_3^6 - \frac{5}{2}z_3^5 + \frac{83}{8}z_3^4 - \frac{15}{4}z_3^3 - \frac{117}{8}z_3^2, \\
 g_{-1}(G_4) &= z_4^{16} + \frac{34693728}{105313}z_4^{15} + \frac{4469888320}{105313}z_4^{14} + \dots + \frac{609004552192}{8101}z_4^2, \\
 g_{-1}(G_5) &= z_5^{32} + \dots + \frac{22334709456027247610838975}{396908347641439}
 \end{aligned}$$

One can clearly see that the degree of the last polynomial gets bigger and bigger. This is something that happens in virtually every computation of Gröbner basis, due to the fact that the computation of the S -polynomial, while cancelling the leading term, increases the other terms by multiplying them with another monomial, and this may not disappear when taking the remainder of the division by the other elements of the basis.

Imagine the system has a total of d solutions, which we write as

$$z^{(1)} = (z_1^{(1)}, \dots, z_n^{(1)}), \dots, z^{(d)} = (z_1^{(d)}, \dots, z_n^{(d)}).$$

Then the polynomial

$$p_n(z_n) = \prod_{i=1}^d (z_n^{(i)} - z_n)$$

is the one generated by the last coordinate of all roots of the original system of equations. Therefore

$$p_n(z_n)^m \in I_{n-1} \cap K[z_n] = \langle g_{-1}(z_n) \rangle,$$

where I_{n-1} is the last elimination ideal. The fact that one has to take an exponent m for p_{-1} to be in the ideal is due to Hilbert's *Nullstellensatz*. But this implies that all roots of g_{-1} are also roots of p_{-1} so, if all the $z_n^{(i)}$ are different, we have that $\deg(g_{-1}) \geq d$. Therefore, if the system has many equations with many solutions the last generator will be of higher degree.

These polynomials are of relatively high degree, and although it is not hard to get precise numerical solution any error, as mentioned before, could be multiplied when solving the other variables.

Example 4.8. Consider the network given by the graph in fig. 4.2. This graph has five fundamental cycles. We fix the pressure drop coefficients in all pipes to be 1, and consider 100 random networks by choosing the coefficients randomly from a normal multivariate $q^{\text{nom}} \sim \mathcal{N}(1, 5 \cdot \text{Id}_4)$. We obtain a total of 652 solutions. For each solution, we compute the mean squared error as follows: if

$$F(z) = (F_1(z_1, z_2, z_3, z_4, z_5), \dots, F_5(z_1, z_2, z_3, z_4, z_5))$$

is the formulation of the system of eq. (4.1), and $\hat{z} = (\hat{z}_1, \dots, \hat{z}_5)$ is the solution computed by the algorithm, then we define

$$\text{MSE}(\hat{z}) = \frac{1}{5} \sum_{i=1}^5 |F(\hat{z}_i)|^2,$$

the mean square error of \hat{z} . Essentially, $\text{MSE}(\hat{z})$ tells us how close the solution computed via the Gröbner basis elimination method is from being a solution of the original system. If E is the result of calculating the mean squared error of all solutions, we obtain

$$\begin{aligned} \mu(E) &= 1.37875 \times 10^{-17}, & \mu(-\log(E)) &= 25.8581, \\ \sigma(E) &= 2.46128 \times 10^{-16}, & \sigma(-\log(E)) &= 1.71067, \\ \min(E) &= 8.28303 \times 10^{-31}, & \min(-\log(E)) &= 14.2350, \\ \max(E) &= 5.82034 \times 10^{-15}, & \max(-\log(E)) &= 30.0818, \\ \text{median}(E) &= 1.11122 \times 10^{-26}, & \text{median}(-\log(E)) &= 25.9542. \end{aligned}$$

We also provide the values for $-\log(\text{MSE})$, since the distribution is highly skewed by some of the values.

However, consider this same system for the specific set of parameters

$$q_2 = -7.77600224, \quad q_3 = 0.45935455, \quad q_4 = -0.24251548, \quad q_5 = 5.27920738.$$

On a first glance this set of parameters has nothing particular; however, it leads to a solution \hat{z} with

$$\text{MSE}(\hat{z}) = 0.157.$$

Compared to the previous values, this is a really high error. Note that in this case the system gives rise to a Gröbner basis that has a *good* shape, as in the Shape Lemma below, hence the problem comes from the numerical error introduced when solving a polynomial of very high degree.

There is a particular case in which we can guarantee that at least the numerical error will not create a cascade effect and generate even bigger errors.

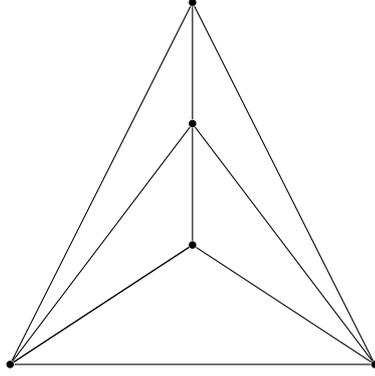


Figure 4.2. Network from example 4.8.

Theorem 4.9 (*Shape Lemma*). Let I be a radical, zero-dimensional ideal in $\mathbb{C}[z_1, \dots, z_n]$ such that the last coordinate of all the d points in $V(I)$ is different. Then a $>_{\text{lex}}$ basis of I is given by

$$\{ z_1 - g_1(z_n), \dots, z_{n-1} - g_{n-1}(z_n), g_n(z_n) \},$$

where $\deg(g_n) = d$ and $\deg(g_i) < d$ for $1 \leq i \leq n - 1$.

For a proof of the theorem one can check Becker et al. 1994.

The meaning for us is that, under some circumstances, the problem of the error in the calculations of z_n, z_{n-1} down to z_1 being propagated disappears, since the values of z_i for $1 \leq i \leq n - 1$ depend only on z_n and not on each other.

From the definition, one should expect our Gröbner basis to have this good shape. From a theoretical point of view, all points having different last coordinate makes sense: if we fix the last coordinate we're left with a polynomial system with one more equation than variables. The system is therefore overdetermined, and solutions to this system, if they exist, should be scarce, so it would be unlikely to find two solutions for this fixed last variable. Testing also supports this conclusion: we performed random tests for various systems with integer loads between 1 and 10^6 , and loads taken either from a uniform distribution $q \sim U(0, 1)$ or from a normal distribution $q \sim \mathcal{N}(0, 1)$. In all tests, the resulting Gröbner basis had the wanted shape.

Unfortunately, there are cases where this fails. In Nitsche 2018, § 7, an example is computed where one of the Gröbner basis resulting from a Gröbner system has an unwanted shape. In that particular case, it stems from a “bad choice” of the loads q . And indeed one could imagine that the previous argument about an underdefined system fails if additional term cancellations appear due to the choice of loads. In terms of the Gröbner system, this would mean that we could expect the first (and biggest) branch of the system to be

$$V_1 = \mathbb{Q}^n - V(N_i)$$

with $V(N_i)$ a small variety (in practical terms all algebraic varieties are small) defined by a set of equations that produce (unwanted) cancellation of z terms in the definition of the system.

4.3 Finding the flow directions

In algorithm 3, iterating over all possible signs and then solving the corresponding equations gives us an $\mathcal{O}(2^{|A|})$ extra layer of complexity, which is of course very undesirable. There are two possible ways to reduce the number of flow directions one has to consider. The first is simply looking at the inherent restrictions of the network. While we are only considering nodes and pipes, our network will also have control valves and compressors that we have just simplified. This type of components has inherent restrictions that force the gas to flow in one particular direction, therefore fixing the flow through some of the pipes which include the components.

For the second, we will start from the work in Nitsche 2018, Prp. 5.7.

Proposition 4.10. Let G be a graph representing a given network, \mathcal{T} a depth-first tree rooted in the reference node r , \vec{G} the orientation of G with respect to \mathcal{T} and Q_0 a vector of flows. Then:

1. $\deg_{\text{gas}}^{\text{out}}(u) \geq 1$ for all entry nodes with $q_u^{\text{nom}} < 0$.
2. $\deg_{\text{gas}}^{\text{in}}(u) \geq 1$ for all exit nodes with $q_u^{\text{nom}} > 0$.
3. Either $\deg_{\text{gas}}(u) = 0$, or $\deg_{\text{gas}}^{\text{in}}(u) > 0$ and $\deg_{\text{gas}}^{\text{out}}(u) > 0$ for all innodes with $q_u^{\text{nom}} = 0$.
4. There are no cycles in \vec{G} directed with respect to the gas flow.

Here, the ‘gas’ subtitle means the graph G oriented in the direction of the gas flow. The original proposition has some more conditions for a possible flow when there is only a single entry node (or equivalently a single exit node), but we will not cover them here.

Proof.

1. We have

$$q_u^{\text{nom}} = \sum_{\text{head}_{\text{gas}}(a)=u} |Q_{0,a}| - \sum_{\text{tail}_{\text{gas}}(a)=u} |Q_{0,a}|.$$

If $\deg_{\text{gas}}^{\text{out}}(u) = 0$, the second sum will be empty, so we are left with a sum with only positive terms, leading to

$$0 > q_u^{\text{nom}} = \sum_{\text{head}_{\text{gas}}(a)=u} |Q_{0,a}| \geq 0,$$

giving a contradiction.

2. The proof is just as above, just changing the signs.
3. In this case we have

$$0 = q_u^{\text{nom}} = \sum_{\text{head}_{\text{gas}}(a)=u} |Q_{0,a}| - \sum_{\text{tail}_{\text{gas}}(a)=u} |Q_{0,a}|,$$

so either both sums are empty or both have at least one element, which gives the statement of the proposition. Let the cycle be given by vertices $(u_1, u_2, \dots, u_n, u_1)$

and edges $a_1 = (u_1, u_2), \dots, a_n = (u_n, u_1)$, with all flows positive. Then

$$0 = \sum_{i=1}^n p_{u_{i+1}}^2 - p_{u_i}^2 = \sum_{i=1}^n (-\phi_{a_i} Q_{0,a_i}^2) = - \sum_{i=1}^n \phi_{a_i} Q_{0,a_i}^2.$$

Since everything inside the sum is positive, the only way for the equality to hold is to have $\phi = 0$, which we discard since it has no physical meaning to us, or to have no actual flow throughout the cycle. \square

The above result allows us to define a strictly smaller set of possible flow directions,

$$\mathcal{S} \subset \{ +1, -1 \}^{|A|}.$$

The computation of this set is hard in general, because the number of pipes with fixed flow directions may be small and the conditions on the degree of incoming and outgoing edges at every node may not be useful whenever the graph is highly connected.

Proposition 4.11. By abuse of notation, let $\mathcal{S}(G)$ be the number of possible flow directions of the graph, i.e. the size of the set \mathcal{S} associated to \vec{G} . Assume that there is only one entry node r , with $\deg_{\text{gas}}^{\text{in}}(r) = 0$, and that all other nodes are exit nodes. Then

1. $\mathcal{S}(\mathcal{T}) = 1$ for any tree.
2. $\mathcal{S}(C_n) = n - 1$ for a cycle graph.
3. $\mathcal{S}(W_n) = 2^{n-1}$ for a wheel graph, if the entry node r is at the center.
4. $\mathcal{S}(K_n) = (n - 1)!$ for a complete graph.

Proof.

1. For an intuitive proof, look at a single leaf. If it is the root node r , the flow through the connecting edge must go outwards to the rest of the tree; otherwise the flow goes inwards to ensure that the exit node gets gas. Therefore we can remove all leaves and look at the parent. Repeating this process gets us the result.

Alternatively, the proof for Nitsche 2018, Prp. 5.7.(v) gives us that the flow through edges that are not in a cycle are directed away from the node r . Since every edge in a tree is not part of a cycle, this means that the flow through every edge is predetermined.

2. Let the vertices be $u_1 = r, u_2, \dots, u_n$, with $a_i = (u_i, u_{i+1})$. We can immediately settle the direction of a_1 since it flows outwards from r . We can then choose a direction for a_2 . If $\vec{a}_2 = (u_2, u_3)$, we jump to choosing a_3 . But if $\vec{a}_2 = (u_3, u_2)$, there are no more choices, and every other edge will be $\vec{a}_i = (a_{i+1}, a_i)$ for $i \geq 3$, since otherwise a vertex in the middle would have no incoming edges.

Therefore any viable direction will be a directed path from u_1 to u_k in the two possible ways around the cycle. Therefore the amount of possible directions is the number of choices for k , which is $n - 1$.

3. Let the vertices of the graph W_{n+1} be r, u_1, \dots, u_n . Then all edges $a_i = (r, u_i)$ have a fixed direction, away from r . The remaining n vertices $b_j = (u_j, u_{j+1})$ can be

ordered arbitrarily, giving a total number of 2^n possibilities. However, we have to discard those that lead to cycles. But since the a_i are all ordered outwards, the only possible cycles would be having $\vec{b}_j = (u_j, u_{j+1})$ for all j or the other way around, giving two combinations that we have to remove, and leading to $\mathcal{S}(W_{n+1}) = 2^n - 2$, which is equivalent to what was claimed.

4. Similarly with the case of wheels, we can *remove* from the picture the first vertex as all directions are fixed, and what we get is a smaller complete graph K_{n-1} . We want to orient all edges in this subgraph such that no cycles appear (because no cycles including the root vertex are possible). One can show that every acyclic graph has an acyclic ordering of its vertices, where acyclic ordering means that if $\vec{a} = (u_i, u_j)$ then $i < j$, see for instance Bang-Jensen and Gutin 2001, Prp. 1.4.3. Similarly, for any ordering of the vertices we can create an acyclic graph: if the order is $u_{t_1} < \dots < u_{t_{n-1}}$, create vertices $\vec{a}_{ij} = (u_{t_i}, u_{t_j})$ for $1 \leq i < j \leq n - 1$.

The above implies that an acyclic complete graph on $n - 1$ vertices is equivalent to an ordering of $\{1, \dots, n - 1\}$. Since there are $(n - 1)!$ such orderings, one gets the result $\mathcal{S}(K_n) = (n - 1)!$. \square

Remark 4.12. For the case of trees, one does not even need to consider the direction of the flows. Looking back to eq. (2.6), we have

$$\mathcal{A}Q_0 = q^{\text{nom}},$$

but \mathcal{A} is now a nonsingular square matrix, so we can get the network flows directly.

Note that the vertex connectivity is respectively $\kappa(T) = 1$, $\kappa(C_n) = 2$, $\kappa(W_n) = 3$ and $\kappa(K_n) = n - 1$, for $n \geq 4$. The values for the edge connectivity λ are all equal. This means that even a small increase in connectivity can cause an explosion in the potential size of \mathcal{S} if we do not have additional information. The hope, however, is the fact that for graphs with small connectivity the size is relatively small, and in particular in the case of trees.

Proposition 4.13. Let G be a graph with edge connectivity $\lambda(G) = 1$, and divide it into subgraphs G_1 and G_2 such that there is only one edge a between them. Then

$$\mathcal{S}(G) \leq \mathcal{S}(G_1)\mathcal{S}(G_2).$$

Proof. One has the trivial inequality $\mathcal{S}(G) \leq 2\mathcal{S}(G_1)\mathcal{S}(G_2)$ by splitting every set of flow directions into a flow direction for each of G_1 and G_2 , with the factor of two coming from the two possible directions for \vec{a} . However, there is only a possible direction for a : let

$$q_1^{\text{nom}} = \sum_{u \in V(G_1)} q_u^{\text{nom}}, \quad q_2^{\text{nom}} = \sum_{u \in V(G_2)} q_u^{\text{nom}}.$$

Since the network is supposed to be balanced, $q_1 = -q_2$. If, without loss of generality, $q_1 < 0 < q_2$, we must have that the flow of \vec{a} goes from G_1 to G_2 . Therefore this direction is fixed, which means that one can decompose a flow direction of G into one in G_1 and other in G_2 . This then gives

$$\mathcal{S}(G) \leq \mathcal{S}(G_1)\mathcal{S}(G_2). \quad \square$$

One could give a similar result for graphs with vertex connectivity $\kappa(G) = 1$. One could even go further: this result is not only about the number of flow directions $\mathcal{S}(G)$, but the argument can also be applied to say that the two subnetworks G_1 and G_2 can be solved separately, and once the flows have been found they can be patched together to get a solution for G . Therefore, solving a network with edge connectivity $\lambda(G) = 1$ is as hard as solving its subnetworks. This is the starting point for the reduction method in chapter 5.

4.3.1 Parametrized flow directions

For now we have been looking at the set \mathcal{S} and solving for z once the directions have been fixed. However, there is another possible approach. Consider instead the polynomial ring to be

$$R_s = K[s_1, \dots, s_{|A|}, z_1, \dots, z_n]$$

where the s_i variables are associated to each edge. The main condition on the directions is the fact that $s_i \in \{ +1, -1 \}$, which we could encode by working in the ring

$$R_s / \langle s_1^2 - 1, \dots, s_{|A|}^2 - 1 \rangle = R_s / I_s.$$

The ideal which we quotient enforces $s_i^2 - 1 = 0$, which implies $s_i = \pm 1$. If we define the system of equations which we are trying to solve as

$$I = \langle \mathcal{A}_N^\top (\mathcal{A}_B^{-1})^\top \Phi_B \text{diag}(s_B) \left(A_B^{-1} (q^{\text{nom}} - \mathcal{A}_N z) \right)^2 - \Phi_N \text{diag}(s_N) z^2 \rangle,$$

we can find the solutions with the corresponding signs in the variety

$$V_{R_s}(I + I_s).$$

The addition of ideals is simply the set of elements that are the sum of elements in each ideal. This variety is essentially equivalent to the system

$$\begin{aligned} \mathcal{A}_N^\top (\mathcal{A}_B^{-1})^\top \Phi_B \text{diag}(s_B) \left(A_B^{-1} (q^{\text{nom}} - \mathcal{A}_N z) \right)^2 &= \Phi_N \text{diag}(s_N) z^2, \\ s^2 &= \mathbb{1}. \end{aligned}$$

The solution (s, z) of this system will give us both the flow values and the orientation of the network. Of course this would be really desirable since it removes all the previous discussion. The question then becomes whether this is practical: is the increase in computational time required to solve this expanded system worth it compared to checking $|\mathcal{S}|$ many smaller systems?

Remark 4.14. There is no need to consider the parametric signs s_i for all variables. According to the discussion above one may be able to fix some of the signs, and therefore reduce the size of R_s . This in turn will lead to a faster execution time.

4 Solving gas networks

The first thing one has to examine is the ordering of variables. The *inner* ordering is not really important (that is, whether $s_1 > s_2$ or viceversa). However, the choice of $s_i > z_j$ or $z_i > s_j$ does matter. We call these orderings $>_{df}$ and $>_{ef}$, respectively, meaning either *directions first* or *edges first*. What would each ordering mean?

- If using $>_{df}$, in the elimination process the last equations would be polynomials in z , and we would start with values for the flow before finding the direction of the gas. More specifically, one could imagine that we would get numerical values for the flow that would be valid for different orientations.
- If using $>_{ef}$, the last equations will involve only equations in s . Then the process of finding solutions will start by fixing more and more flow directions, and then finding the appropriate flows. This also means that the last equations, with fewer variables, may force some flow directions: for instance, some s_i would be fixed independently of the other directions.

For a very simple case, consider the smallest possible network with 3 nodes and 3 edges. Let the pressure drop coefficients be all 1, and set the loads of the nodes that are not the root to +1. The equation one has to solve is

$$f = -s_1 z_1^2 + s_2 z_1^2 - 6s_2 z_1 + 9s_2 + s_3 z_1^2 - 2s_3 z_1 + s_3,$$

and we add the conditions

$$s_1^2 - 1, \quad s_2^2 - 1, \quad s_3^2 - 1.$$

If we calculate the Gröbner basis of the ideal the results are as follows. For the $>_{df}$ ordering, one gets the system

$$\begin{aligned} g_1 &= s_1 - s_2 + \frac{53}{300}s_3 z_1^6 - \frac{53}{50}s_3 z_1^5 + \frac{109}{900}s_3 z_1^4 + \frac{1481}{225}s_3 z_1^3 - \frac{15559}{900}s_3 z_1^2 \\ &\quad + \frac{8363}{450}s_3 z_1 - \frac{427}{60}s_3, \\ g_2 &= s_2^2 - 1, s_2 z_1 - s_2 - \frac{1}{1200}s_3 z_1^6 + \frac{11}{120}s_3 z_1^5 - \frac{1613}{3600}s_3 z_1^4 - \frac{113}{300}s_3 z_1^3 \\ &\quad + \frac{10487}{3600}s_3 z_1^2 - \frac{629}{120}s_3 z_1 + \frac{49}{16}s_3, \\ g_3 &= s_3^2 - 1, \\ g_4 &= z_1^7 - 7z_1^6 + \frac{23}{3}z_1^5 + \frac{95}{3}z_1^4 - \frac{419}{3}z_1^3 + \frac{709}{3}z_1^2 - 205z_1 + 75 \end{aligned}$$

If we instead take the $>_{ef}$ ordering, we get

$$\begin{aligned} g_1 &= z_1^2 - 2z_1 s_1 s_3 - 2z_1 s_2 s_3 - 2z_1 - \frac{2}{3}s_1 s_2 + \frac{7}{3}s_1 s_3 + \frac{5}{3}s_2 s_3 + \frac{5}{3}, \\ g_2 &= s_1^2 - 1, \\ g_3 &= s_2^2 - 1, \end{aligned}$$

$$g_4 = s_3^2 - 1.$$

The first thing one notes is that a lot of the elements of the basis are simply the conditions. Note that the set of conditions by themselves forms a Gröbner basis of the ideal it generates: indeed,

$$\begin{aligned} S(s_i^2 - 1, s_j^2 - 1) &= \frac{s_i^2 s_j^2}{s_i^2} (s_i^2 - 1) - \frac{s_i^2 s_j^2}{s_j} (s_j^2 - 1) \\ &= s_i^2 - s_j^2. \end{aligned}$$

And if we apply the division algorithm to the S -polynomial, one gets the remainder zero. Therefore the important information for the Gröbner basis comes from the interaction of f and each condition. And if we come back to the idea of elimination, it makes all the sense that the conditions will appear untouched at the end, since any univariate polynomial in $I \cap K[s_3]$ will be a multiple of $s_3^2 - 1$.

Something similar happens if we instead consider a cycle with four elements. The polynomial we have to solve is

$$-z_1^2 s_1 + z_1^2 s_2 + z_1^2 s_3 + z_1^2 s_4 - 6z_1 s_2 - 2z_1 s_3 - 4z_1 s_4 + 9s_2 + s_3 + 4s_4,$$

to which we have to add the usual direction conditions. For the $>_{\text{df}}$ ordering we get a basis

$$\begin{aligned} g_1 &= s_1 - s_2 - \frac{128}{2205} s_4 z_1^{10} + \frac{709}{833} s_4 z_1^9 - \frac{11401}{2142} s_4 z_1^8 + \frac{277699}{14994} s_4 z_1^7 - \frac{5592479}{149940} s_4 z_1^6 \\ &\quad + \frac{5302123}{149940} s_4 z_1^5 + \frac{81395}{4284} s_4 z_1^4 - \frac{449543}{4284} s_4 z_1^3 + \frac{1183597}{8330} s_4 z_1^2 - \frac{6999161}{74970} s_4 z_1 + \frac{45032}{1785} s_4, \\ g_2 &= s_2^2 - 1, \\ g_3 &= s_2 z_1 - \frac{3}{2} s_2 + \frac{11}{49} s_4 z_1^{10} - \frac{1501}{441} s_4 z_1^9 + \frac{39469}{1764} s_4 z_1^8 - \frac{5263}{63} s_4 z_1^7 + \frac{659273}{3528} s_4 z_1^6 \\ &\quad - \frac{779677}{3528} s_4 z_1^5 - \frac{23927}{3528} s_4 z_1^4 + \frac{1710137}{3528} s_4 z_1^3 - \frac{1433251}{1764} s_4 z_1^2 + \frac{53617}{84} s_4 z_1 - \frac{419}{2} s_4, \\ g_4 &= s_3 + \frac{3874}{765} s_4 z_1^{10} - \frac{3874}{51} s_4 z_1^9 + \frac{151075}{306} s_4 z_1^8 - \frac{92954}{51} s_4 z_1^7 + \frac{12328571}{3060} s_4 z_1^6 \\ &\quad - \frac{1594463}{340} s_4 z_1^5 - \frac{165485}{612} s_4 z_1^4 + \frac{2137693}{204} s_4 z_1^3 - \frac{2953803}{170} s_4 z_1^2 + \frac{2311491}{170} s_4 z_1 - \frac{379247}{85} s_4, \\ g_5 &= s_4^2 - 1, \\ g_6 &= z_1^{11} - \frac{33}{2} z_1^{10} + 120 z_1^9 - \frac{2025}{4} z_1^8 + 1336 z_1^7 - 2121 z_1^6 + 1338 z_1^5 + \frac{8595}{4} z_1^4 \\ &\quad - 6539 z_1^3 + 7839 z_1^2 - 4914 z_1 + 1323 \end{aligned}$$

4 Solving gas networks

And for the $>_{\text{ef}}$ ordering, one gets

$$\begin{aligned}
g_1 &= z_1^2 - \frac{3}{2}z_1s_1s_4 + \frac{1}{2}z_1s_2s_3 - 2z_1s_2s_4 - 3z_1 - \frac{9}{8}s_1s_2 + \frac{5}{16}s_1s_3 + \frac{41}{16}s_1s_4 \\
&\quad - \frac{17}{16}s_2s_3 + \frac{43}{16}s_2s_4 + \frac{1}{8}s_3s_4 + \frac{7}{2}, \\
g_2 &= z_1s_1s_2 - z_1s_1s_4 + z_1s_2s_3 - z_1s_3s_4 - \frac{3}{2}s_1s_2 + \frac{3}{2}s_1s_4 - \frac{3}{2}s_2s_3 + \frac{3}{2}s_3s_4, \\
g_3 &= z_1s_1s_3 - z_1s_1s_4 + z_1s_2s_3 - z_1s_2s_4 - \frac{3}{2}s_1s_3 + \frac{3}{2}s_1s_4 - \frac{3}{2}s_2s_3 + \frac{3}{2}s_2s_4, \\
g_4 &= s_1^2 - 1, \\
g_5 &= s_1s_2s_3 + s_1s_2s_4 - s_1s_3s_4 - s_1 + s_2s_3s_4 + s_2 - s_3 - s_4, \\
g_6 &= s_2^2 - 1, \\
g_7 &= s_3^2 - 1, \\
g_8 &= s_4^2 - 1.
\end{aligned} \tag{4.2}$$

The same phenomenon appears again, with the conditions appearing at the end, and in fact can be seen if we consider longer cycles.

The question is now: does this difference in Gröbner basis lead to an improvement in running time? From a theoretical point of view, the answer should be yes. If we start solving the system from the last equation and work our way upwards, as in the extension process, the first few equations one has to solve are trivial: all give either a $+1$ or a -1 . And once we have given values to some s_i , we arrive to the equations involving z variables which will be of a relatively low degree.

On the other hand, when the last equations are the ones related to the edge flows, those high-degree polynomials have to be solved first. In the last example we see that g_6 has degree 11, which means we have to start by finding up to 11 roots (which will potentially be real numbers) and then retrace the steps to solve for s_i . An added inconvenience is that the equations we have to solve to find the s_i are harder than the conditions (although they are still of low degree), and it is not trivial to get either $+1$ or -1 . In a worst case scenario, one could even have errors stemming from the computation of the roots of z_1 as floating point numbers leading to s_i not being either $+1$ or -1 , but some close values instead.

Remark 4.15. At the beginning of this section we mentioned that the naïve approach of trying the $2^{|A|}$ combinations of signs was not effective, and yet when choosing the $>_{\text{ef}}$ ordering that is more or less what we do in the extension step, by solving for the conditions first. This is not a contradiction however: with this method we start by finding a Gröbner basis for the extended system, and then we start trying combinations of directions. This means that the most time-consuming process happens only once, instead of $2^{|A|}$ times for all possible vectors s .

Indeed, experimental data supports that the choice of $>_{\text{ef}}$ is better: in table 4.3 the timings to solve several networks with the two different orderings are displayed. One can see that $>_{\text{df}}$ is a better choice. Further testing with an increased number of edges

Table 4.3. Timing comparison between for the solution of different networks using either $>_{df}$ or $>_{ef}$, with all coefficients set to 1. Average of 10 samples.

Network shape	Number of nodes	Number of cycles	Number of directions	Timing (s)	
				$>_{df}$	$>_{ef}$
	3	1	3	0.099111	0.095982
	6	1	6	1.476457	1.078483
	4	2	5	1.623969	1.292350
	5	2	6	239.9154	6.431786

supports this conclusion.

Once we have settled for the ordering, the next thing we look at is our approach to building a solution. Instead of simply computing the Gröbner basis, we will go one step ahead and also compute a triangular decomposition, as in section 3.3.

The question to answer is then why is this decomposition not useful in the previous cases. If we look at the systems arising only from the network (that is, without parametric s terms) we expect them to have the shape predicted by the Shape Lemma (see theorem 4.9). In particular, they would directly form a triangular set, and would allow for a fast calculation of all terms. But in our new networks this is no longer the case. For instance, in the system with one cycle and four edges, with Gröbner basis given in eq. (4.2), even once the values of (s_1, s_2, s_3, s_4) has been established, we are left with a system of three equations in z_1 . Although this case is trivial to solve since all equations will be of low degree, one can see that for a larger number of cycles this will create complications, and the process of simply testing solutions will not work as well. The introduction of an additional step for the calculation of triangular systems allows us to step over this inconvenience.

In table 4.4, a comparison of the time required to find the solutions to different networks can be found. In this case, the $>_{ef}$ order was used, and the solver used an additional step to build a triangular set after computing the Gröbner basis (the SINGULAR function used is `solve`, from Wenk and Pohl 2019). A comparison with table 4.3 shows that using triangular sets leads to a much better performance.

Unsurprisingly, the exponential nature of the Gröbner basis algorithms leads to a big increase in the timings whenever we either increase the number of nodes or the number of cycles. In this setting, more nodes usually means more directions, because if C is the set of fundamental cycles we have

$$|S| = |A| - |\mathcal{J}| = |A| - (|V| - 1) \implies |A| = |S| + |V| - 1$$

so $s \in K^{|A|}$ will be of higher dimension. Meanwhile, more cycles means more z values

4 Solving gas networks

Table 4.4. Timing comparison for the solution of different networks using Gröbner basis and triangular sets. Average of 10 samples.

Network shape	Number of nodes	Number of cycles	Number of directions	Timing (s)	
				$q = \mathbb{1}$	$q \sim U(0, 1)$
	3	1	3	0.053925	0.059505
	4	1	4	0.095676	0.126080
	5	1	5	0.269690	0.384535
	6	1	6	0.606041	14.55476
	4	2	5	0.627715	4.163850
	5	2	6	5.593009	230.7818
	6	2	7	114.6725	-
	4	3	6	583.7949	-

to solve. Due to the simple nature of the restrictions for s , namely $s_i^2 - 1 = 0$, the additional edges do not cause as big of a problem as the additional cycles.

The choice of the loads does also play an important role. For random loads, taken from a uniform distribution on the interval $[0, 1)$, the time it takes to find a solution is notably increased. For the last two graphs, no solution could be found in reasonable time (over an hour).

5 Network reduction

The previous chapter tells us that, while the Gröbner-based method to find the flows of a gas network is very powerful, it fails when the size starts increasing. In particular, for even moderate networks (for instance with four fundamental cycles), the resulting problem is intractable.

To solve this problem, we turn to the idea of network reduction. This means that, instead of dealing with the full system, we reduce the size to get to a smaller problem, of a size that is solvable for us. Once the coarsest network is solved, the flows obtained are fixed and used to solve the finer networks.

Our approach is based on a topological reduction, similar to Ríos Mercado et al. 2002 (although without looking at the optimization problem), so we simplify the network and not the underlying system of equations.

We will consider that we start from an instance of the problem, given by the tuple $(G, q^{\text{nom}}, \phi, p_r)$. Here, $G = (V^*, E)$ is the graph representing the geometry of the network, $q^{\text{nom}} \in (\mathbb{R} \cup \{-\infty\})^{|V^*|}$ is the set of loads at every node, $\phi \in \mathbb{R}^{|E|}$ is the set of pressure drops at every pipe and p_r is the reference pressure.

We have changed the definition of q^{nom} to allow for $-\infty$ values. The nodes that have this $-\infty$ load are understood to act as reservoirs. If only one node u has $-\infty$ load, then we can simply redefine the load to be

$$q_u^{\text{nom}} = - \sum_{v \neq u} q_v^{\text{nom}},$$

and then use u as the root r of the graph. For the case where many nodes u_1, \dots, u_k have $-\infty$ load, we explain in section 5.1.4 how to proceed. To be consistent, we assume that p_r gives us the pressure value of the gas at all entry points, so at all nodes that have initial $-\infty$ load.

For cases where all loads are known, we will only consider graphs where the edge connectivity $\lambda(G)$ is bigger than one. If the edge connectivity is 1, by the analysis in section 4.3 one can study the two resulting subgraphs separately.

5.1 Topological reductions

The first thing we have to describe is our method for reducing the size of the networks in consideration. There are roughly four steps:

1. Remove the parts of the network that can be deduced from the rest.

2. Identify appropriate *complex* subgraphs of the original network, and substitute them with something simpler.
3. Solve the simplified network.
4. Use the information obtained from the previous solution to find the gas flow through the subgraphs we simplified.

The first step, pruning, is detailed in section 5.1.1. The second step involves deleting both fundamental cycles (section 5.1.2) and long paths (section 5.1.3). The third step is simply applying the methods of chapter 4, but we also mention in section 5.1.4 how to deal with networks without fixed inputs. And for the fourth step, we can either solve the subgraphs directly or further reduce them by applying this same scheme.

5.1.1 Pruning

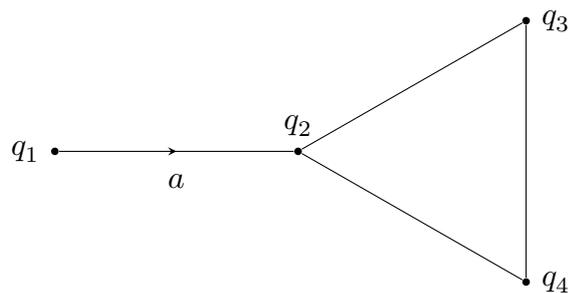
The first step of the process is removing all leaves. We saw before that trees have their directions uniquely determined, and we can just derive the flows directly from the values of q^{nom} . For example, let us assume that we have a leaf u , with an edge $a = (u, v)$. If $q_u^{\text{nom}} > 0$, then we know that

$$\vec{a} = (u, v), \quad Q_{0,a} = q_u^{\text{nom}}.$$

Therefore we can simply remove both u and a , and replace q_v^{nom} with $q_v^{\text{nom}} + q_u^{\text{nom}}$.

Starting from an instance $(G, q^{\text{nom}*}, \phi, p_r)$, we repeat this process until the resulting graph has no more leaves. The exception is supply nodes u with load $q_u^{\text{nom}} = -\infty$, that are considered separately in order to improve the precision of the system. After the pruning process, we arrive to an instance $(G_p, q_p^{\text{nom}*}, \phi, p_r)$. Note that the pressure drop coefficients and the root node pressure stay the same, and the only changes in $q^{\text{nom}*}$ are in the coefficients of nodes that were adjacent to a leaf that has been removed.

Example 5.1. Consider the following simple network:



We can remove the leaf formed by the node q_1 and the edge a , by redefining $q_2' = q_2 + q_1$. If $q_1 > 0$, the flow must go from against the direction of a , and we will have $Q_{0,a} = -q_1$. If $q_1 < 0$, the flow will follow a , and $Q_{0,a} = q_1$. In both cases, there is no error introduced by considering the reduced network.

5.1.2 Cycle collapse

In order to reduce the number of cycles of the network, which is directly related to the number of variables (and therefore to the complexity) of the system we want to solve, we *collapse* some of them. To do so, if G_c is the subgraph that defines the cycle that we want, we remove it and substitute it with a node u such that

$$q_u^{\text{nom}} = \sum_{v \in G_c} q_v^{\text{nom}}.$$

For every vertex of the remaining graph, if there was an edge between that vertex and G_c we add a vertex between the vertex and u .

There are some heuristics to take into consideration:

- It is possible to remove several cycles at the same time, and is in fact preferred. If the cycle to be removed shares an edge with another fundamental cycle, it is better to collapse both of them at the same time.
- Similarly, when collapsing cycles, one has to take into consideration the possibility of an external node v being connected to two different vertices of G_c . This would lead to two (u, v) edges, which is not desirable, and therefore should be avoided. However, note that for this situation to happen, we must have that v is already in a fundamental cycle containing an edge of G_c , and by the above should be collapsed too.
- It is preferable if all the loads of the vertices in G_c are known. This is the case in general, but should be noted nonetheless.

If $(G, q^{\text{nom}*}, \phi, p_r)$ is our instance of the graph, let $\mathcal{C} = \{ C_1, \dots, C_k \}$ be the set of cycles we want to remove. We contract all edges in the cycles of \mathcal{C} to obtain a graph $G_c = G/\mathcal{C}$, where “/” means edge contraction. Note that we take the convention that edge contraction will not create several edges between two nodes, so the result will not be a multigraph. We then arrive to an instance $(G_c, q_c^{\text{nom}*}, \phi, p_r)$ where the values of ϕ are unchanged but, for every cycle C_i , if u_i is the vertex to which all others collapse we let

$$q_{u_i}^{\text{nom}} = \sum_{v \in V(C_i)} q_v^{\text{nom}}.$$

We also get an additional k instances of problems of the C_i networks. The pressure drops are the same, but the cycles do not have to be balanced. Therefore we need additional information. So we instead consider the problems $C_i \cup \delta(C_i)$ where we add the neighbours, and then we take, for each $u \in \delta(C_i)$, the load

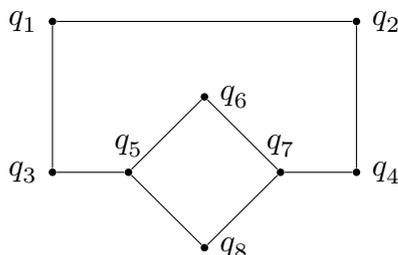
$$q_u^{\text{nom}} = - \sum_{\substack{a=(u, v_i) \\ v_i \in C_i}} Q_{0,a}.$$

This gives us the k modified instances $(C_i \cup \delta(C_i), q_{C_i}^{\text{nom}}, \phi, p_r)$.

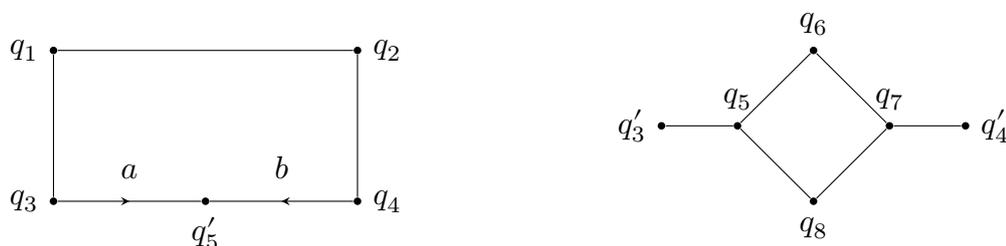
The underlying problem however is that, whenever the cycle has more than one entry point (meaning it is connected by pipes to more than one outside node) we will shorten

the paths between those nodes, and we have no compensation for the pressure drop that would happen when the gas crosses the cycle but will not be modeled, since there is no such thing as pressure drop at a node.

Example 5.2. We now look at the following network:



We want to collapse the cycle $C = (c_1, c_2, c_3, c_4)$. This leads to the following two networks:



We start by solving the left one, where $q'_5 = q_5 + q_6 + q_7 + q_8$. This gives us all the flows around the large cycle, including flows $Q_{0,a}$ and $Q_{0,b}$. We then set $q'_3 = -Q_{0,a}$ as well as $q'_4 = -Q_{0,b}$, and solve the subsystem of $C \cup \delta(C)$ to get the flows around this cycle. Once this is done, all flows around the network are established, and hence we have a solution.

5.1.3 Path simplification

Whenever we have a long path, it is desirable to shorten it. The reason is not that the computation of the flows becomes harder, since the number of cycles (which is the main factor in making the problem difficult to solve) remains unchanged. The improvement comes from the fact that it removes directions to consider.

As one could expect, if we have a path with vertices (u_1, \dots, u_n) and edges $a_i = (u_i, u_{i+1})$ for $1 \leq i \leq n - 1$, we can substitute it by considering instead the path (u_1, v, u_n) with

$$q_v^{\text{nom}} = \sum_{i=2}^{n-1} q_i^{\text{nom}}.$$

We then add edges $b_1 = (u_1, v)$ and $b_n = (v, u_n)$ to connect this extra vertex to the rest of the network.

The question arising now is how to define the pressure drop coefficients ϕ_{b_1} and ϕ_{b_n} . The problem is there is no good answer. If there were no entry node among the u_i and we knew the flow direction of each edge we could determine it. If all the flow goes straight from u_1 to u_k , and from u_n to u_k too, one has

$$\begin{aligned} p_1^2 - p_k^2 &= \sum_{i=1}^{k-1} p_i^2 - p_{i+1}^2 \\ &= - \sum_{i=1}^{k-1} \phi_{a_i} \left(Q_{0,b_1}^2 - \sum_{j=2}^i q_j \right)^2 \\ &= -\phi_{b_1} Q_{0,b_1}^2 \end{aligned}$$

which gives us

$$\phi_{b_1} = \sum_{i=1}^k \phi_k \left(1 - \sum_{j=2}^i \frac{q_j}{Q_{0,b_1}} \right)^2.$$

We will always have

$$Q_{0,b_1} \geq q_2 + q_3 + \dots + q_{k-1},$$

which means that the pressure drop coefficients in the simplified edge is the sum of the pressure drops for each pipe, multiplied by a factor that accounts for the reduced flow due to exits at every node. But still we have

$$\phi_{b_1} = \phi_{b_1}(Q_{0,b_1}),$$

so it is not really practical as we will not know the flow before needing the pressure drops. As a compromise, a (simple) possible choice may be taking

$$\phi_{b_1} = \sum_{i=1}^{n-1} \frac{\phi_{a_i}}{i}, \quad \phi_{b_n} = \sum_{i=1}^{n-1} \frac{\phi_{a_{n-i}}}{n-i}.$$

Once the coarse system has been solved, we get both directions and values. In particular, we know that

$$|Q_{0,a_1}| = |Q_{0,b_1}| \quad \text{and} \quad |Q_{0,a_{n-1}}| = |Q_{0,b_n}|,$$

and we also know their directions. This allows us to consider instead the path (u_2, \dots, u_{n-1}) with q_{u_2} being now $q_{u_2} \pm Q_{0,b_1}$ (the sign depending on the direction of the gas through b_1) and $q_{u_{n-1}}$ becoming $q_{u_{n-1}} \pm Q_{0,b_n}$. Then we simply have a tree, and we can apply the pruning procedure to find all the flows and directions.

To sum up, we have an instance $(G, q^{\text{nom}*}, \phi, p_r)$ and a set $\mathcal{P} = \{P_1, \dots, P_k\}$ of paths that we want to simplify. If $P = (u_1, \dots, u_n) \in \mathcal{P}$ we define $P^{\text{int}} = (u_2, \dots, u_{n-1})$, as well as

$$\mathcal{P}^{\text{int}} = \{P_i^{\text{int}} \mid 1 \leq i \leq k\}.$$

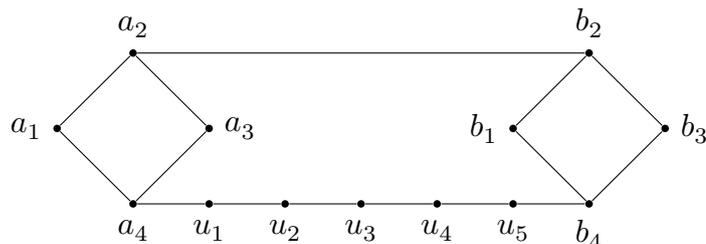
We then obtain a new instance by considering the graph $G_s = G/\mathcal{P}^{\text{int}}$. As in the case of the cycles one gets, for u_{P_i} a vertex representing P_i^{int} , that

$$q_{u_{P_i}}^{\text{nom}} = \sum_{v \in P_i^{\text{int}}} q_v^{\text{nom}}.$$

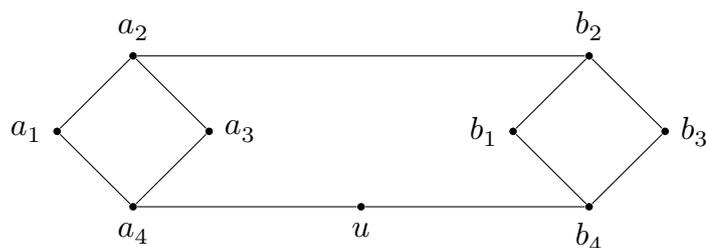
With respect to the pressure drop coefficients, we get a new vector ϕ_p . The edges that have been left untouched keep the same value of ϕ , and for the edges in the paths that have been collapsed, we choose some new values. This choice is specific to the problem. We therefore get the new instance $(G_p, q_p^{\text{nom}}, \phi_p, p_r)$.

As before, this process creates k additional subproblems $(P_i, q_{P_i}^{\text{nom}}, \phi, p_r)$. For any of these problems, ϕ stays the same. If $P_i = (u_{i,1}, \dots, u_{i,n})$, the loads are unchanged for the values in the middle. For the extremes, we get $q_{u_{i,1}} = -Q_{0,a}$, for $\vec{a} = (u_1, u_{P_i})$ in the reduced instance, and similarly for $q_{u_{i,n}}$.

Example 5.3. Consider the following network:



We want to collapse the path $P = (a_4, u_1, \dots, u_5, b_4)$. We do so by considering an edge in the middle and getting



The issue then becomes the choice of ϕ for the (a_4, u) and (u, b_4) edges – any of the strategies mentioned before could be attempted, depending on the knowledge we have about how we can expect the gas to flow through the system. Once this smaller system is solved, it is trivial to solve the path since there is only one possible direction for the flows.

5.1.4 Dealing with unknown gas inputs

Up until now we've been working with balanced networks, meaning that as much gas is injected into the system as is taken out. In practical terms, however, this does not have to be the case. For instance, one could assume that the exit loads q_+^{nom} are not fixed, but instead drawn according to some distribution X , which could depend on many factors (like weather or time of the day). And to satisfy that demand there would be gas injected at different entry points and at a fixed pressure. These entry points would act as reservoirs, and we assume they always have enough capacity to deal with the demand.

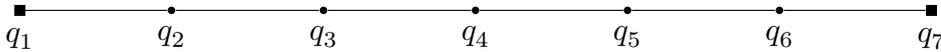
Remark 5.4. The opposite case is also possible: a gas network could have a several tanks distributed throughout the network to act as reservoirs and store extra gas whenever the demand is smaller than the compromised amount of gas entered into the network.

To know how much gas is injected into the system at each entry point, we define an additional artificial entry node r , which we will set as the root. For each of the entry nodes u_1, \dots, u_n of the original system we add a pipe $\vec{a}_i = (r, u_i)$. All the gas will enter the system at r and will then be distributed as needed to the entry points. Since these are *virtual pipes*, we let $\phi_{a_i} = 0$.

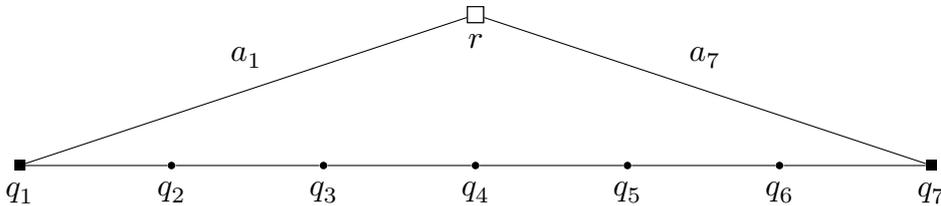
Remark 5.5. The choice of $\phi_{a_i} = 0$ is the only one that makes sense, since those pipes do not exist in reality and any other choice would mean that in a way we give priority to one of the reservoirs (the one with smallest pressure drop). Because of this it's important not to remove leaves containing entry nodes of unknown load in the pruning step. If one does that, what happens is that entry nodes become isolated, and pipes linking them become inactive. This makes sense: the reason why all the demand of a single node would not come from the closest supply is that bigger flows increase the pressure drop quadratically, and therefore it is more convenient to balance the flows to prevent such losses. But if we set the pressure drop to zero, suddenly the best option becomes to have exit nodes connected to only one entry.

Once we get a solution for this modified instance, we can come back to the original problem by setting $q_{u_i} = -|Q_{0,a_i}|$. The gas flow from the root r will be in a way optimized.

Example 5.6. Considering a network consisting only of one long path of pipes and nodes, as below:



In the depicted network, the squares represent the source nodes with unknown loads (so that $q_1 = q_7 = -\infty$), while all nodes in the middles are sinks. One could simply supply everything from one of the nodes and cut off the other, but that would mean that a lot of pressure would be lost after the gas travels through the entire network, so it is a better option to supply from both endpoints. To find how much gas should come from each source, we consider the network



where the upper node has been artificially added, with load

$$q_r = - \sum_{i=2}^6 q_i$$

so that the overall network is now balanced when we set $q_1 = q_7 = 0$. We solve the network with pressure drop coefficients $\phi_{a_1} = \phi_{a_7} = 0$, with the others the same as in the original network, and get values Q_{0,a_1} and Q_{0,a_7} . Then, looking at where the gas flow comes from, we know that, in the solution to the original network, one must have

$$q_1 = -|Q_{0,a_1}|, \quad q_7 = -|Q_{0,a_7}|.$$

5.2 Node pressures

Although the complicated part of solving a gas network is the calculation of the flows, once this is done it is also important to have a method to compute the associated pressures at all the nodes. If we know the value of the pressure p_r at a reference node, we can compute the rest of the pressures in a trivial way, spreading it arc by arc. By the definition of the system whose solution we compute, the values of the pressure computed that way will be consistent, meaning that calculating the pressure of a node by following two different paths to the reference node will lead to the same result.

A more mechanical result can be found in Nitsche 2018; Gotzes, Heitsch, et al. 2016.

Theorem 5.7. For a network G , define the functions F and g as in theorem 2.6. Let \hat{z} be the solution of the system $F(z) = 0$ arising from the network, and let p_r be the pressure of the reference node chosen in the computation of the system. Then

$$p_{u_i} = \sqrt{p_r^2 - g_i(q^{\text{nom}}, \hat{z})}.$$

Proof. Going back to eq. (2.10) we have

$$p^2 = \mathbb{1}p_r^2 - (\mathcal{A}_B^\top)^{-1}\Phi_B|Q_{0,B}|Q_{0,B}.$$

But theorem 2.6 implies that

$$(\mathcal{A}_B^\top)^{-1}\Phi_B|Q_{0,B}|Q_{0,B} = g(q^{\text{nom}}, \hat{z}),$$

hence going element by element we get that

$$p_{u_i}^2 = p_r^2 - g_i(q^{\text{nom}}, \hat{z}),$$

which is equivalent to what we wanted. \square

This allows us to very simply calculate the pressures at every node just by looking at g . However, the definition of g refers to the system generated by the entire network, and the solution z we have may not be an exact solution to that system (as it comes from a reduction process that could introduce errors).

Therefore, we need a system that is adequate to our construction of the solution, and takes into consideration the fact that we may be working with several sources that load gas into the network at a constant pressure. To solve this problem we define the following algorithm.

Algorithm 4 Network pressure calculation algorithm.

Require: $G = (V(G), E(G))$ **Require:** $p = (p_{u_1}, \dots, p_{u_{|V(G)|}})$ **Require:** $V = (Q_{0,a_1}, \dots, Q_{0,a_{|E(G)|}})$

```

1:  $r(u) \leftarrow \{ \emptyset \mid u \in V(G) \}$ 
2: while  $G \neq \emptyset$  do
3:    $\mathcal{S} \leftarrow \{ \text{sources of } G \}$ 
4:   for  $u \in \mathcal{S}$  do
5:     if  $p(u) = -\infty$  then
6:        $p(u) \leftarrow (\sum_{(Q,p) \in r(u)} pQ) / (\sum_{(Q,p) \in r(u)} Q)$ 
7:     for  $a = (u, v) \in \delta^+(u)$  do
8:        $r(v) \leftarrow r(v) \cup (V(a), p(u))$ 
9:        $E(G) \leftarrow E(G) - \{a\}$ 
10:     $V(G) \leftarrow V(G) - \{u\}$ 
11: return  $p$ 

```

Some things should be noted. First, we require p , which is also the output. But for the algorithm to work, all that is needed is to have the values of p for the sources of the graph G , and set $p_{u_i} = -\infty$ otherwise. Also note that the definition of source is a vertex with no incoming edges, meaning that a disconnected node is considered a source.

The idea behind the algorithm is as follows. Starting from a source u , the volumetric flow Q and starting pressure p of the gas through an edge $\vec{a} = (u, v)$ arriving at v is added to $r(v)$. Then the edge is deleted and, once all outgoing edges from u are considered, the node u itself is deleted.

After this process finishes, the resulting graph will have some new sources. At those new sources we compute the pressure as

$$p(u) = \frac{\sum_{\vec{a}=(v,u) \in \delta^-(u)} Q_{0,a} p_v}{\sum_{\vec{a}=(v,u) \in \delta^-(u)} Q_{0,a}}.$$

To arrive to this formula we consider the gas to behave like an ideal gas, and therefore to follow the ideal gas law $pV = nRT$. Assume that at node u we get volume V_1 at pressure p_1 from a pipe and volume V_2 at pressure p_2 at another. Then

$$\begin{aligned} p_u &= \frac{nRT}{V_u} = \frac{(n_1 + n_2)RT}{V_1 + V_2} = \frac{(p_1 V_1 / RT + p_2 V_2 / RT) RT}{V_1 + V_2} \\ &= \frac{p_1 V_1 + p_2 V_2}{V_1 + V_2}. \end{aligned}$$

A trivial generalization leads to the representation of $p(u)$ as the arithmetic mean of the incoming pressures weighted by the flows. In practice this is simply an application of Dalton's law, see Tschoegl 2000.

Once the new pressures are calculated, new partial pressures are added to the nodes connected to these sources. Repeating this process allows us to compute the pressure at every node, as we wanted.

Remark 5.8. While this algorithm is very simple (there are no smart choices for the edges to take as one would see in a Dijkstra-type algorithm, for example), it is not too complex. Indeed, the first WHILE loop gives at most $\mathcal{O}(|V|)$ steps (since every loop removes at least a vertex), the first FOR has at most $\mathcal{O}(|V|)$ sources and the last FOR is bounded by $\mathcal{O}(\min(|V|, |E|))$, giving a total of

$$\mathcal{O}(|V|^2 \min(|V|, |E|)).$$

In practice however this is far from the actual runtime. If the graph is very connected, there will not be many steps before termination (in fact the number of steps is bounded by the longest path starting at a source). And similarly, if there are many steps, it's because at each step \mathcal{S} is small. One could analyze the algorithm in more detail, but in practical terms this is much faster than the limiting factor, which is the calculation of the flows.

5.3 Improving the restrictions

Once we have finished a full round of the network reduction scheme, we obtain a vector Q_0 of flows through the original network. However, there is no guarantee that this vector is actually a solution of the original problem we had posed (and we should not expect it to be). The reason is, as mentioned before, the errors introduced by the elimination of certain pressure drops.

This does not mean that the computed solution is useless. In general, one should expect it to be close to the actual solution, and we can exploit that by using the knowledge gained in order to improve our choices when reducing and solving the network. This is similar to how multigrid methods work (see, for instance, Trottenberg, Oosterlee, and Schüller 2001). When working with W - and F -cycles, one starts with the coarsest possible mesh, solves it exactly, and goes on to restrict it to a finer one. However, after that, one uses the values of the finer mesh as a starting value for another round of resolution of the coarse mesh.

The question is, how can we apply this idea in our case? There are two aspects:

1. Once we have solved a particular network for some values q^{nom} of the loads, one could expect that if we try to solve a network for loads given by $q^{\text{nom}} + \varepsilon$, where ε is *small*, the flow would be oriented in a similar way. Here, *small* could mean that we take $\varepsilon \sim \mathcal{N}(0, \Sigma)$ with $\max(|\Sigma_{ij}|) \ll \max(|q_i^{\text{nom}}|)$, so that the new loads are in the order of magnitude of the previous ones, with just some minor perturbations.

Therefore, once we have solved one network, a majority of the flow directions could be fixed, in particular those that are untouched in the reduction process. Of course this has limitations: for instance, one should not fix the direction \vec{a} of a pipe a if $Q_{0,a} < \min(|\Sigma_{ij}|)$.

2. The other improvement one can make once the finer model has been established is related to the choice of pressure drop coefficients ϕ for the coarse model. As seen in section 5.1.3, the more we know about how the gas will flow through a path, the better our definition of ϕ will be for the pipe that acts as substitute of that path.

The latter point allows us to consider a procedure based on iterated V -cycles (keeping the multigrid terminology). This is illustrated in fig. 5.1, and a high-level description of the algorithm can be found in algorithm 5.

Algorithm 5 Iterated V -cycles algorithm for network reduction.

Require: $(G, q^{\text{nom}}, \phi, p_r)$

- 1: $G_p \leftarrow G - \{ \text{leaves of } \vec{G} \}$ ▷ Stage p
- 2: **repeat**
- 3: $\mathcal{C} \leftarrow \{ \text{cycles to collapse} \}$
- 4: $G_c \leftarrow G_p / \mathcal{C}$ ▷ Stage c
- 5: $\mathcal{P} \leftarrow \{ \text{paths to simplify} \}$
- 6: $G_s \leftarrow G_c / \mathcal{P}^{\text{int}}$ ▷ Stage s
- 7: **if** $q_i^{\text{nom}} = -\infty$ for some i **then**
- 8: $G_R = (V(G_s) \cup \{r\}, E(G_s) \cup \{ (r, q_i) \mid q_i^{\text{nom}} = -\infty \})$
- 9: **else**
- 10: $G_R = G_s$
- 11: $Q_{0,R} \leftarrow \text{SOLVE}(G_R, q_p^{\text{nom}}, \phi_p)$ ▷ Stage R
- 12: **for** $P \in \mathcal{P}$ **do**
- 13: $q_P^{\text{nom}} \leftarrow \text{loads from } Q_{0,R}$ ▷ Stage i
- 14: $Q_{0,P} \leftarrow \text{SOLVE}(P, q_P^{\text{nom}}, \phi)$
- 15: $Q_{0,c} \leftarrow \text{combined flow from } Q_{0,R} \text{ and the } Q_{0,P}$
- 16: **for** $C \in \mathcal{C}$ **do**
- 17: $q_{C_i}^{\text{nom}} \leftarrow \text{loads from } Q_{0,c}$ ▷ Stage i
- 18: $Q_{0,C} \leftarrow \text{SOLVE}(C \cup \delta(C), q_C^{\text{nom}}, \phi)$
- 19: $Q_{0,p} \leftarrow \text{combined flows from } Q_{0,c} \text{ and the } Q_{0,C}$ ▷ Stage Q
- 20: **until** criterion is met
- 21: $Q_0 \leftarrow \text{combined flows from } Q_{0,p} \text{ and the leaves}$ ▷ Stage \bar{p}
- 22: $p \leftarrow \text{CALCULATEPRESSURES}(G, Q_0, p_r)$ ▷ Stage P
- 23: **return** $Q_{0,p}$

We start from a graph G , part of an instance $(G, q^{\text{nom}}, \phi, p_r)$. The first step, p , simply means pruning all leaves from the graph, since as mentioned they do not add any additional information.

Steps c and s involve the choice of a set \mathcal{C} of cycles first and a set \mathcal{P} afterwards that we want to collapse. We then arrive to a reduced graph G_s , and from it we define G_R , the final reduced graph, by adding an extra vertex if necessary to find an unknown load. We then SOLVE this graph exactly using the methods of chapter 4: this is step R in the cycle.

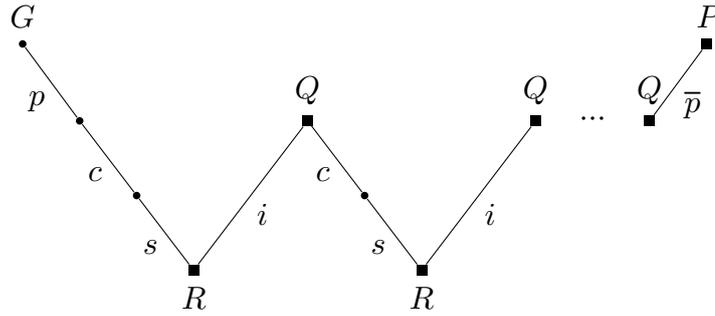


Figure 5.1. The iterated V -cycle for the network reduction.

Once we have a solution $Q_{0,R}$, we start solving the $|\mathcal{P}| + |\mathcal{C}|$ problems generated by the reduction, using the information obtained from the step before. We start with paths, since the results may be needed to solve some of the cycles. We then **combine** all partial solutions obtained to define $Q_{0,p}$.

At this point the cycle is finished, and a decision has to be made about whether to perform another cycle or to finish using this computed $Q_{0,p}$ flow. A possible choice is computing the mean squared error as in section 4.2.2 for the solution $Q_{0,p}$ of the whole network, and requiring it to be below a certain threshold. Other criteria can be defined depending on the problem.

The other option is to continue with an additional cycle. The difference between each cycle is in the choice of the reductions of c and s . As mentioned before, the clearest example would be doing the same topological reduction but changing the associated pressure drop coefficients of the reduced network to account for the knowledge, stemming from the calculation of $Q_{0,R}$, of the direction of flow along a path. Similarly, in the second cycle many directions could be ignored when solving the networks (in particular one could assume that the directions that worked in the first cycle will be the ones that work for the entire network, and set them for all cycles). However more complex differences are possible: for instance one may decide to keep a whole cycle untouched in c , or collapse an additional path in s . In general, the value of Q is completely independent (in a numerical sense) of the previous value of Q .

Finally, once we are happy with our calculated flows $Q_{0,p}$ (which can be because some parameters reach a fixed point, or the difference between computed flows goes below a certain threshold), all that is left is to include the leaves removed at the beginning and calculate the pressure as shown in section 5.2, with `CALCULATEPRESSURES` being algorithm 4. This gives us the full description of the network, as we wanted.

Remark 5.9. It would be possible to perform more drastic reductions in our cycles. For instance, if we have a pipe that actually has a control valve, and the calculated flow in the first cycle clearly exceeds the maximum safe operational flow of the valve, one could remove the pipe directly for the next cycles.

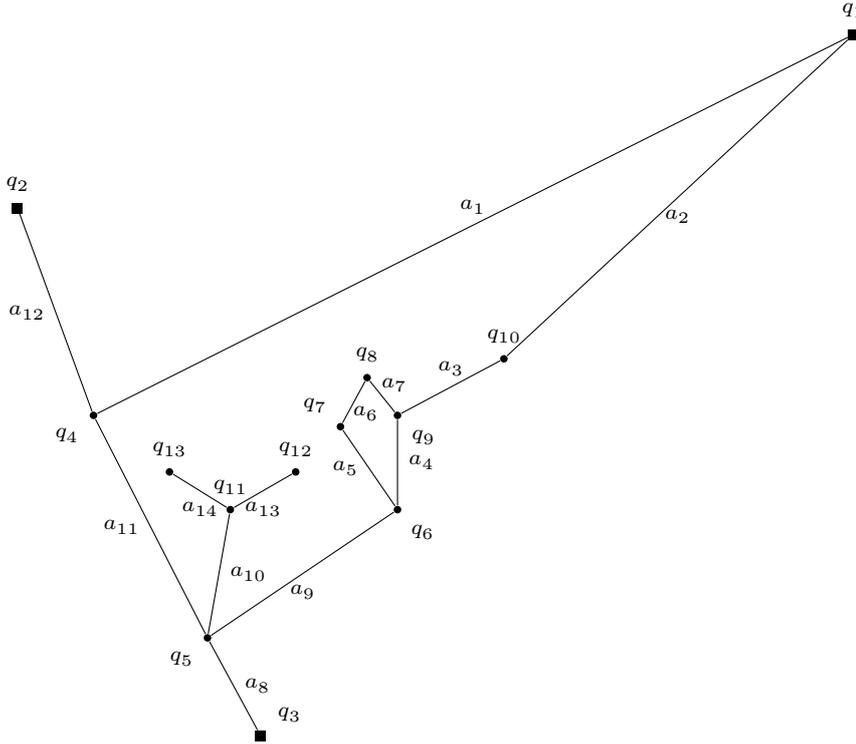


Figure 5.2. Schematic model of the reduced Irish gas network.

5.4 The Irish gas network

We now turn to a concrete example of how one can apply the network reduction technique to a slightly bigger network (with 13 nodes and 14 edges). We will throughout this section consider a simplified version of the Irish gas network, based on Ekhtiari et al. 2019. A representation can be seen in fig. 5.2. There are three entry nodes q_1 , q_2 and q_3 , corresponding to the Moffat interconnector, the Corrib gas field and the Kinsale Head/Seven Heads gas fields, respectively.

The first step to reduce the network is pruning, that is, removing leaves with fixed load. We therefore remove the nodes q_{11} , q_{12} and q_{13} . While q_{11} by itself is not a leaf, it becomes one when we remove the other two, and therefore can also be removed without problems. The only modification one needs to do to the remaining network is setting

$$q'_5 = q_5 + q_{11} + q_{12} + q_{13}.$$

Next, we remove the $C = (q_6, q_7, q_8, q_9)$ cycle, turning it into a single node between q_5 and q_{10} . This means that in a sense q_5 and q_{10} are now *closer*, as the pressure drop through both possible arcs connecting them, (a_9, a_4, a_3) and $(a_9, a_5, a_6, a_7, a_3)$, is reduced due to the loss of the pressure drops associated to the edges we have collapsed together. We define

$$q_C = q_6 + q_7 + q_8 + q_9.$$

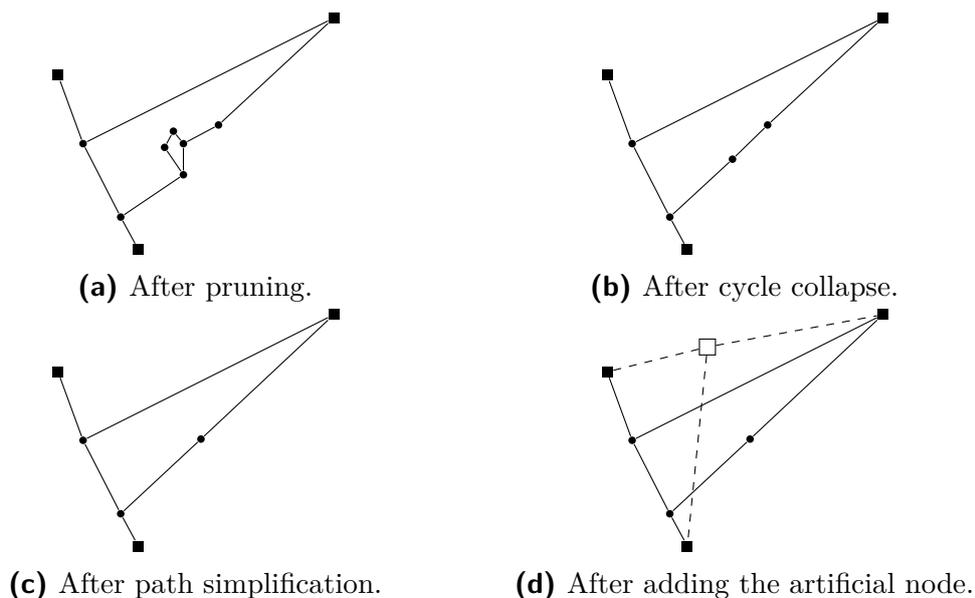


Figure 5.3. Different steps of the network reduction method.

The last remaining simplification of the Irish network is removing the two nodes in the path linking q_1 and q_5 , including the one we have just created by collapsing the cycle C . We can now choose the values for the pressure drops. If we knew, for instance, that gas coming from q_1 would only reach q_{10} , we could define

$$\phi_{b_1} = \phi_{a_2}, \quad \phi_{b_n} = \phi_{a_9} + \phi_{a_3} \left(1 - \frac{q_6 + q_7 + q_8 + q_9}{q_5 + q_6 + q_7 + q_8 + q_9} \right)^2.$$

However the most general scenario would be not having any such information. Then we should simply do

$$\phi_{b_1} = \phi_{a_2} + \frac{1}{2}\phi_{a_3}, \quad \phi_{b_n} = \phi_{a_9} + \frac{1}{2}\phi_{a_3}.$$

Finally, one has to take care of the three supply nodes, q_1 , q_2 and q_3 . As explained before, we simply define an additional artificial node connected to all of them, and set the pressure drop along the connections to be zero.

The evolution of this process appears in fig. 5.3.

We now turn towards solving the network. The coefficients are taken from Ekhtiari et al. 2019. The network parameters are

$$q^{\text{nom}} = (-\infty, -\infty, -\infty, 40, 30, 10, 5, 5, 5, 60, 10, 8, 7)$$

as well as

$$\begin{aligned} \phi_{a_1} &= 2021152507, & \phi_{a_8} &= 272631999, \\ \phi_{a_2} &= 3825126631, & \phi_{a_9} &= 766010664, \end{aligned}$$

$$\begin{aligned}
\phi_{a_3} &= 603344912, & \phi_{a_{10}} &= 710214368, \\
\phi_{a_4} &= 180950610, & \phi_{a_{11}} &= 1636367083, \\
\phi_{a_5} &= 241253384, & \phi_{a_{12}} &= 450972262, \\
\phi_{a_6} &= 236922319, & \phi_{a_{13}} &= 196766854, \\
\phi_{a_7} &= 140398411, & \phi_{a_{14}} &= 170531273.
\end{aligned}$$

The value of $\phi_{a_{14}}$ does not appear in the original article, so we have extrapolated it from $\phi_{a_{13}}$ and the physical conditions of the pipe.

A more detailed representation of the reduced network can be seen in fig. 5.4. Most of the structure of the network is intact, and in fact the only problems appear in the arc linking q_1 to q_5 via q_{10} . For this network, the node loads we have are

$$\begin{aligned}
q_{r,1} &= q_1 = 0, \\
q_{r,2} &= q_2 = 0, \\
q_{r,3} &= q_3 = 0, \\
q_{r,4} &= q_4 = 40, \\
q_{r,5} &= q_5 + q_{11} + q_{12} + q_{13} = 50, \\
q_{r,6} &= q_6 + q_7 + q_8 + q_9 + q_{10} = 85,
\end{aligned}$$

and for the pressure drop coefficients we get

$$\begin{aligned}
\phi_{a_{r,1}} &= \phi_{a_{r,2}} = \phi_{a_{r,3}} = 0, \\
\phi_{a_{r,4}} &= \phi_{a_1}, \\
\phi_{a_{r,5}} &= \phi_{a_8}, \\
\phi_{a_{r,6}} &= \phi_{a_{12}}, \\
\phi_{a_{r,7}} &= \phi_{a_{11}}, \\
\phi_{a_{r,8}} &= \phi_{a_9} + \phi_{a_3}/2, \\
\phi_{a_{r,9}} &= \phi_{a_2} + \phi_{a_3}/2.
\end{aligned}$$

As we mentioned in section 5.1.3, the choice of pressure drop coefficients is quite arbitrary at this point. One also has to take into consideration the possible flow directions. The previous figure represents the arcs already oriented with respect to a depth-first search tree. The vector of directions is

$$s = (+1, +1, +1, s_{a_{r,4}}, +1, -1, s_{a_{r,7}}, s_{a_{r,8}}, s_{a_{r,9}}).$$

This gives us a maximum of $2^4 = 16$ possible directions to check. However some of them can be removed, since we know that $s_{a_{r,8}} = -1, s_{a_{r,9}} = 1$ is not a valid direction (there would be no flow towards $q_{r,6}$, which is a sink), and that removes $2^2 = 4$ possibilities.

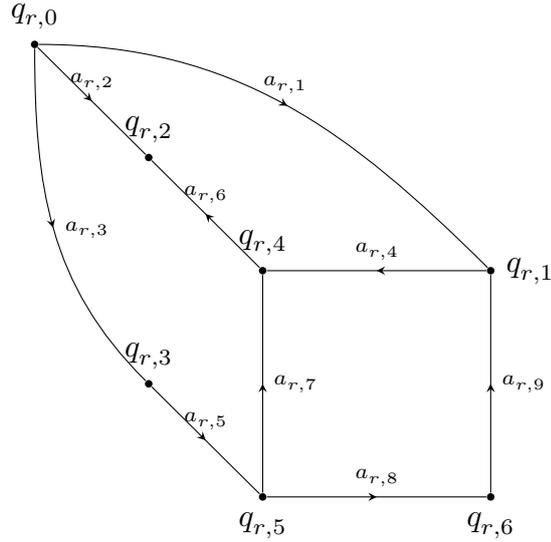


Figure 5.4. The detailed reduced network.

Similarly we can't have a cycle being formed, which rules two more directions, giving a total of 10 directions to check.

Once this system is solved, we know for the original that

$$q_1 = -Q_{0,a_{r,1}}, \quad q_2 = -Q_{0,a_{r,2}}, \quad q_3 = -Q_{0,a_{r,3}}.$$

The only remaining computation is the (q_6, q_7, q_8, q_9) cycle. We solve it using the parametric directions system. The pressure drops stay the same, but the loads are changed as

$$\begin{aligned} q_{c,6} &= q_6 - s_{a_{r,8}} Q_{0,a_{r,8}}, \\ q_{c,7} &= q_7, \\ q_{c,8} &= q_8, \\ q_{c,9} &= q_9 + q_{10} + s_{a_{r,9}} Q_{0,a_{r,9}}. \end{aligned}$$

After this step, all pipe flows can be easily determined. We obtain the vector of flow directions

$$s = (+1, +1, +1, +1, +1, -1, -1, +1, -1).$$

At this point we are, in the terms of the cycle described in section 5.3, at the first Q point. There are two choices: either stopping, calculating the pressures and finishing or trying to refine our calculation. We will do the latter.

The results of table 5.1, in the column of the flows after cycle 1 show that q_1 only supplies (partially) the q_{10} node. Therefore, we can work under that assumption, and redefine

$$\phi_{a_{r,9}} = \phi_{a_2}.$$

Table 5.1. Network flow in the Irish network computed several ways: our reduction method at the first and at the last cycle, and the three methods (SAINT custom solver, MATLAB numerical solver and their own algorithm) reported in Ekhtiari et al. 2019, Table 4. All values in $\text{m}^3 \text{s}^{-1}$.

Pipe	Cycle 1	Cycle 8	SAINT	MATLAB	EDLS
Q_{0,a_1}	20.57	20.63	21.62	18.99	18.89
Q_{0,a_2}	34.44	33.77	34.83	34.53	34.23
Q_{0,a_3}	25.55	26.22	25.16	25.46	24.53
Q_{0,a_4}	23.98	24.41	23.78	23.73	22.39
Q_{0,a_5}	16.57	16.81	16.38	16.73	16.22
Q_{0,a_6}	11.57	11.81	11.38	11.73	11.64
Q_{0,a_7}	6.57	6.81	6.38	6.73	6.63
Q_{0,a_8}	81.43	81.90	85.80	88.13	88.11
Q_{0,a_9}	50.55	51.22	50.16	50.46	50.39
$Q_{0,a_{10}}$	25.00	25.00	25.00	25.00	25.00
$Q_{0,a_{11}}$	24.12	24.32	29.35	27.33	27.10
$Q_{0,a_{12}}$	43.55	43.68	37.73	38.34	38.03
$Q_{0,a_{13}}$	8.00	8.00	8.00	8.00	8.00
$Q_{0,a_{14}}$	7.00	7.00	7.00	7.00	7.00

What is now missing is the definition of $\phi_{a_{r,8}}$. While we could try and make a heuristic definition, a better choice is defining it directly from the data we have. As the flow will go through the path (a_9, a_4, a_3) , we can let

$$\phi_{a_{r,8}} Q_{0,a_{r,9}}^2 = \phi_{a_9} Q_{0,a_9}^2 + \phi_{a_4} Q_{0,a_4}^2 + \phi_{a_3} Q_{0,a_3}^2,$$

with the right hand side flows calculated from the solution we have put together. This leads to

$$\phi_{a_{r,8}} = \phi_{a_9} + \phi_{a_4} \frac{Q_{0,a_4}^2}{Q_{0,a_{r,8}}^2} + \phi_{a_3} \frac{Q_{0,a_3}^2}{Q_{0,a_{r,8}}^2}.$$

We get a value of $\phi_{a_{r,8}} = 966418684$ (we only take the integer part), which allows us to repeat the cycle to arrive to a new set of flows. Again we can either stop here or keep going. We choose to keep iterating and improving the value of $\phi_{a_{r,8}}$. The idea is that at some point this should become stable, and the expected value of the pressure drop coefficient will match the actual calculated value. This is indeed the case: after eight cycles, with pressure drops

$$\begin{aligned} \phi_{a_{r,8}}^{(0)} &= 1067683120, \\ \phi_{a_{r,8}}^{(1)} &= 960923719, \\ \phi_{a_{r,8}}^{(2)} &= 965454804, \end{aligned}$$

$$\begin{aligned}
\phi_{a_{r,8}}^{(3)} &= 965257843, \\
\phi_{a_{r,8}}^{(4)} &= 965266396, \\
\phi_{a_{r,8}}^{(5)} &= 965266025, \\
\phi_{a_{r,8}}^{(6)} &= 965266041, \\
\phi_{a_{r,8}}^{(7)} &= 965266040,
\end{aligned}$$

the system reaches a *fixed point*, meaning that $\phi_{a_{r,8}}^{(8)} = \phi_{a_{r,8}}^{(7)}$. Note that other fixed points are possible too, and are reached depending on the run (sometimes it even enters a loop between two values), although all computed flows are equal up to a high degree of precision. Therefore we stop at this iteration. We can now assume that our reduction of the network is as precise as possible, and the calculated flows are the actual solution of the original network. The flows calculated at the beginning and the end of the process can be found in table 5.1.

Once the flows are established, what remains is the computation of the pressure at every node. We will use the algorithm in section 5.2, with starting pressures

$$p_1 = p_2 = p_3 = 7.0 \times 10^5 \text{ Pa.}$$

The resulting values for the pressures appear in table 5.2. One can now check how accurate our flows are. If the calculation were completely precise, then the pressure at every node could be calculated with any path starting at a source, and there would be no reason to deal with partial pressures in the algorithm. In our case, four nodes will have gas coming from two different nodes: q_4 , q_5 , q_9 (the two ways to go around the cycle) and q_{10} . The partial pressures are

$$\begin{aligned}
p_4 &= 6.938\,246\,066\,449\,20 \times 10^6 \text{ Pa} & \text{and} & & p_4 &= 6.938\,246\,066\,449\,20 \times 10^6 \text{ Pa}, \\
p_5 &= 6.868\,118\,614\,236\,92 \times 10^6 \text{ Pa} & \text{and} & & p_5 &= 6.868\,118\,614\,236\,92 \times 10^6 \text{ Pa}, \\
p_9 &= 6.712\,140\,011\,377\,00 \times 10^6 \text{ Pa} & \text{and} & & p_9 &= 6.712\,140\,012\,269\,95 \times 10^6 \text{ Pa}, \\
p_{10} &= 6.681\,146\,697\,585\,50 \times 10^6 \text{ Pa} & \text{and} & & p_{10} &= 6.681\,146\,697\,750\,05 \times 10^6 \text{ Pa}.
\end{aligned}$$

The fact that the computed partial pressures are so similar (in the case of p_4 and p_5 up to 15 digits of precision) suggests that our final model is indeed really precise.

Although we have arrived at a solution that is precise, an important aspect that cannot be overshadowed is the timing: an accurate solution that takes too long to be found is not useful.

The timing of the complete calculation can be found in table 5.3, where the start and end times are given for each of the cycle, as well as the last step for computation of pressures. We also add the timings of the two main steps that involve solving systems with Gröbner methods: step R is solving the big reduced network, while step Q is solving the reduced network with four edges collapsed at the beginning. Additional time within each cycle was devoted to other processes such as redefining the pressure drops or finding the loads.

Table 5.2. Load and pressure at each node computed several ways: using algorithm 4, and the three methods (SAINT custom solver, MATLAB numerical solver and their own algorithm) reported in Ekhtiari et al. 2019, Table 3. Loads in $\text{m}^3 \text{s}^{-1}$, pressures in $\times 10^6 \text{ Pa}$.

Node	Load	Cycle 8	SAINT	MATLAB	EDLS
p_4	40	6.938	6.966	6.941	6.910
p_5	30	6.868	6.832	6.827	6.793
p_6	10	6.720	6.626	6.658	6.648
p_7	5	6.715	6.607	6.653	6.593
p_8	5	6.712	6.604	6.651	6.623
p_9	5	6.712	6.602	6.650	6.624
p_{10}	60	6.681	6.596	6.619	6.600
p_{11}	10	6.835	6.789	6.789	6.772
p_{12}	8	6.834	6.785	6.789	6.743
p_{13}	7	6.835	6.745	6.789	6.779

The times for the first cycle are very high due to the issue of directions. For R , we have to check a priori ten directions, which gives a major decrease in performance since we are forced to check them one by one until one of them gives a solution. One could try and do this in parallel, however this leads to problems when using SageMath (because although each direction starts running in a different thread, when they require interaction with SINGULAR they all connect to the same instance, and the commands sent start to overlap, giving errors). In the following calculations of R , the direction that was used for the solution in cycle 1 is used by default. Similarly, for the computation of step Q , since we do not know the solution either we apply the method of section 4.3.1, which allows us to obtain directly a set of directions that we will use in the next iterations. Since there are only a few directions to check, it would also be possible to make the process faster by simply running the algorithm in parallel. An even faster option would be realizing that the equation is simply a quadratic equation, so one could simply calculate and compute the algebraic solution in terms of the loads. However, for the sake of providing an approach using only the methods developed in the previous sections, we use the Gröbner approach.

For the cycles two to eight, the time it takes for the steps R and Q to finish is always at a similar level of around 0.08 s and 0.03 s, respectively. This is a relatively fast calculation, since it means that the total time spent calculating solutions in each cycle is around 0.12 s. The additional time spent could probably be reduced by optimizing the code as much as possible: for instance, there is always some time lost in interfacing between SageMath and SINGULAR, and on computing values that may not be used at the end.

One last thing to consider is that the solutions at the end of each cycle do not change much. In table 5.1 one can see that between the first cycle (with a wild guess for $\phi_{a_r,8}$ and $\phi_{a_r,9}$) and the last cycle (with the pressure drop at a fixed point) the variation of

Table 5.3. Timestamps for the execution of the iterated V -cycles algorithm for the Irish network, and timing for the calculation of the loads (at steps R and Q) and pressures (using algorithm 4).

Stage	Time (s)		Time for R (s)	Time for Q (s)
Cycle 1	0.000400	Start	0.619333	0.182528
	0.806086	End		
Cycle 2	0.879807	Start	0.084666	0.031643
	1.049376	End		
Cycle 3	1.049493	Start	0.082903	0.031463
	1.210725	End		
Cycle 4	1.210776	Start	0.081302	0.032587
	1.374903	End		
Cycle 5	1.375007	Start	0.080368	0.030595
	1.534096	End		
Cycle 6	1.534264	Start	0.086031	0.029402
	1.696714	End		
Cycle 7	1.696762	Start	0.082029	0.034126
	1.860312	End		
Cycle 8	1.860362	Start	0.084933	0.030509
	2.023596	End		
Step P	2.024733	Finished		

the values of the flow is relatively small. Even more can be said: if the results after the last cycle are considered the “baseline” results, by the end of the second cycle all flows are accurate to 3 digits, by the end of the fourth cycle all flows are accurate to 5 digits, and by the end of the fifth cycle all flows are accurate to 7 digits.

Finally, it can be seen that the computation of the pressures has a negligible impact on the overall performance of the algorithm, taking around a 1 ms.

Overall, while the performance is definitely worse than the one reported in Ekhtiari et al. 2019, Table 5 by at least an order of magnitude, improvements to this are possible. For instance, other runs in the same network for different loads (still similar to the ones that we are using) could profit from knowing already the directions to set and having a better approximation to the pressure drop. One could also precompute a Gröbner system for step Q to get a much faster solution (as mentioned before this would simply end up being a solution to a quadratic equation), and reduce the number of cycles being considered to obtain a less precise but still acceptable solution.

Once we have solved the network for the aforementioned values of q^{nom} , we can use the

knowledge of the solution to solve other similar networks faster. For instance, consider the initial exit loads

$$q_+^{\text{nom}} = (40, 30, 10, 5, 5, 5, 60, 10, 8, 7) \in \mathbb{R}^{10}.$$

This vector represents the demand for gas by end users, which we could model by a probability distribution. In particular, let us assume that q_+^{nom} is the *average* demand, and that the actual demand is given by a normal multivariate distribution centered on q_+^{nom} . We will define $\mu = q_+^{\text{nom}}$, as well as a matrix Σ as

$$\Sigma = (\sigma_{i,j})_{1 \leq i, j \leq 10} = \begin{cases} 0.05\mu_i, & \text{if } i = j, \\ 0.025\sqrt{\mu_i\mu_j}, & \text{if nodes } i \text{ and } j \text{ are connected,} \\ 0, & \text{otherwise.} \end{cases}$$

The above matrix gives us a variation in the loads of every node that is not one of the $-\infty$ source nodes. We can then consider a network where the source nodes are left unchanged but the loads of the at the exit nodes are now given by

$$q_+^{\text{nom}} \sim \mathcal{N}(\mu, \Sigma).$$

Remark 5.10. Of course, the choice of having q_+^{nom} follow a normal distribution is a simplification: the problem of forecasting the demand is much more complicated. But in any case we could always write the demand as $q_+^{\text{nom}} = \mu + \varepsilon$ with an error term stemming from some probability distribution $\varepsilon \sim X$, and with $\mathbb{P}_X(\varepsilon > \delta\mu) \ll 1$ for some $0 < \delta < 1$, so that we can expect the flow directions to stay the same.

To study this case, we consider 100 networks with exit loads taken from 100 samples of the normal distribution above. The construction of Σ is done to make the load q_u^{nom} of an exit node be close to the mean load, and to couple together loads for nodes that share an edge. For each sample of the normal multivariate distribution, we only take 5 decimal digits in every element, to improve the performance of the algorithm. For each network, we compute the solution after one cycle and after the calculated pressure drops $\phi^{(i)}$ converge (to prevent the network from entering a loop, we simply require $|\phi^{(i+1)} - \phi^{(i)}| \leq 1$). We then compute, for each network and each pipe a , the relative error

$$\eta_a = \left| \frac{Q_{0,a}^{(1)} - Q_{0,a}^{(i_{\max})}}{Q_{0,a}^{(i_{\max})}} \right|. \quad (5.1)$$

Table 5.4 shows the results of the relative errors of the 100 samples. Note that the cases for pipes a_{10} , a_{13} and a_{14} are not really interesting since they form a tree and so the flow can be calculated independently of the rest of the network. One can see that all of the pipes present a relative error of less than 1% on average, and only for one of the pipes was there a case where the relative error crossed the 1% threshold (for η_{a_7}). As one could expect, the largest relative errors appeared in the pipes linking q_1 and q_5 , which is where the network reduction takes place, and in particular in the pipes along the removed cycle.

Table 5.4. Statistical parameters (mean μ , standard deviation σ , minimum and maximum value) of the relative error η_a (as defined in eq. (5.1)) between the first and last computed flows across 100 sampled networks.

Pipe	μ	σ	min	max
Q_{0,a_1}	0.00037	0.00029	0.00007	0.00118
Q_{0,a_2}	0.00238	0.00188	0.00005	0.00756
Q_{0,a_3}	0.00306	0.00241	0.00006	0.00982
Q_{0,a_4}	0.00211	0.00166	0.00004	0.00660
Q_{0,a_5}	0.00173	0.00137	0.00003	0.00553
Q_{0,a_6}	0.00245	0.00192	0.00005	0.00763
Q_{0,a_7}	0.00424	0.00339	0.00008	0.01542
Q_{0,a_8}	0.00069	0.00055	0.00001	0.00220
Q_{0,a_9}	0.00157	0.00124	0.00003	0.00507
$Q_{0,a_{10}}$	0.00000	0.00000	0.00000	0.00000
$Q_{0,a_{11}}$	0.00098	0.00078	0.00002	0.00308
$Q_{0,a_{12}}$	0.00037	0.00029	0.00001	0.00118
$Q_{0,a_{13}}$	0.00000	0.00000	0.00000	0.00000
$Q_{0,a_{14}}$	0.00000	0.00000	0.00000	0.00000

Table 5.5. Statistical parameters (mean μ , standard deviation σ , minimum and maximum value) of the time required to compute the flows in one cycle or until convergence.

Stage	μ (s)	σ (s)	min (s)	max (s)
After one cycle	0.083	0.007	0.074	0.111
After convergence	0.533	0.051	0.417	0.656

In table 5.5 the times it took to compute the first and last cycle are shown. As we have more information about the network, several improvements were applied: the directions were fixed from the start, and for the computation of the flows through the small collapsed cycle a Gröbner system was calculated beforehand, which allowed us to simply substitute into the solution of a degree 2 equation. The starting value of $\phi_{a_r,s}$ was set to the fixed point calculated before. All of this together leads to a more streamlined approach that arrives to an approximately good solution in the first iteration and really fast. Of course more improvements would be possible, for instance by reducing the overhead cost of switching between SageMath and SINGULAR. Still, the average time of 0.083s is close to the 0.058s in Ekhtiari et al. 2019.

6 Conclusions

Throughout this thesis we have investigated how to use a Gröbner basis approach in order to solve a set of polynomial equations stemming from the flow in a gas network, in the context of the *nomination problem*.

In chapters 2 and 3 we derive the system of equations that defines the gas flow through the fundamental cycles of a network, which is enough to describe the flow through the entire network. We also introduce the basic concepts and ideas from the theory of Gröbner basis, as well as the algebro-geometric tools that justify the use of Gröbner basis to solve systems of polynomial equations.

We then turn in chapter 4 to the study of how the shape and the coefficients of the network can affect the runtime of the algorithm. While simple networks with simple coefficients give often a good enough performance, once we take loads that are probably closer to a real scenario the complexity of the computation of the solution increases dramatically.

We also study the errors associated to a solution computed using Gröbner basis, and introduce the idea of the comprehensive Gröbner system as a tool to solve parametrically the equations. This allows us to remove the error due to the construction of the Gröbner basis, but it also gives us relations between the flow and the loads that we can use to study the numerical conditioning of the system. We also introduce the Shape Lemma, that tells us that under some circumstances (that we expect to hold almost always) the shape of the Gröbner basis will minimize the error stemming from the approximation of the solutions of univariate polynomials. We also study the problem of the determination of feasible flow directions, and show how a small increase in network connectivity can create many new possible flow directions. To deal with this problem we explain how a parametric approach may be used to obtain directly a solution of the network with its respective flow direction, although this comes with an increase in runtime that makes it practical only for situations where few of the directions are unknown.

For larger networks, however, the computation of a full Gröbner basis becomes impractical due to the time required, when compared with other numerical methods, so the methods explained before are not useful. To be able to tackle larger networks we therefore propose a scheme of network reduction based on iterated V -cycles, similar in design to multigrid algorithms. We perform a topological reduction of the network to arrive to a reduced version that can be solved directly using Gröbner basis, and then obtain a solution to the full network by extending the reduced solution to the removed subnetworks. While this is still slower than other approaches when we have no previous information about the network, once we have computed a solution for a load q^{nom} it is possible to compute

solutions to similar loads up to a high degree of accuracy by considering a smaller amount of cycles and in a fraction of the time.

Overall, the negative aspects about the Gröbner basis (mainly the exponential runtime) outweigh its benefits (for instance computing a very precise solution, and being able to detect infeasibility very fast). The main limit is therefore the inherent exponential behaviour of any algorithm that computes a Gröbner basis. Although quantum algorithms have been proposed to solve the multivariate quadratic problem (see J.-C. Faugère et al. 2017), they only deal with the case $K = \mathbb{F}_2$, so one would need to translate those results to the framework of characteristic zero fields.

While comprehensive Gröbner basis would be a good solution, the fact that they are so hard to compute (not getting results even after days of computations) prevents us from realizing their full potential (even though the computation of a Gröbner system only has to be done once). Still, for smaller networks with at most two fundamental cycles (and not too many nodes) they can be a very useful tool to study the conditioning of the flow with respect to loads at some of the nodes. A possible use case would be to try to apply it to search for congestion points, as defined in Fügenschuh et al. 2011, identifying the key pipes when the load in a node is increased from a set baseline load.

While the method of network reduction gives us a good result to apply Gröbner basis to larger networks, it is still not as efficient as other numerical algorithms. In particular, the main issue is that the model of gas networks we are considering is very simplified: we include only nodes and pipes and forget about compressor stations, control valves and other elements. The description of the pressure drop is also simplified to allow us to break up the network between the depth-first search tree and the fundamental cycles, which decreases the amount of equations to solve. In order to increase the accuracy of the network it would be necessary to consider a more complex model of the pipe flow, which would add further nonlinearities, as well as additional elements like compressors, even with simplified models. Therefore, the biggest improvements to our methods could be to define and model more complex networks in terms of systems of polynomial equations with as few variables as possible, and using a Gröbner basis reduction approach to solve these systems.

Bibliography

- Aubry, Philippe, Daniel Lazard, and Marc Moreno Maza (1999). “On the theories of triangular sets”. In: vol. 28. 1-2. Polynomial elimination—algorithms and applications, pp. 105–124. DOI: 10.1006/jsc.1999.0269 (cit. on p. 26).
- Aubry, Philippe and Marc Moreno Maza (1999). “Triangular sets for solving polynomial systems: a comparative implementation of four methods”. In: vol. 28. 1-2. Polynomial elimination—algorithms and applications, pp. 125–154. DOI: 10.1006/jsc.1999.0270 (cit. on p. 26).
- Bang-Jensen, Jørgen and Gregory Gutin (2001). *Digraphs*. Springer Monographs in Mathematics. Springer-Verlag London, Ltd., London, pp. xxii+754. ISBN: 1-85233-268-9 (cit. on p. 44).
- Bardet, Magali (Nov. 2002). “On the Complexity of a Gröbner Basis Algorithm”. In: *Algorithms Seminar, 2002–2004*. Ed. by Frédéric Chyzak. INRIA, pp. 85–92 (cit. on p. 13).
- Bardet, Magali, Jean-Charles Faugère, and Bruno Salvy (2015). “On the complexity of the F_5 Gröbner basis algorithm”. In: *J. Symbolic Comput.* 70, pp. 49–70. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2014.09.025 (cit. on p. 25).
- Becker, Eberhard, Teo Mora, Maria Grazia Marinari, and Carlo Traverso (1994). “The Shape of the Shape Lemma”. In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation*. ISSAC '94. Oxford, United Kingdom: Association for Computing Machinery, pp. 129–133. ISBN: 0897916387. DOI: 10.1145/190347.190382 (cit. on p. 41).
- Caniglia, Léandro, André Galligo, and Joos Heintz (1989). “Some new effectivity bounds in computational geometry”. In: *Applied algebra, algebraic algorithms and error-correcting codes (Rome, 1988)*. Vol. 357. Lecture Notes in Comput. Sci. Springer, Berlin, pp. 131–151. DOI: 10.1007/3-540-51083-4_54.
- Courtois, Nicolas, Alexander Klimov, Jacques Patarin, and Adi Shamir (2000). “Efficient algorithms for solving overdefined systems of multivariate polynomial equations”. In: *Advances in cryptology—EUROCRYPT 2000 (Bruges)*. Vol. 1807. Lecture Notes in Comput. Sci. Springer, Berlin, pp. 392–407. DOI: 10.1007/3-540-45539-6_27 (cit. on p. 28).
- Cox, David A., John Little, and Donal O’Shea (1998). *Using algebraic geometry*. Vol. 185. Graduate Texts in Mathematics. Springer-Verlag, New York, pp. xii+499. ISBN: 0-387-98492-5. DOI: 10.1007/978-1-4757-6911-1 (cit. on p. 25).
- (2015). *Ideals, varieties, and algorithms*. 4th ed. Undergraduate Texts in Mathematics. Springer, Cham, pp. xvi+646. ISBN: 978-3-319-16720-6. DOI: 10.1007/978-3-319-16721-3 (cit. on p. 13).

- Decker, Wolfram, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann (2020). *SINGULAR 4-1-2 — A computer algebra system for polynomial computations*. <http://www.singular.uni-kl.de> (cit. on p. 32).
- Ekhtiari, Ali, Ioannis Dassios, Muyang Liu, and Eoin Syron (2019). “A Novel Approach to Model a Gas Network”. In: *Applied Sciences* 9.6. ISSN: 2076-3417. DOI: 10.3390/app9061047 (cit. on pp. 11, 63, 64, 67, 69, 70, 72).
- Faugère, J. C., P. Gianni, D. Lazard, and T. Mora (1993). “Efficient computation of zero-dimensional Gröbner bases by change of ordering”. In: *J. Symbolic Comput.* 16.4, pp. 329–344. ISSN: 0747-7171. DOI: 10.1006/jscs.1993.1051 (cit. on pp. 24, 25).
- Faugère, Jean-Charles (1999). “A new efficient algorithm for computing Gröbner bases (F_4)”. In: *J. Pure Appl. Algebra* 139.1-3. Effective methods in algebraic geometry (Saint-Malo, 1998), pp. 61–88. ISSN: 0022-4049. DOI: 10.1016/S0022-4049(99)00005-5 (cit. on p. 24).
- (2002). “A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5)”. In: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. ACM, New York, pp. 75–83. DOI: 10.1145/780506.780516 (cit. on p. 24).
- Faugère, Jean-Charles, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret (2017). *Fast Quantum Algorithm for Solving Multivariate Quadratic Equations*. eprint: arXiv:1712.07211 (cit. on p. 74).
- Fuchs, Barbara (2018). “Numerical Methods for Uncertainty Quantification in Gas Network Simulation”. PhD thesis. Rheinische Friedrich-Wilhelms-Universität Bonn.
- Fügenschuh, A., B. Hiller, J. Humpola, T. Koch, T. Lehmann, R. Schwarz, J. Schweiger, and J. Szabó (May 2011). “Gas network topology optimization for upcoming market requirements”. In: *2011 8th International Conference on the European Energy Market (EEM)*, pp. 346–351. DOI: 10.1109/EEM.2011.5953035 (cit. on p. 74).
- Gotzes, Claudia, Holger Heitsch, René Henrion, and Rüdiger Schultz (2016). “On the quantification of nomination feasibility in stationary gas networks with random load”. In: *Math. Methods Oper. Res.* 84.2, pp. 427–457. ISSN: 1432-2994. DOI: 10.1007/s00186-016-0564-y (cit. on p. 58).
- Gotzes, Claudia, Sabrina Nitsche, and Rüdiger Schultz (Mar. 2017). “Probability of Feasible Loads in Passive Gas Networks with up to Three Cycles”. Preprint available at <https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/135>. (cit. on p. 37).
- Hillebrand, D. (2018). *triang.lib. A SINGULAR 4-1-2 Library to Decompose Zero-dimensional Ideals into Triangular Sets* (cit. on p. 27).
- Hiller, Benjamin and Tom Walther (Nov. 2017). “Modelling compressor stations in gas networks”. In: *ZIB Report* 17-67. ISSN: 1438-0064 (cit. on p. 3).
- Kapur, Deepak, Yao Sun, and Dingkang Wang (2010). “A new algorithm for computing comprehensive Gröbner systems”. In: *ISSAC 2010—Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*. ACM, New York, pp. 29–36. DOI: 10.1145/1837934.1837946 (cit. on p. 36).
- Kipnis, Aviad and Adi Shamir (1999). “Cryptanalysis of the HFE public key cryptosystem by relinearization”. In: *Advances in cryptology—CRYPTO ’99 (Santa Barbara, CA)*.

- Vol. 1666. Lecture Notes in Comput. Sci. Springer, Berlin, pp. 19–30. DOI: 10.1007/3-540-48405-1_2 (cit. on p. 28).
- Kondratyev, A., H.J. Stetter, and F. Winkler (2004). “Numerical Computation of Gröbner Bases”. In: *Proc. 7th Workshop on Computer Algebra in Scientific Computing (CASC-2004)* (St. Petersburg, Russia, 2004). Technische Univ. München, pp. 295–306. ISBN: 3-9808546-2-0 (cit. on p. 35).
- Lazard, Daniel (1983). “Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations”. In: *Computer algebra (London, 1983)*. Vol. 162. Lecture Notes in Comput. Sci. Springer, Berlin, pp. 146–156. DOI: 10.1007/3-540-12868-9_99 (cit. on p. 33).
- Matsumoto, Tsutomu and Hideki Imai (1988). “Public quadratic polynomial-tuples for efficient signature-verification and message-encryption”. In: *Advances in cryptology—EUROCRYPT ’88 (Davos, 1988)*. Vol. 330. Lecture Notes in Comput. Sci. Springer, Berlin, pp. 419–453. DOI: 10.1007/3-540-45961-8_39 (cit. on p. 28).
- Mohring, Jan, Jochen Hoffmann, Thomas Halfmann, Aivars Zemitis, Giuliano Basso, and Per Lagoni (Jan. 2004). “Automated model reduction of complex gas pipeline networks”. In: *PSIG Annual Meeting, 2004*. Pipeline Simulation Interest Group.
- Möller, H. Michael (1993). “On decomposing systems of polynomial equations with finitely many solutions”. In: *Appl. Algebra Engrg. Comm. Comput.* 4.4, pp. 217–230. ISSN: 0938-1279. DOI: 10.1007/BF01200146 (cit. on p. 27).
- Möller, H. Michael and Ferdinando Mora (1984). “Upper and lower bounds for the degree of Groebner bases”. In: *EUROSAM 84 (Cambridge, 1984)*. Vol. 174. Lecture Notes in Comput. Sci. Springer, Berlin, pp. 172–183. DOI: 10.1007/BFb0032840.
- Montes, Antonio (1998). “Algebraic solution of the load-flow problem for a 4-nodes electrical network”. In: vol. 45. 1-2. Non-standard applications of computer algebra (Linz/Wailea, HI, 1996/1997), pp. 163–174. DOI: 10.1016/S0378-4754(97)00092-X (cit. on p. 36).
- Montes, Antonio and Jordi Castro (Jan. 1995). “Solving the load flow problem using Gröbner basis”. In: *ACM SIGSAM Bulletin* 29, pp. 1–13. DOI: 10.1145/216685.216686 (cit. on p. 36).
- Montes, Antonio and Hans Schoenemann (2018). *grobcov.lib. A SINGULAR 4-1-2 Library for Groebner Cover for Parametric Ideals* (cit. on p. 36).
- Nagasaka, Kosaku (2009). “A Study on Gröbner Basis with Inexact Input”. In: *Computer Algebra in Scientific Computing*. Vol. 5743. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 247–258. ISBN: 978-3-642-04103-7. DOI: 10.1007/978-3-642-04103-7_22.
- Nitsche, Sabrina (2018). “Analytical and Algebraic Approaches to Gas Transportation with Uncertain Loads”. PhD thesis. Universität Duisburg-Essen (cit. on pp. 36, 37, 41–43, 58).
- Pfetsch, Marc E., Armin Fügenschuh, Björn Geißler, Nina Geißler, Ralf Gollmer, Benjamin Hiller, Jesco Humpola, Thorsten Koch, Thomas Lehmann, Alexander Martin, Antonio Morsi, Jessica Rövekamp, Lars Schewe, Martin Schmidt, Rüdiger Schultz, Robert Schwarz, Jonas Schweiger, Claudia Stangl, Marc C. Steinbach, Stefan Vigerske, and Bernhard M. Willert (2015). “Validation of nominations in gas network optimization:

- models, methods, and solutions”. In: *Optimization Methods and Software* 30.1, pp. 15–53. DOI: 10.1080/10556788.2014.888426 (cit. on pp. 1, 11).
- “Regulation (EC) No 715/2009 of the European Parliament and of the Council of 13 July 2009 on conditions for access to the natural gas transmission networks and repealing Regulation (EC) No 1775/2005” (2009). In: *Official Journal* L211, pp. 36–49. DOI: 10.3000/17252555.L_2009.211.eng.
- Renshukh, B., Kh. Roloff, and G. G. Rasputin (1987). “On forgotten papers of N. M. Günter on the theory of polynomial ideals”. In: *Vestnik Leningrad. Univ. Mat. Mekh. Astronom.* vyp. 1, pp. 119–122, 127. ISSN: 0024-0850 (cit. on p. 20).
- Ríos Mercado, Roger Z., Suming Wu, L. Ridgway Scott, and E. Andrew Boyd (2002). “A Reduction Technique for Natural Gas Transmission Network Optimization Problems”. In: *Ann. Oper. Res.* 117.1-4, pp. 217–234. ISSN: 1572-9338. DOI: 10.1023/A:1021529709006 (cit. on pp. 10, 51).
- The Sage Developers (2020). *SageMath, the Sage Mathematics Software System (Version 9.0)*. <https://www.sagemath.org> (cit. on p. 32).
- Sasaki, Tateaki and Fujio Kako (2007). “Computing floating-point Gröbner bases stably”. In: *SNC’07*. ACM, New York, pp. 180–189 (cit. on p. 35).
- (2010). “Term Cancellations in Computing Floating-Point Gröbner Bases”. In: *Computer Algebra in Scientific Computing*. Vol. 6244. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 220–231. ISBN: 978-3-642-15274-0. DOI: 10.1007/978-3-642-15274-0_20 (cit. on p. 34).
- Shafarevich, Igor R. (2013). *Basic algebraic geometry. 1*. Russian. Varieties in projective space. Springer, Heidelberg, pp. xviii+310. ISBN: 978-3-642-37955-0; 978-3-642-37956-7 (cit. on p. 30).
- Thomae, Enrico and Christopher Wolf (2010). *Solving Systems of Multivariate Quadratic Equations over Finite Fields or: From Re-linearization to MutantXL*. Cryptology ePrint Archive, Report 2010/596. <https://eprint.iacr.org/2010/596> (cit. on p. 28).
- Trottenberg, U., C. W. Oosterlee, and A. Schüller (2001). *Multigrid*. Academic Press, Inc., San Diego, CA, pp. xvi+631. ISBN: 0-12-701070-X (cit. on p. 60).
- Tschoegl, Nicholas W. (2000). *Fundamentals of Equilibrium and Steady-State Thermodynamics*. Elsevier Science, p. 280. ISBN: 9780444504265. DOI: 10.1016/B978-0-444-50426-5.X5000-9 (cit. on p. 59).
- Weispfenning, Volker (1992). “Comprehensive Gröbner bases”. In: *J. Symbolic Comput.* 14.1, pp. 1–29. ISSN: 0747-7171. DOI: 10.1016/0747-7171(92)90023-W (cit. on p. 35).
- Wenk, Moritz and Wilfred Pohl (2019). *solve.lib. A SINGULAR 4-1-2 Library for Complex Solving of Polynomial Systems* (cit. on pp. 32, 49).