Technische Universität Berlin
Institut für Mathematik

Masterarbeit

Fast Approximation of Equations of transient Gasflow

Andre Thorsten Weltsch

Berlin 2018

Erstgutachter: Prof. Dr. Thorsten Koch
Zweitgutachter: Prof. Dr. Martin Skutella

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbständige und eigenständige Anfertigung versichert an Eides statt:

Berlin, den 7. Dezember 2018

Andre Thorsten Weltsch

Abstract

In this thesis, we study transient (time-dependent) gas flow in natural gas networks with a focus on passive gas networks. We compare different algorithms to approximate flow and pressure values in a gas network.

A gas network consists of pipelines that deliver gas from suppliers to consumers. The gas flow in a pipe can be described by the Euler Equations which are a set of nonlinear partial differential equations. We use the so-called friction-dominated model to simplify the gas flow equations and apply an implicit box scheme to derive a suitable discretization.

We compare different linearization methods to solve the resulting nonlinear system of equations.

We introduce six known mixed-integer formulations for piecewise linear functions and describe how to use these formulations to approximate the nonlinear functions in the gas flow model.

Further, we develop an iterative method, to solve the pipe flow system. We call the method "iterative velocity approximation", and it is based on a linearization of the momentum equation. We linearize the equation by fixing the absolute gas flow velocity. We show convergence on single pipes.

We evaluate and compare the performance of generic nonlinear solvers, the piecewise linear approximations and the iterative velocity approximation on nine different networks.

Acknowledgments

The work for this thesis has been conducted in cooperation with Zuse Institute Berlin (ZIB) within the Research Campus MODAL funded by the German Federal Ministry of Education and Research (BMBF) (fund number 05M14ZAM).

Contents

1. Introduction

1.1. The Problem

Natural gas is among the most used energy sources in Germany. In the year 2016 approximately 24% of the final energy consumption was covered by natural gas (see Table A.1 in Appendix A) As a consequence large amounts of natural gas have to be transported from suppliers to consumers.

The gas that is transported flows through pipelines, and due to friction in the pipes, the pressure drops. A gas network typically contains additional elements that can control the flow in some way. For example, compressor machines are used to increase pressure, which counteracts the pressure loss incurred by pipes. This kind of *active* elements introduces discrete decisions into the network control. Typically you try to optimize the network control w.r.t. some objective (e.g., fuel consumption of the compressor machines). This kind of optimization is often called *Transient Technical Optimization* ([Moritz, 2007] and [Ehrhardt and Steinbach, 2005]). Based on the state of the network and future supply/demand values, the goal is to find an optimal network control, which serves as a recommendation to human operators (so-called dispatchers), who can then adjust the network control based on the recommendation from the optimization process. Due to the inherently nonlinear nature of the gas flow, large mixed-integer nonlinear problems need to be solved. Current solution approaches are not able to solve large-scale problems (see [Ríos-Mercado and Borraz-Sánchez, 2015]).

To make the optimization problem manageable, we try to linearize the gas flow constraints. By linearizing the problem, we can make use of powerful mixed-integer linear programming solvers.

In this thesis, we want to compare different algorithms to calculate flow and pressure values in a gas network. In contrast to *stationary* calculations, in which a constant capacity is requested over an infinite amount of time, we take a look at a *transient* flow problem. We are given a gas network, as well as, gas supply and demand values at entry and exit points of the network, which are fixed over a specified time horizon. We then want to compute the flow and pressure values at each node in the network for all time points, s.t. the network can transport the specified amount of natural gas to satisfy all given demands.

The intent is to create a linearization of the pipe flow that can be incorporated into transient technical optimization. Therefore, we concentrate on pipe-only networks and present and compare strategies to linearize the discretization of the so-called friction-dominated pipe flow model (see [Domschke et al., 2017]).

In the second chapter, we introduce the notation and physical relations for gas flow in networks. We present the *Euler Equations* which are a set of nonlinear partial differential equations that describe gas flow in a pipe. We then deduce the friction-dominated model for gas flow in pipelines and define the discretization that we use throughout the thesis. The second chapter summarizes all mixed-integer formulations for piecewise linear functions from [Vielma et al., 2010], and we explain how we can use these formulations to approximate the nonlinear functions that occur in the discretization for gas flow. The third chapter discusses an iterative method which we call *iterative velocity approximation* for which we prove convergence on a single pipe. In Chapter 5 we test the convergence properties of iterative velocity approximation on different networks and compare different solution approaches for the gas flow system. The approaches include generic methods as implemented by nonlinear solvers, as well as all of the piecewise linear formulations from Chapter 3 and iterative velocity approximation introduced in Chapter 4. In the last chapter, we classify and asses our results and discuss ideas for further research.

1.2. Literature Survey

Transient gas network optimization is an area of ongoing research. The corresponding optimization problems are large mixed-integer nonlinear programs, and a number of different solution approaches based on linearization of the underlying nonlinearities have been developed.

Piecewise linear functions have already been proposed and used to approximate nonlinear functions in transient calculations. For example, [Moritz, 2007] and [Correa-Posada and Sánchez-Martín, 2014] use mixed-integer formulation for piecewise linear functions to approximate nonlinearities in the pipe flow and the compressor model. In contrast to [Moritz, 2007], we use different formulations for piecewise linear functions, and we use a different discretization of the pipe flow equation than both [Moritz, 2007] and [Correa-Posada and Sánchez-Martín, 2014].

[Burlacu et al., 2017b] propose a general iterative approach to solve MINLPs. They use piecewise linear functions to create a relaxation that encloses the nonlinearity within a given error bound. They use the relaxation to solve a mixed-integer problem to fix discrete decisions in the network. The solution can then be used to refine the discretization of the piecewise linear functions until some error bound is achieved. In contrast to their method of adaptive refinement we do *not* use relaxations for the nonlinearities, we only approximate the nonlinearities. Also, we do not refine the discretization. Instead, we only work with a priori discretizations.

[van der Hoeven, 2004] and [Pratt and Wilson, 1984] present solution methods based on an iterated solution of linear programs. They linearize the gas flow equations by fixing the absolute flow value on a pipe. Our approach differs because instead of fixing the flow value we fix the absolute velocity of the gas (which also involves pressure). Furthermore, the focus of both [van der Hoeven, 2004] and [Pratt and Wilson, 1984] are stationary calculations, whereas we solely deal with transient flows.

2. Modeling Transient Gas Flow

The goal of this chapter is to discuss the different elements of a gas network and their physical characteristics and mathematical models. We then define the appropriate models that we use for computations throughout the rest of this thesis. The information is based on [Domschke et al., 2017], [Fügenschuh et al., 2015] and [Moritz, 2007].

As we have already mentioned in the introduction, gas networks are used to distribute natural gas. The distribution takes place between several entry points and exit points, and the gas has to be transported via different network elements. The entry points add gas to the network, and at exit points, gas is extracted. The most well-known network element is the ordinary pipe as part of a bigger pipeline. The pipe is the main component that we examine in this thesis. Before we talk about the details of modeling gas flow in pipes we give a short overview of the most common network elements:

- *Pipelines* represent the major part of a gas network, they are used to connect entries and exits of the network and can be considered the basic building block of the network. Pipes can come in different shapes, but most of the transport networks that we consider only contain pipes with a cylindrical shape, so similar to [Fügenschuh et al., 2015] we only model straight cylindrical pipes. As we will later see in greater detail, the length, diameter and the internal roughness of the pipe have a significant influence on the gas flowing through it. Depending on these parameters, pipes impose a pressure drop between their endpoints.
- Another element commonly found in gas networks is called *resistor*. These elements are artificial constructs that represent different kinds of pressure loss due to a diverse set of reasons (e.g., flow diversion). To account for complex pressure loss phenomena, different parameters are estimated from experimental data.
- *Short cuts* are very short pipes that do not impose a pressure loss between the start node and the end node.
- *Valves* are used to regulate flow between certain parts of the network. In our model, a valve can be open or closed, i.e., enabling or disabling flow between two endpoints. We neglect any friction inside a valve and thus assume the same pressure at both endpoints if the valve is open.
- *Control valves* are devices relatively similar to valves, which are used as pressure regulators between high-pressure and low-pressure parts of the network. Control valves have an additional state (called "active"), in which the control valve is partially opened, thus introducing a high pressure loss between both endpoints. In contrast to a classic valve, control valves have a working direction.

- *Compressor machines* are probably the most complicated of the network elements. They serve the vital function to counter pressure loss incurred by the rest of the network. Thus they are crucial to enable long-distance transport of gas. They are active elements and can increase pressure in the flow direction, but they need some energy to operate.

2.1. Network Structure and Element Coupling

We try to follow the notation from [Fügenschuh et al., 2015] and [Geißler et al., 2015], but we have to adapt their notation to the transient case, i.e., we add an index for the time to each of the variables.

In a gas network, the different network elements are joined and connected in some way. Therefore gas networks can be modeled as directed graphs, which are represented as $G = (V, A)$, where V is the set of nodes and A is the set of arcs. We do not allow self-loops, but parallel arcs are possible. Additionally, we define the incoming and the outgoing arcs of a node $u \in V$ as

$$\begin{aligned} \delta^{+}(u) &= \{(v,u) \in A\}, \\ \delta^{-}(u) &= \{(u,v) \in A\}. \end{aligned}$$

The set of nodes is divided into three subsets

$$\begin{aligned} V_{+} &\triangleq \text{set of entry nodes (nodes that supply gas)}, \\ V_{0} &\triangleq \text{set of inner nodes (neither supply or demand)}, \\ V_{-} &\triangleq \text{set of exit nodes (nodes that demand gas)}, \end{aligned}$$

so the complete set of nodes is $V = V_{+} \cup V_{0} \cup V_{-}$.

The set of arcs is also divided into subsets, one set for each of the network elements:

$$\begin{aligned} A_{pi} &\triangleq \text{set of pipes}, \\ A_{rs} &\triangleq \text{set of resistors}, \\ A_{sc} &\triangleq \text{set of short cuts}, \\ A_{va} &\triangleq \text{set of valves}, \\ A_{cv} &\triangleq \text{set of control valves}, \\ A_{cs} &\triangleq \text{set of compressor stations}. \end{aligned}$$

The set of all arcs is then $A = A_{pi} \cup A_{rs} \cup A_{sc} \cup A_{va} \cup A_{cv} \cup A_{cs}$. Details on pipes and its parameters can be found in Subsection 2.2.1 Additional information on how to integrate the other network elements (valves, control valves, etc.) can be found in Subsection 2.2.2.

We want to model gas flow over a discrete time horizon $\mathcal{T} = \{t_0 < t_1 < \ldots < t_n\}$. We want to find flow and pressure values s.t. a given flow demand can be satisfied. The flow demand is given as *mass flow* in the nodes for each of the timesteps. For some node

$u \in V$ and some time point $t_i \in \mathcal{T}$ we denote the demand as q^{nom}_{u,t_i}. For the different types of nodes we restrict the values of the demand in the following way:

$$q^{nom}_{u,t_i} \begin{cases} \geq 0, & u \in V_+ \\ = 0, & u \in V_0 \\ \leq 0, & u \in V_- \end{cases} .$$

So in entry nodes, we introduce gas, in inner nodes, no gas can enter or exit the network, and in exit nodes, gas is withdrawn from the network. These values serve as the boundary conditions to the transient flow problem.

The flow in the network is transported through the different elements, so the mass flow q in a connection $a \in A \setminus A_{pi}$ at time t is referred to as $q_{a,t}$. For pipes, we have a different mass flow at the inlet and the outlet, thus we define the inlet mass flow at $a \in A_{pi}$ at time t as $q^{in}_{a,t}$ and for the outlet, we use $q^{out}_{a,t}$. The pressure at a node $u \in V$ at time t is referred to as $p_{u,t}$.

We assume that there are bounds for the flow on each network element, i.e.,

$$\underline{q}_a \leq q_{a,t} \leq \overline{q}_a, \quad \forall t \in \mathcal{T}. \tag{2.1}$$

Each node in the network couples different incoming/outgoing network elements. At some node $v \in V$, the pressure at the end of an element (for incoming elements) and the pressure at the start of an element (for outgoing elements) need to be the same. So as a consequence we only need a single pressure variable $p_{v,t}$ per node $v \in V$.

There are also limits on the pressure at each node $u \in V$:

$$\underline{p}_u \leq p_{u,t} \leq \overline{p}_u, \quad \forall t \in \mathcal{T}. \tag{2.2}$$

At nodes, different network elements are connected, and therefore we introduce a coupling condition for the network elements. Mass conservation results in a flow conservation constraint so at each timestep t for the node $u \in V$ we get:

$$\begin{aligned} &\sum_{a \in \delta^+(u) \setminus A_{pi}} q_{a,t} + \sum_{a \in \delta^+(u) \cap A_{pi}} q^{out}_{a,t} \\ &- \sum_{a \in \delta^-(u) \setminus A_{pi}} q_{a,t} - \sum_{a \in \delta^-(u) \cap A_{pi}} q^{in}_{a,t} = q^{nom}_{u,t}, \quad \forall t \in \mathcal{T}. \end{aligned} \tag{2.3}$$

Now that we have explained the network structure the next step is to describe the physical behaviour and mathematical models in each of the network elements.

2.2. Models for Passive Elements

In this work we are mostly interested in passive subnetworks, i.e., we will not go into the details of active elements. How we treat these active elements can be gathered from the subsection 2.2.2.

2.2.1. Pipe Flow Model

Gas flow in pipelines is a complex phenomenon, and accurate calculations depend on a lot of different parameters and are computationally very expensive. As we are interested in modeling large gas networks and therefore we need to find a way to simplify the calculations to make them feasible for large-scale calculations. As part of the simplifications, we lose accuracy of the result, and we need to make sure, that the simplifications are accurate enough for our purposes. We will introduce a rather complex model, and step by step simplify the model. These model simplifications are performed according to [Domschke et al., 2017].

To describe the gas flow through a pipeline, there are different quantities of interest: mass flow q, density ρ and velocity v of the gas as well as the pressure p at the endpoints of a pipe. The shape and properties of the pipe itself also play a crucial role in the calculation. To achieve the aforementioned simplifications we will only consider gas flow through cylindrical pipes and model the flow only one-dimensional as opposed to three-dimensional calculations in fluid-dynamics. Gas flow in pipes is modeled via the so-called *Euler Equations* (2.4) - (2.6) which are a set of nonlinear partial differential equations:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0, \tag{2.4}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(p + \rho v^2)}{\partial x} + g\rho s + \lambda \frac{|v|\, v}{2D}\rho = 0, \tag{2.5}$$

$$\begin{aligned} &\frac{\partial}{\partial x}\left(\rho v\left(\frac{1}{2}v^2 + e\right) + pv\right) \\ &+\frac{\partial}{\partial t}\left(\rho\left(\frac{1}{2}v^2 + e\right)\right) + \frac{k_w}{D}(T - T_w) = 0. \end{aligned} \tag{2.6}$$

Table 2.1 lists the parameters of the Euler Equations with short descriptions and the standard unit for each parameter. Equation (2.4) is commonly referred to as the *continuity equation*. Equation (2.5) is called the *momentum equation*. Equation (2.6) describes *energy conservation*.

According to [Schewe et al., 2015] most pipes are buried underground, so the surrounding temperature only changes slowly. Additionally, the typical network contains gas preheaters and coolers, so we assume to have a constant temperature in the pipelines. As a consequence, we can simplify the Euler Equations. We can then drop the energy conservation equation (2.6), and we get a simplification referred to as the *isothermal Euler Equations*:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0, \tag{2.7}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(p + \rho v^2)}{\partial x} + g\rho s + \lambda \frac{|v|\, v}{2D}\rho = 0. \tag{2.8}$$

As additional constraints, we have the state equation for real gases

$$p = \rho R_s T z(p, T), \tag{2.9}$$

Parameter	Description	Unit
p	pressure of gas	Pa
q	mass flow	kg/s
ρ	density of gas	kg/m^3
v	velocity of gas	m/s
D	diameter of pipe	m
A	cross sectional area of pipe	m^2
g	gravitational acceleration	m/s^2
s	slope of pipe	1
λ	friction factor	1
z	compressibility factor	1
e	internal energy	J
T	gas temperature	K
T_w	wall temperature of pipe	K
k_w	heat transfer coefficient	J/(m^{2}K)

Table 2.1.: Physical Quantities used in the Euler Equations

and the definition of the mass flow, which describes the mass flow in terms of gas velocity and density:

$$q = A\rho v. \tag{2.10}$$

The compressibility factor $z(p,T)$ is dependent on the gas composition, pressure, as well as temperature and there are different approximative formulas available. The most well-known is the formula by Papay (see [Papay, 1968] and [Saleh, 2002]) and an equation from the American Gas Association (AGA) (see [Králik et al., 1988]). In our calculations, we always assume a constant compressibility factor. For a constant compressibility factor, the speed of sound in gas c is given as

$$c = \sqrt{\frac{p}{\rho}}.$$

The gas flow velocity in large-scale networks is relatively small compared to the speed of sound, so for the assumption $v \ll c$ we can also assume that $v^2/c^2 \approx 0$ and thus

$$p + \rho v^2 = p\left(1 + \frac{v^2}{c^2}\right) \approx p.$$

Further, we assume that the term $\frac{\partial(\rho v)}{\partial t}$ is small and hence neglect it. This results in the so-called friction dominated model from [Brouwer et al., 2011]

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial x} &= 0, \\ \frac{\partial p}{\partial x} + g\rho s + \lambda \frac{|v|\, v}{2D}\rho &= 0. \end{aligned}$$

We also assume that the friction coefficient lambda is constant and to calculate the coefficient, we use the formula from [Nikuradse, 1950]

$$\lambda = \left(2\log_{10}\left(\frac{D}{k}\right) + 1.138\right)^{-2},$$

where D is the *diameter* and k is the *integral roughness* of the pipe.

In our case, the values we are interested in (and which we have measurements for) are the pressure as well as the flow along a pipe, so we use the equation (2.10) and the state equation (2.9) in order to reformulate the friction dominated model in terms of the pressure p and the mass flow q.

$$\frac{A}{R_sTz}\frac{\partial p}{\partial t} + \frac{\partial q}{\partial x} = 0, \tag{2.11}$$

$$\frac{\partial p}{\partial x} + \frac{\lambda R_sTz}{2DA^2}\frac{|q|\,q}{p} + \frac{gs}{R_sTz}p = 0. \tag{2.12}$$

These two equations (2.11) and (2.12) will serve as the basis for the solution strategies. As we are still dealing with a set of nonlinear PDEs, we discretize the system, so that we can work with solutions in a finite-dimensional vector space as opposed to general function spaces.

To discretize the equations, we need to discretize in the two dimensions time t and space x. For the discretization, we consider a pipe of length L from start time t_0 to end time $t_1 > t_0$. We discretize via an implicit box scheme from [Domschke et al., 2011]. We start with equation (2.11), where we integrate by space and time. Remaining integrals are approximated by the trapezoid rule for space and the right rectangle rule for the time integral. Thus the result is:

$$\begin{aligned}
&\int_{t_0}^{t_1}\int_0^L \frac{\partial p}{\partial t} + \frac{R_sTz}{A}\frac{\partial q}{\partial x}dxdt \\
&= \int_0^L (p_{x,t_1} - p_{x,t_0})dx + \int_{t_0}^{t_1}\frac{R_sTz}{A}(q_{L,t} - q_{0,t})dt \\
&= \frac{L}{2}\Big(\big(p_{0,t_1} - p_{0,t_0}\big) + \big(p_{L,t_1} - p_{L,t_0}\big)\Big) \\
&\quad + \frac{R_sTz(t_1 - t_0)}{A}(q_{L,t_1} - q_{0,t_1}).
\end{aligned} \tag{2.13}$$

In the second equation there is only a derivative in space, so we only need to integrate via space. We get:

$$\begin{aligned}
&\int_0^L \frac{\partial p}{\partial x} + \frac{\lambda R_sTz}{2DA^2}\frac{|q|\,q}{p} + \frac{gs}{R_sTz}p\,dx \\
&= p_{L,t} - p_{0,t} + \frac{\lambda R_sTz}{2DA^2}\frac{L}{2}\left(\frac{|q_{0,t}|\,q_{0,t}}{p_{0,t}} + \frac{|q_{L,t}|\,q_{L,t}}{p_{L,t}}\right) \\
&\quad + \frac{gsL}{2R_sTz}(p_{0,t} + p_{L,t}).
\end{aligned} \tag{2.14}$$

In a gas network several pipes are coupled, so for each pipe $a = (u, v) \in A_{pi}$ we introduce an additional index for its parameters. When we then formulate (2.13) and (2.14) in terms of inflow and outflow for some time point $t_i \in \mathcal{T}$, we end up with the following system

$$\begin{aligned} & p_{u,t_i} - p_{u,t_{i-1}} + p_{v,t_i} - p_{v,t_{i-1}} \\ & \quad \frac{2R_sTz(t_i - t_{i-1})}{A_a}(q^{out}a, t_i - q^{in}_{a,t_i}) && = 0, && (2.15) \\ & \left(1 + \frac{gsL_a}{2R_sTz}\right) p_{v,t_i} - \left(1 - \frac{gsL_a}{2R_sTz}\right) p_{u,t_i} \\ & \qquad + \frac{\lambda R_sTzL_a}{4D_aA_a^2}\left(\frac{\left|q^{in}_{a,t_i}\right| q^{in}_{a,t_i}}{p_{u,t_i}} + \frac{\left|q^{out}_{a,t_i}\right| q^{out}_{a,t_i}}{p_{v,t_i}}\right) && = 0. && (2.16) \end{aligned}$$

We will call (2.15) the *discretized continuity equation* and (2.16) the *discretized momentum equation.* The system of nonlinear equations consisting of the discretized continuity and momentum equation will be the main focus of this thesis. We try to find ways to solve the system. This is especially hard since the system contains the nonlinear *friction term*

$$f(p, q) = \frac{\lambda R_sTz}{2DA^2}\frac{|q|\,q}{p}. \tag{2.17}$$

We explore different ways to approximate the solution in chapters 3 and 4.

2.2.2. Dealing with Networks that Contain Non-Pipe Elements

We only want to consider passive networks. Thus we replace all elements with *short cuts.* Short cuts are easily modeled by adding a constraint that enforces equality of the pressure at both endpoints. We do not need to add the Euler Equations. Instead, for a short cut from some node u to another node v, we add the constraint

$$p_u = p_v. \tag{2.18}$$

For the active elements, this makes sense, because open valves and open control valves, as well as compressor stations in bypass mode show similar behaviour to short cuts, i.e., no pressure loss between endpoints.

With this modification, we can then examine our algorithms on networks that contain elements that are different to pipes, e.g., the GasLib networks (see Appendix B) contain active elements.

3. Modeling Piecewise Linear Functions

As we have seen in the previous section 2.2.1 about pipe flow models, the discretized system for gas flow in pipelines contains nonlinear terms in the pressure p and the flow q. These nonlinearities prevent us from using linear programming/mixed-integer programming techniques to optimize the network control.

Our goal is to approximate the nonlinearities in a way that is compatible with mixed-integer programming (MIP), so we can use powerful MIP-solvers to obtain a solution. One idea for such an approximation is to use piecewise linear functions. By discretizing the domain of a nonlinear function, we can then interpolate the discretized points with a piecewise linear function. As we will see, piecewise linear functions can be modeled by using mixed integer constraints. This property helps us to incorporate (approximations of) the nonlinear constraints into a MIP model. The goal of this section is to explore different MIP formulations for piecewise linear functions. [Vielma et al., 2010] contains an excellent overview and comparison of some models that are applicable to multivariate functions. We list these models and give a short explanation for each model and its corresponding formulation. We try to follow the naming convention from [Vielma et al., 2010]. Later we want to compare the performance of the different models when we use them to approximate the pipe flow system.

3.1. Introduction to Piecewise Linear Functions

Definition 3.1.1 (Continuous Piecewise Linear Function [Vielma et al., 2010])**.** *Let $\Omega \subseteq \mathbb{R}^n$ be a compact set. A continuous function $f : \Omega \to \mathbb{R}$ is a* piecewise linear function *(short: PWL) if and only if there exist a finite family of polytopes $\mathcal{P}$, vectors $\{m_P\}_{P\in\mathcal{P}} \subseteq \mathbb{R}^n$ and numbers $\{c_P\}_{P\in\mathcal{P}} \subseteq \mathbb{R}$ such that $\Omega = \bigcup_{P\in\mathcal{P}} P$ and*

$$f(x) := \begin{cases} m_P x + c_P & x \in P \, \forall P \in \mathcal{P}. \end{cases}$$

In the one-dimensional case, a piecewise linear function is uniquely defined via its breakpoints and their function values, i.e., the function value at each point can be inferred from the surrounding breakpoints. For functions with at least two variables, this is not necessarily true anymore, so the actual triangulation is essential, example 3.1.2 highlights this property. Especially if you are trying to approximate a nonlinear function, the choice of your simplices has a large influence on the accuracy of the approximation. Figure 3.1 presents an example of a univariate piecewise function.

Example 3.1.2. *Let us consider some function*

$$\begin{aligned} f : [0,1]^2 &\to \mathbb{R} \\ (x_1, x_2) &\mapsto 2x_1^2 + x_1x_2 + x_2^2 \end{aligned}$$

that we want to approximate with a PWL function. For the discretization, we choose some points

$$P = \{(0,0), (0,1), (1,0), (1,1)\}$$

with respective function values

$$\begin{aligned} f(0,0) &= 0, \\ f(0,1) &= 1, \\ f(1,0) &= 2, \\ f(1,1) &= 4. \end{aligned}$$

For P there exist two different triangulations:

$$\begin{aligned} \mathcal{P}_1 &= \{\{(0,0),(0,1),(1,0)\}, \{(0,1),(1,0),(1,1)\}\}, \\ \mathcal{P}_2 &= \{\{(0,0),(1,0),(1,1)\}, \{(0,0),(0,1),(1,1)\}\}. \end{aligned}$$

Then for $\mathcal{P}_1$, we can define a PWL function

$$f_1(x) := \begin{cases} (2,1) \cdot x & x \in \operatorname{conv}\{(0,0),(0,1),(1,0)\} \\ (3,2) \cdot x - 1 & x \in \operatorname{conv}\{(0,1),(1,0),(1,1)\} \end{cases},$$

and for $\mathcal{P}$ we define

$$f_2(x) := \begin{cases} (2,2) \cdot x & x \in \operatorname{conv}\{(0,0),(1,0),(1,1)\} \\ (3,1) \cdot x & x \in \operatorname{conv}\{(0,0),(0,1),(1,1)\} \end{cases},$$

We have equality $f(x) = f_1(x) = f_2(x)$ for all points $x \in P$, but

$$f_1(0.5, 0.5) = 1.5 \neq 2 = f_2(0.5, 0.5).$$

We will assume that the domain $\Omega \subset \mathbb{R}^n$ is given via a triangulation (i.e., each polytope is a simplex) $\mathcal{P}$, i.e., $\Omega = \bigcup_{P \in \mathcal{P}} P$. $\mathcal{V}(\mathcal{P})$ is the set of the vertices of all polytopes in $\mathcal{P}$. The goal is to model a piecewise linear function $f : \Omega \to \mathbb{R}$ as a MIP. In the MIP formulations, we always denote the input of the function as $x \in \Omega$, and the output $f(x)$ is represented by the variable $z \in \mathbb{R}$.

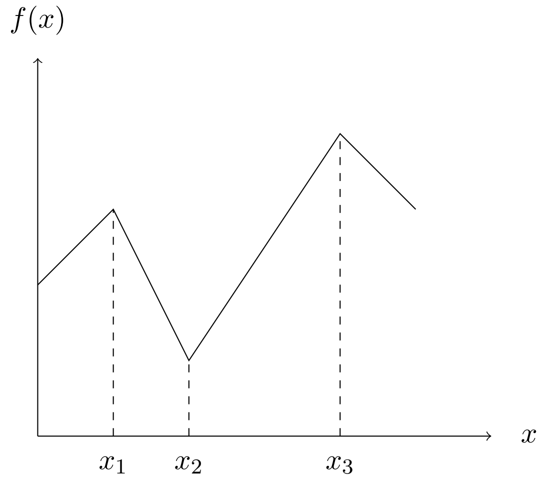


Figure 3.1.: Example of a Univariate Piecewise Linear Function

3.2. Disaggregated Convex Combination Models (DCC)

Convex combination models are based on the fact, that each point in a polytope is a convex combination of the vertices of the polytope. There are several different formulations based on the idea. So in the most basic model, binary decision variables $y_P \in \{0, 1\}$ are introduced for each polytope $P \in \mathcal{P}$, to decide in which polytope the point x is located. Additionally, for each $v \in P$, there are continuous variables $\lambda_{P,v} \in [0, 1]$ that represent the coefficient of v in the convex combination of x. The model we introduce is called disaggregated because even though a vertex can be part of several polytopes, there is a separate coefficient for each of the polytopes. As we will later see, there is also a formulation that only uses one continuous variable per vertex. The single variable case has appeared in [Meyer, 1976], and the multivariable case was developed in [Jeroslow and Lowe, 1984], [Lowe, 1984] and [Jeroslow, 1987]. The multivariable case describes a disjoint union of polyhedra. The MIP formulation is given by the constraints (3.1)-(3.6). We abbreviate the model (3.1)-(3.6) with DCC.

$$\sum_{P\in\mathcal{P}}\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} f(v) = z, \tag{3.1}$$

$$\sum_{P\in\mathcal{P}}\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} v = x, \tag{3.2}$$

$$\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} = y_P \qquad \forall P\in\mathcal{P}, \tag{3.3}$$

$$\sum_{P\in\mathcal{P}} y_P = 1, \tag{3.4}$$

$$\lambda_{P,v} \geq 0 \qquad \forall P\in\mathcal{P}\,\forall v\in\mathcal{V}(P), \tag{3.5}$$

$$y_P \in \{0,1\} \qquad \forall P\in\mathcal{P}. \tag{3.6}$$

The constraint (3.1) gives the output variable z as the convex combination of the function values of the vertices in the discretization. (3.2) states that the input variable x needs to be represented as a convex combination of the vertices. Constraint (3.4) states that only a single polytope can be chosen. (3.3) and (3.5) together enforce the convex combination property.

3.2.1. The Logarithmic Disaggregated Convex Combination Model (DLOG)

Similar to the previous model, this model is based on a disaggregated convex combination of the vertices in each polytope. The main difference in this model is that the number of binary variables is reduced to a number asymptotically logarithmic in the number of polytopes. This is achieved by identifying each polytope with a binary vector, where the corresponding mapping is

$$B : \mathcal{P} \to \{0,1\}^{\lceil \log_2 |\mathcal{P}| \rceil}.$$

We use the following definitions in the formulation:

$$\begin{aligned}
\mathcal{P}^0(B,l) &= \{P\in\mathcal{P} \mid B(P)_l = 0\},\\
\mathcal{P}^+(B,l) &= \{P\in\mathcal{P} \mid B(P)_l = 1\},\\
L(\mathcal{P}) &= \{1,\dots,\lceil \log_2 |\mathcal{P}| \rceil\}.
\end{aligned}$$

The set $\mathcal{P}^0(B,l)$ is the set of polytopes $P\in\mathcal{P}$ that are mapped by B to a binary vector in which the l-th coordinate is 0. The set $\mathcal{P}^+(B,l)$ is the complement, i.e., it consists of the polytopes that have a 1 in the l-th coordinate. L is the index set for the coordinates of the binary vectors.

For example, we can number the polytopes $\{P_1,\dots,P_{|\mathcal{P}|}\} = \mathcal{P}$. Then an example for a mapping B could map P_i to the binary representation of i. This formulation has been developed in [Vielma and Nemhauser, 2008]. A MIP formulation for the model is given by (3.7)-(3.13) We abbreviate the model as DLOG.

$$\sum_{P\in\mathcal{P}}\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} f(v) = z, \tag{3.7}$$

$$\sum_{P\in\mathcal{P}}\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} v = x, \tag{3.8}$$

$$\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} = 1, \tag{3.9}$$

$$\sum_{P\in\mathcal{P}^{+}(B,l)}\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} \leq y_l \qquad \forall l \in L(\mathcal{P}), \tag{3.10}$$

$$\sum_{P\in\mathcal{P}^{0}(B,l)}\sum_{v\in\mathcal{V}(P)} \lambda_{P,v} \leq (1-y_l) \qquad \forall l \in L(\mathcal{P}), \tag{3.11}$$

$$\lambda_{P,v} \geq 0 \qquad \forall P \in \mathcal{P},\ \forall v \in \mathcal{V}(P), \tag{3.12}$$

$$y_l \in \{0,1\} \qquad \forall l \in L(\mathcal{P}). \tag{3.13}$$

Constraint (3.7) gives the value of the PWL function as a convex combination of the function values in the vertices of the discretization. Constraint (3.8) represents the function input as a convex combination of the vertices of the discretization. (3.9) and (3.12) enforce that the coefficients form a convex combination. A polytope P_i can be chosen if for $B(P_i) = (b_1, \dots, b_{|\mathcal{P}|})$ the variable values of the binary variables are set to $y_1 = b_1, \dots, y_{|\mathcal{P}|} = b_{|\mathcal{P}|}$. This is enforced by the constraints (3.10) and (3.11). As a consequence, a polytope $P \in \mathcal{P}$ is selected if for all $l \in L(\mathcal{P})$ the choice variables y_l are set to the l-th coordinate of $B(P)$.

3.3. Convex Combination Models (CC)

Similar to DCC and DLOG, the following two models are based on convex combinations. These models have been studied extensively, among others in [Lowe, 1984], [Jeroslow and Lowe, 1984] and in [Wilson, 1998]. The difference to DCC is, that instead of having one variable for each combination of polytopes and their vertices we only have one coefficient variable λ_v for each vertex $v \in \mathcal{V}(\mathcal{P})$. For this model, we also introduce decision variables

$y_P \in \{0,1\}$ for each polytope $P \in \mathcal{P}$. We refer to the model (3.14)-(3.20) as CC.

$$\sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v f(v) = z, \tag{3.14}$$

$$\sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v v = x, \tag{3.15}$$

$$\sum_{v \in \mathcal{V}(\mathcal{P})} \lambda_v = 1, \tag{3.16}$$

$$\sum_{P \in \mathcal{P}} y_P = 1, \tag{3.17}$$

$$\lambda_v \leq \sum_{P \in \mathcal{P}(v)} y_P \qquad \forall v \in \mathcal{V}(\mathcal{P}), \tag{3.18}$$

$$\lambda_v \geq 0 \qquad \forall v \in \mathcal{V}(\mathcal{P}), \tag{3.19}$$

$$y_P \in \{0,1\} \qquad \forall P \in \mathcal{P}. \tag{3.20}$$

Similar to DCC and DLOG, constraint (3.14) describes the output of the PWL function as a convex combination of the value in the vertices. Constraint (3.15) represents the input x as a convex combination of vertices. Constraints (3.16) and (3.19) enforce that the coefficients λ_v are coefficients of a convex combination. Due to the vertex variables being aggregated, constraint (3.18) is needed, so only coefficients of vertices in the chosen polytope can be nonzero. Constraint (3.17) states that only one polytope can be chosen.

3.3.1. The Logarithmic Convex Combination Model (CCLOG)

The following model is the most complex one. The idea is based on the CC model, but similar to the DLOG model there is a trick to reduce the number of decision variables. The idea is to formulate a binary branching scheme for the coefficient variables. A complete description and further details are available in [Vielma and Nemhauser, 2008]. This model seems to outperform all the other models we tested. This observation matches the results from [Vielma et al., 2010]. Unfortunately, this model can only be used with certain types of triangulations (see definition 3.3.2), but for our purpose, this is not relevant, because we can easily discretize our domain according to given constraints.

Before we can formulate the model, we need to introduce the notion of a binary branching scheme.

Definition 3.3.1 (Binary Branching Scheme, [Vielma et al., 2010])**.** *A* binary branching scheme *is a family of bipartitions $\{L_s, R_s\}$ of the vertices $\mathcal{V}(\mathcal{P})$ indexed by a finite set S. Additionally the branching scheme needs to fulfill that for every $P \in \mathcal{P}$ we have $V(P) = \bigcap_{s \in S}(\mathcal{V}(\mathcal{P}) \setminus T_s)$, where $T_s = L_s$ or $T_s = R_s$ for each $s \in S$.*

The constraints (3.21)-(3.27) define the CCLOG model. Now for a given binary branching scheme $\{L_s, R_s\}_{s\in S}$, we can define the model:

$$\sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v f(v) = z, \tag{3.21}$$

$$\sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v v = x, \tag{3.22}$$

$$\sum_{v\in\mathcal{V}(\mathcal{P})} \lambda_v = 1, \tag{3.23}$$

$$\sum_{v\in L_s} \lambda_v \leq y_s \qquad \forall s \in S, \tag{3.24}$$

$$\sum_{v\in R_s} \lambda_v \leq (1 - y_s) \qquad \forall s \in S, \tag{3.25}$$

$$\lambda_v \geq 0 \qquad \forall v \in \mathcal{V}(\mathcal{P}), \tag{3.26}$$

$$y_s \in \{0, 1\} \qquad \forall s \in S. \tag{3.27}$$

The choice of a binary branching scheme is crucial to the size of the formulation. To achieve a logarithmic number of variables, we need to find a branching scheme with a logarithmic number of partitions. [Vielma and Nemhauser, 2011] present such a branching scheme, which is based on the so-called J_1 triangulation.

Definition 3.3.2 ("Union Jack"/J_1 Triangulation)**.** *The J_1 triangulation is defined on $D = [0, K]^n$ where $K \in \mathbb{N}$ is an even number. The vertices of J_1 are given via the set $V = \{0, \ldots, K\}^n$. Let $V_0 = \{v \in V \mid \forall i \in \{1, \ldots, n\} : v_i \text{ is odd } \}$ be the set of vertices with coordinates that are all odd, $\mathbb{S}_n$ be the set of permutations on n elements, and $e_i \in \mathbb{R}^n$ the i-th unit vector. For each element $(v_0, \pi, s) \in V^0 \times \mathbb{S}_n \times \{-1, 1\}^n$ we define $j_1(v_0, \pi, s)$ as the simplex with the vertices $\{y_i \mid 0 \leq i \leq n\}$ where $y_0 = v_0$ and $y_i = y_{i-1} + s_{\pi(i)} e_{\pi(i)}$. The triangulation is then given by*

$$\mathcal{P} = \left\{ j_1(v_0, \pi, s) \mid (v_0, \pi, s) \in V^0 \times \mathbb{S}_n \times \{-1, 1\}^n \right\}.$$

Examples of the J_1 triangulation can be seen in figures 3.2 and 3.3.

Now let us gather some insights into the structure of J_1 triangulations. First, we realize, that we do not have to restrict ourselves to a discretization of $[0, K]^n$, but J_1 can be generalized to other domains, as well.

Remark 3.3.3. *A domain*

$$\Omega = [a_1, b_1] \times \ldots \times [a_n, b_n]$$

with discretizations

$$D_i = \left\{ a_i = p_i^0 < p_i^1 < \ldots < p_i^K = b_i \right\}$$

can be discretized via $D = D_1 \times \ldots \times D_n$ in the same way as the standard J_1 triangulation from 3.1.1 using the bijection

$$\pi : D \to [0, K]^n$$
$$(p_1^{k_1}, \ldots, p_n^{k_n}) \mapsto (k_1, \ldots, k_n).$$

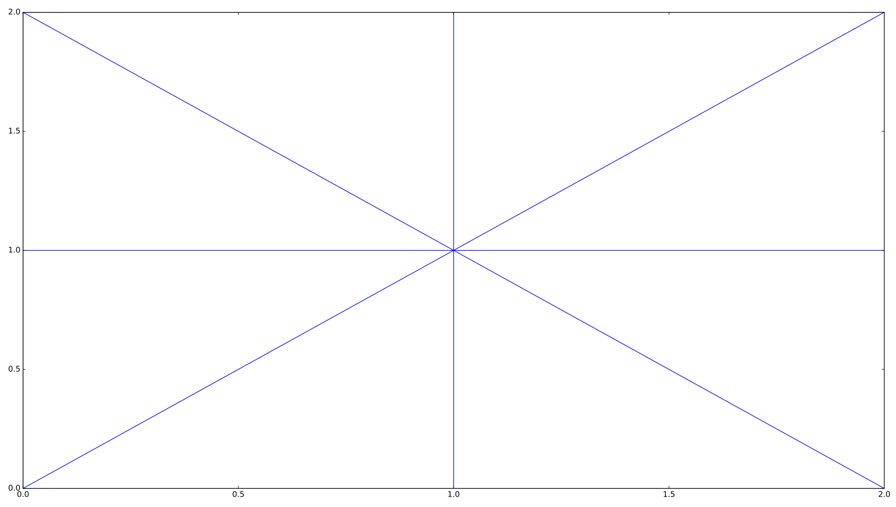


Figure 3.2.: Union-Jack triangulation of $D = [0,2]^2$

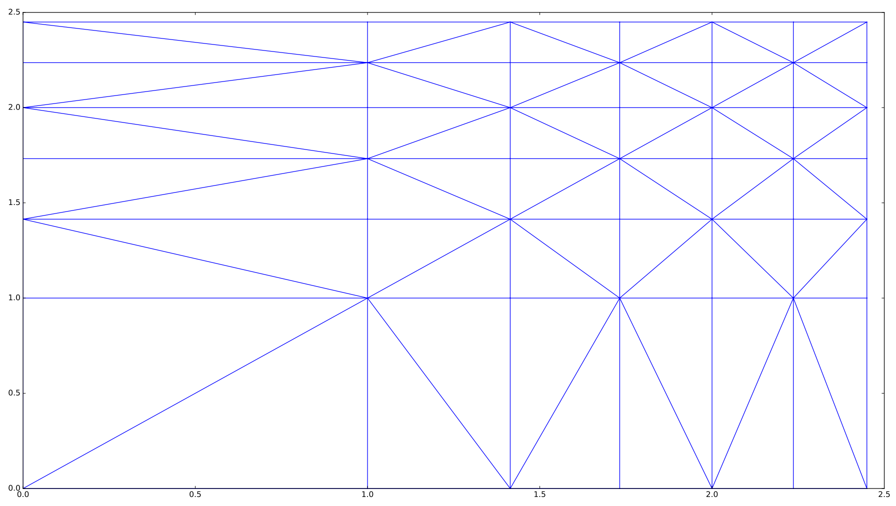


Figure 3.3.: Union-Jack triangulation of $D = [0,\sqrt{6}]^2$ with discretizations $\{0, 1, \sqrt{2}, \sqrt{3}, 2, \sqrt{5}, \sqrt{6}\}$ using Remark 3.3.3

A J_1 triangulation yields a binary branching scheme with a logarithmic number of bipartitions (i.e., the set S has a size logarithmic in the size of the triangulation).

Remark 3.3.4. *Let $\mathcal{P}$ be a J_1 triangulation of $[0,K]^n$. Let $(G^k)_{k=1}^K \subset \{0,1\}^{\lceil \log_2(K) \rceil}$ be a Gray binary code (see [Knuth, 2011]), i.e., a set of binary vectors s.t. G^k and G^{k+1} differ in at most one coordinate. Then we define the set*

$$O(l,b) := \left\{ k \in \{0,\dots,K\} \mid (k = 0 \vee G_l^k = b) \wedge (k = K \vee G_l^{k+1} = b) \right\}.$$

Let

$$S_1 := \{1,\dots,n\} \times \{1,\dots,\lceil \log_2(K) \rceil\},$$

and for $(s_1,s_2) \in S_1$ define

$$L(s_1,s_2) := \{v \in \mathcal{V}(\mathcal{P}) \mid v_{s_1} \in O(s_2,1)\},$$
$$R(s_1,s_2) := \{v \in \mathcal{V}(\mathcal{P}) \mid v_{s_1} \in O(s_2,0)\}.$$

Also, let

$$S_2 := \{(s_1,s_2) \mid s_1,s_2 \in \{1,\dots,n\} : s_1 < s_2\},$$

and for $(s_1,s_2) \in S_2$ define

$$L(s_1,s_2) := \{v \in \mathcal{V}(\mathcal{P}) \mid v_{s_1} \text{ is even } \wedge v_{s_2} \text{ is odd }\},$$
$$R(s_1,s_2) := \{v \in \mathcal{V}(\mathcal{P}) \mid v_{s_1} \text{ is odd } \wedge v_{s_2} \text{ is even }\}.$$

Then $\{L_s, R_s\}_{s \in S_1 \cup S_2}$ is a binary branching scheme for $\mathcal{P}$.

Proof. See [Vielma and Nemhauser, 2011] and [Vielma et al., 2010]. □

As a consequence with the branching scheme of remark 3.3.4, we can encode a piecewise linear function with a logarithmic number of binary variables.

3.4. Multiple Choice Model (MC)

The multiple choice model is also a very intuitive model. The idea is that for each polytope of the triangulation, we introduce a decision variable $y_P \in \{0,1\}$ and we add the linear inequalities that describe the polytope $P \in \mathcal{P}$. In contrast to the convex combination methods where we use the vertex representation of a polytope, here we use the half-space representation. The model has been studied in [Jeroslow and Lowe, 1984] and [Lowe, 1984]. Now, for any polytope $P \in \mathcal{P}$ we assume the following half-space representation:

$$P = \{x_P \in \mathbb{R}^n \mid A_P x_P \le b_P\}.$$

With this the MC model can then be given as:

$$\sum_{P\in\mathcal{P}}(m_P x_P + c_P y_p) = z, \tag{3.28}$$

$$\sum_{P\in\mathcal{P}} x_P = x, \tag{3.29}$$

$$A_P x_P \leq y_P b_P \qquad \forall P \in \mathcal{P}, \tag{3.30}$$

$$\sum_{P\in\mathcal{P}} y_P = 1 \qquad \forall P \in \mathcal{P}, \tag{3.31}$$

$$y_P \in \{0,1\} \qquad \forall P \in \mathcal{P}. \tag{3.32}$$

Constraint (3.28) describes the value of the PWL function as the function value of the linear function on the chosen polytope (i.e., the restriction of f onto the polytope). Constraint (3.30) achieves two things, first if some polytope $P \in \mathcal{P}$ is chosen (i.e., $y_P = 1$), then x_P is restricted to be a point in P. The second property is that for all the other polytopes $P' \in \mathcal{P}$, $P' \neq P$ that are not chosen $A_{P'}x_{P'} \leq 0$ holds. This implies that $x_{P'}$ is restricted to the null vector. If there were some solution $y_{P'}$, P' would not be bounded, because then for all $\delta > 0$, $A_{P'}\delta y_{P'} \leq b_P$. This is a contradiction to Ω being bounded (and thus also P' being bounded). Only one polytope can be chosen due to constraint (3.31).

3.5. Incremental Model (INC)

The so-called incremental model (also delta-method) has first occurred for the special case of one-dimensional functions in [Markowitz and Manne, 1957]. It has then been further extended and developed in [Wilson, 1998] to represent functions with multivariate input.

This formulation imposes an additional constraint on the triangulation of the domain.

Definition 3.5.1 (Ordering Properties for Triangulation). *The triangulation $\mathcal{P}$ needs to have the following properties, to be compatible with the incremental model for piecewise linear functions:*

1. *The simplices in $\mathcal{P}$ can be ordered as $P_1, \ldots, P_{|\mathcal{P}|}$ s.t. two consecutive simplices always share a point, i.e., $P_i \cap P_{i+1} \neq \emptyset$.*
2. *The vertices $V(P_i) = \left\{v_i^0, \ldots, v_i^{|V(P_i)|-1}\right\}$ of each simplex P_i can be ordered s.t. the last vertex of P_i is the first vertex of P_{i+1}, i.e.,*
$$v_i^{|V(P_i)|-1} = v_{i+1}^0.$$

Now we want to show that the J_1 triangulation has the desired properties. We start with an insight into the structure of the J_1 triangulation and realize that a large J_1 triangulation consists of smaller J_1 triangulations.

Remark 3.5.2. *A J_1 triangulation of $[0,K]^n$ can be described as the union of J_1 triangulations of $[0,2]^n$ that are shifted around the odd vertices V_0.*

Proof. This follows immediately from the definition of J_1. For any fixed $v_0 \in V_0$, we can see that for all $(\pi, s) \in \mathbb{S}_n \times \{-1,1\}^n$

$$j_1(v_0, \pi, s) = v_0 + \left\{ \sum_{i=1}^{k} s_{\pi(i)} e_i \mid k \in \{0, \dots, n\} \right\}.$$

In particular J_1 of $[0,2]^n$ is

$$J_1([0,2]^n) = (1, \dots, 1) + \left\{ \left\{ \sum_{i=1}^{k} s_{\pi(i)} e_i \mid k \in \{0, \dots, n\} \right\} \mid (\pi, s) \in \mathbb{S}_n \times \{-1,1\}^n \right\},$$

and

$$\begin{aligned}
&\{j_1(v_0, \pi, s) \mid (\pi, s) \in \mathbb{S}_n \times \{-1,1\}^n\} \\
=&\left\{ v_0 + \left\{ \sum_{i=1}^{k} s_{\pi(i)} e_i \mid k \in \{0, \dots, n\} \right\} \mid (\pi, s) \in \mathbb{S}_n \times \{-1,1\}^n \right\} \\
=&(v_0 - (1, \dots, 1)) \\
&\quad \underbrace{+ (1, \dots, 1) + \left\{ \left\{ \sum_{i=1}^{k} s_{\pi(i)} e_i \mid k \in \{0, \dots, n\} \right\} \mid (\pi, s) \in \mathbb{S}_n \times \{-1,1\}^n \right\}}_{J_1([0,2]^n)}.
\end{aligned}$$

□

With the additional information on the structure of the J_1 triangulation, we can give an explicit algorithm to order the simplices according to the ordering properties from Definition 3.5.1.

Lemma 3.5.3. *The two-dimensional J_1 triangulation fulfills the ordering properties from Definition 3.5.1.*

Proof. We give an explicit construction of the ordering of J_1. From remark 3.5.2, that J_1 on $[0, K^2]$ consists of J_1 on $[0,2]^2$ around the odd vertices (V_0 in definition 3.1.1). This means that if we find a path going from sub-triangulation to sub-triangulation that already has the ordering properties (1 and 2), we have constructed an ordering of the complete triangulation. A traversal of the complete triangulation can be achieved with the vertex sequence given in figure 3.4. To achieve the pattern defined by figure 3.4, we only need to show that in $J_1([0,2]^2)$ we can go from the bottom left corner to the top left corner and from the bottom left corner to the upper right corner while abiding by the ordering properties. An explicit construction is given for these two cases in Figures A.1 and A.2 in Appendix A. The numbers at the points denote their absolute order in the sub-triangulation, and the numbers in the triangles correspond to the order in which the

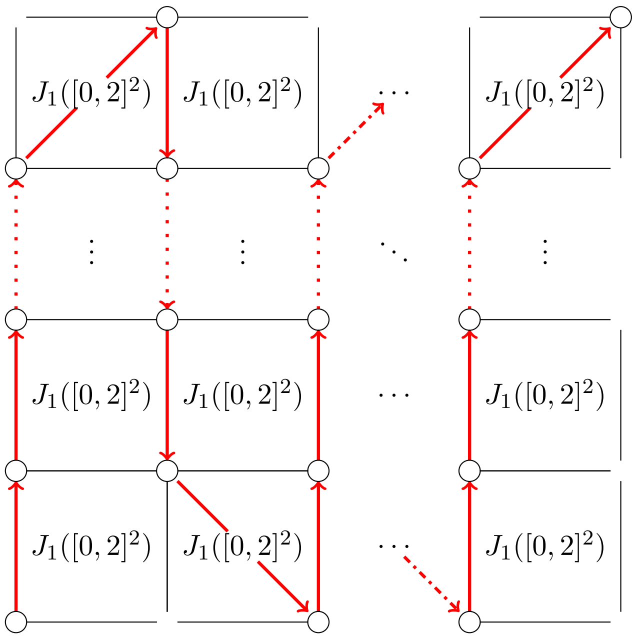


Figure 3.4.: Path through J_1 Triangulation with n^2 sub-triangulations

triangles are visited. The order for the sub-triangulation is then given via the triangles induced by the vertices (1, 2, 3), (3, 4, 5), (5, 6, 7), (7, 8, 9), (9, 10, 11), (11, 12, 13), (13, 14, 15), (15, 16, 17). The definition of these triangles directly implicates that the ordering property holds for the given triangulation. □

$$f(v_1^0) + \sum_{i=1}^{|\mathcal{P}|} \sum_{j=1}^{|V(P_i)|-1} \delta_i^j (f(v_i^j) - f(v_i^0)) = z, \tag{3.33}$$

$$v_1^0 + \sum_{i=1}^{|\mathcal{P}|} \sum_{j=1}^{|V(P_i)|-1} \delta_i^j (v_i^j - v_i^0) = x, \tag{3.34}$$

$$\sum_{j=1}^{|V(P_1)|-1} \delta_1^j \leq 1, \tag{3.35}$$

$$y_i \leq \delta_i^{|V(P_i)|-1}, \tag{3.36}$$

$$\sum_{j=1}^{|V(P_{i+1})|-1} \delta_{i+1}^j \leq y_i, \tag{3.37}$$

$$\delta_i^j \geq 0 \qquad \forall i \in \{1, \dots, |\mathcal{P}|\}, \; \forall j \in \{1, \dots, |V(T_i)|-1\} \tag{3.38}$$

$$y_i \in \{0,1\} \qquad \forall i \in \{1, \dots, |\mathcal{P}|-1\}. \tag{3.39}$$

Model	Constraints	Additional Variables	Binaries
DCC	$n + \lvert\mathcal{P}\rvert + 2$	$\lvert\mathcal{P}\rvert + \sum_{P\in\mathcal{P}} \lvert V(P)\rvert$	$\lvert\mathcal{P}\rvert$
DLOG	$n + 2\lceil\log_2 \lvert\mathcal{P}\rvert\rceil + 2$	$2\lceil\log_2 \lvert\mathcal{P}\rvert\rceil + \sum_{P\in\mathcal{P}} \lvert V(P)\rvert$	$2\lceil\log_2(\lvert\mathcal{P}\rvert)\rceil$
CC	$n + 3 + \lvert\mathcal{V}(\mathcal{P})\rvert$	$\lvert\mathcal{V}(\mathcal{P})\rvert + \lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}\rvert$
CCLOG	$n + 2 + 2\lvert S\rvert$	$\lvert\mathcal{V}(\mathcal{P})\rvert + \lvert S\rvert$	$\lvert S\rvert$
MC	$n + 2 + \sum_{P\in\mathcal{P}} F(P)$	$(n+1)\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}\rvert$
INC	$1 + 2\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}\rvert - 1 + \sum_{P\in\mathcal{P}}(\lvert V(P)\rvert - 1)$	$\lvert\mathcal{P}\rvert - 1$

Table 3.1.: Number of variables and constraints for the different piecewise linear models [Vielma et al., 2010, Table 1]

3.6. Replacing the Friction Term

This section is supposed to give a short explanation on how to use piecewise linear formulations described in this chapter to approximate the friction term (2.17) for a single pipe $a = (u, v)$ for some timepoint $t_i \in \mathcal{T}$. There are two different friction terms in the discretized momentum equation (2.16). We replace both terms with the variables f^{in}_{a,t_i}, and f^{out}_{a,t_i}. This gives us a new momentum equation

$$\left(1 + \frac{gsL_a}{2R_sTz}\right) p_{v,t_i} - \left(1 - \frac{gsL_a}{2R_sTz}\right) p_{u,t_i} + f^{in}_{a,t_i} + f^{out}_{a,t_i} = 0. \tag{3.40}$$

Then let $D_{p_u} = \left\{\underline{p}_u = p_1, \ldots, p_k = \overline{p}_u\right\}$ be a discretization of the pressure domain for the node u, $D_{p_v} = \left\{\underline{p}_v = p_1, \ldots, p_k = \overline{p}_v\right\}$ be a discretization of the pressure domain for the node v, and $D_{q_a} = \left\{\underline{q}_a = q_1, \ldots, q_k = \overline{q}_a\right\}$ be a discretization of the flow domain for the pipe a. Now $\mathcal{P}^{in}$ needs to be a triangulation for the set $D_{p_u} \times D_{q_a}$ and $\mathcal{P}^{out}$ needs to be a triangulation for the set $D_{p_v} \times D_{q_a}$. The linear functions that compose the PWL function are then defined by the values of the friction term on the vertices of each polytope in the domain discretizations.

Then for some type of PWL formulation (e.g. CCLOG) we add constraints with output variable f^{in}_{a,t_i}, input variables p_{u,t_i} and q^{in}_{a,t_i} with the discretization $\mathcal{P}^{in}$, and another set of constraints with output variable f^{out}_{a,t_i}, input variables p_{v,t_i} and q^{out}_{a,t_i} with the discretization $\mathcal{P}^{out}$.

4. Iterative Velocity Approximation

A common approach to solve nonlinear systems, is to find linearizations of the system (see for example [Kelley, 1995]) and successively solve the resulting linear systems. Newton's method is a well-known example of such an algorithm.

In this chapter, we will explore a specialized linearization approach for the discretized friction dominated model (2.15)-(2.16) on pipe only networks, i.e., $G = (V, A_{pi})$. We are trying to find approximations of the gas velocity which then yields a linear system. The idea is based on [Hennings, 2018]. We develop an iterative method and can show convergence on a single pipe under certain conditions on the parameters of the pipe and the flow scenario.

4.1. Formulating the Iterative Method

From the physical relations (2.9) and (2.10) for gas we infer that the gas velocity is proportional to the flow q divided by the pressure p:

$$v = \frac{R_s T z}{A} \frac{q}{p}.$$

By examining the friction term $f(p, q) = \frac{\lambda R_s T z}{2DA^2} \frac{|q| q}{p}$ we observe that we can substitute the velocity term. Resulting in the following expression

$$\hat{f}(v, q) = \frac{\lambda}{2DA} |v| \, q.$$

The function $\hat{f}$ is linear in q if we assume the velocity v to be constant. Thus by replacing the friction term $f(p, q)$ in the discretized momentum equation (2.16) with $f(v_C, q)$ for some fixed velocity v_C, only a linear system remains. [Hennings, 2018] suggests, that this approach only works as a very rough approximation and is especially unfit for situations in which the flow direction on a pipe changes. Still, this approach has several advantages: We can linearize the equations in a very natural way, and we get rid of the absolute value, which is not differentiable in 0. By fixing the *absolute* velocity, we can also preserve symmetry and the property that for zero flow on a pipe no friction is introduced.

Even though the results for a *constant* velocity estimate in the friction term are mixed, this has spawned the idea to approximate the velocity by solving the system (2.15)-(2.16), successively. In each step, we then update the absolute velocity with flow and pressure values from the previous iteration and solve again. We want to highlight, that in each *iteration* we solve (2.15)-(2.16) for *all timesteps* and then update the velocity (for all timesteps) and solve the updated (2.15)-(2.16) again.

For some pipe $a = (u, v) \in A_{pi}$ the idea is to start with some initial guess on the inflow $(q^{in}_{a,t_i})^{(0)}$, outflow $(q^{out}_{a,t_i})^{(0)}$ and pressure $p^{(0)}_{u,t_i}$ and $p^{(0)}_{v,t_i}$. Then for the k-th iteration we fix the velocity for all timesteps to a value calculated in the previous iteration. This results in the following system for $k > 0$:

$$\begin{aligned}
&p^{(k)}_{u,t_i} - p^{(k)}_{u,t_{i-1}} + p^{(k)}_{v,t_i} - p^{(k)}_{v,t_{i-1}} \\
&\quad + \frac{2R_sTz\Delta_t}{LA}\left((q^{in}_{a,t_i})^{(k)} - (q^{out}_{a,t_i})^{(k)}\right) &&= 0 \qquad \forall i = 1, \dots, n, && (4.1)\\
&\left(\frac{gsL}{2R_sTz} + 1\right) p^{(k)}_{v,t_i} + \left(\frac{gsL}{2R_sTz} - 1\right) p^{(k)}_{u,t_i} \\
&\quad + \frac{\lambda R_sTzL_a}{4D_aA_a^2}\left(\frac{\left|(q^{in}_{a,t_i})^{(k-1)}\right|}{p^{(k-1)}_{u,t_i}}(q^{in}_{a,t_i})^{(k)}\right. \\
&\qquad\qquad \left. + \frac{\left|(q^{out}_{a,t_i})^{(k-1)}\right|}{p^{(k-1)}_{v,t_i}}(q^{out}_{a,t_i})^{(k)}\right) &&= 0 \qquad \forall i = 1, \dots, n. && (4.2)
\end{aligned}$$

A single iteration of the algorithm described above is expressed in algorithm 1.

Algorithm 1: Iterative Velocity Approximation

Data: A pipe-only network $G = (V, A_{pi})$
Values from previous iteration:

$$\begin{aligned}
&p^{(k-1)}_{u,t_i} && \forall u \in V, && \forall t_i \in \{t_0, \dots, t_n\},\\
&(q^{in}_{a,t_i})^{(k-1)} \text{ and } (q^{out}_{a,t_i})^{(k-1)} && \forall a \in A_{pi}, && \forall t_i \in \{t_0, \dots, t_n\}.
\end{aligned}$$

Result: solution in the k-th iteration:

$$\begin{aligned}
&p^{(k)}_{u,t_i} && \forall u \in V, && \forall t_i \in \{t_0, \dots, t_n\},\\
&(q^{in}_{a,t_i})^{(k)} \text{ and } (q^{out}_{a,t_i})^{(k)} && \forall a \in A_{pi}, && \forall t_i \in \{t_0, \dots, t_n\}.
\end{aligned}$$

Solve the linear system consisting of the equations (4.1), (4.2) and (2.3) for $p^{(k)}_{u,t_i}$, $(q^{in}_{a,t_i})^{(k)}$, $(q^{out}_{a,t_i})^{(k)}$ and return the result.

Now we want to explore the theoretical properties of Algorithm 1.

4.2. Convergence Properties on a Single Pipe

In this section we want to prove convergence of the iterative method resulting from Algorithm 1. We restrict the analysis to networks that consist of a single pipe between to boundary nodes. We can prove convergence under some conditions on the pipe and the corresponding transient scenario.

If we only look at a single pipe $a = (u, v) \in A_{pi}$ between the boundary nodes u and v, there are no coupling conditions between other network elements, i.e., the flow conservation condition (2.3) disappears.

As a consequence, the inflow into a is always equal to the fixed demand at u and the outflow is equal to the fixed demand at v, i.e.,

$$\begin{aligned} q^{in}_{a,t_i} &= q^{nom}_{u,t_i} && \forall i = 1, \ldots, n, \\ q^{out}_{a,t_i} &= q^{nom}_{v,t_i} && \forall i = 1, \ldots, n. \end{aligned}$$

So the system (2.15)-(2.16) reduces to

$$\begin{aligned} & p_{u,t_i} + p_{v,t_i} - p_{u,t_{i-1}} - p_{v,t_{i-1}} \\ & \quad + \frac{2R_sTz(t_i - t_{i-1})}{L_aA_a}(q^{nom}_{v,t_i} - q^{nom}_{u,t_i}) && = 0 \qquad \forall i = 1, \ldots, n, \qquad (4.3) \\ & \left(1 + \frac{gs_aL_a}{2R_sTz}\right) p_{v,t_i} - \left(1 - \frac{gs_aL_a}{2R_sTz}\right) p_{u,t_i} \\ & \quad + \frac{\lambda_aR_sTzL_a}{4D_aA_a^2}\left(\frac{\left|q^{nom}_{u,t_i}\right| q^{nom}_{u,t_i}}{p_{u,t_i}} + \frac{\left|q^{nom}_{v,t_i}\right| q^{nom}_{v,t_i}}{p_{v,t_i}}\right) && = 0 \qquad \forall i = 1, \ldots, n. \qquad (4.4) \end{aligned}$$

By performing a series of row additions, we can even give a simpler description of the nonlinear system for pipes, in which the multiple timestep case essentially reduces to the single timestep case.

Lemma 4.2.1. *The nonlinear system* (4.3)-(4.6) *can be simplified to*

$$\begin{aligned} & p_{u,t_i} + p_{v,t_i} - p_{u,t_0} - p_{v,t_0} && && (4.5) \\ & \quad + \sum_{k=1}^{i} \frac{2R_sTz(t_k - t_{k-1})}{L_aA_a}\left(q^{nom}_{v,t_k} - q^{nom}_{u,t_k}\right) && = 0 \qquad \forall i = 1, \ldots, n, \\ & \left(1 + \frac{gs_aL_a}{2R_sTz}\right) p_{v,t_i} - \left(1 - \frac{gs_aL_a}{2R_sTz}\right) p_{u,t_i} && && (4.6) \\ & \quad + \frac{\lambda_aR_sTzL_a}{4D_aA_a^2}\left(\frac{\left|q^{nom}_{u,t_i}\right| q^{nom}_{u,t_i}}{p_{u,t_i}} + \frac{\left|q^{nom}_{v,t_i}\right| q^{nom}_{v,t_i}}{p_{v,t_i}}\right) && = 0 \qquad \forall i = 1, \ldots, n. \end{aligned}$$

Proof. We can calculate (4.5) from (4.3) by elementary row operations. We show this by induction: This is already true for $i = 1$, by equation (4.3). For $i > 1$ we can then add equation (4.5) for the index $i - 1$ to (4.3) for the index i. □

Lemma 4.2.1 is surprising because it shows that the solutions p_{u,t_i} and p_{v,t_i} only depend the initial pressure p_{u,t_0} and p_{v,t_i} and are independent of the previous timestep.

For the remainder of section 4.2 We define the constants

$$a_i = \left(p_{u,t_0} + p_{v,t_0} - \sum_{k=1}^{i} \frac{2R_sTz(t_k - t_{k-1})}{L_aA_a} \left(q^{nom}_{v,t_k} - q^{nom}_{u,t_k} \right) \right), \tag{4.7}$$

$$b_u = \left(1 - \frac{gs_aL_a}{2R_sTz} \right), \tag{4.8}$$

$$b_v = \left(1 + \frac{gs_aL_a}{2R_sTz} \right), \tag{4.9}$$

$$c_{u,i} = \frac{\lambda R_sTzL_a}{4D_aA_a^2} \left| q^{nom}_{u,t_i} \right| q^{nom}_{u,t_i}, \tag{4.10}$$

$$c_{v,i} = \frac{\lambda R_sTzL_a}{4D_aA_a^2} \left| q^{nom}_{v,t_i} \right| q^{nom}_{v,t_i}.. \tag{4.11}$$

We can use these constants to give a more compact representation of the system (4.3)-(4.4). We rewrite the discretized continuity and momentum equation (4.5) and (4.6)

$$p_{u,t_i} + p_{v,t_i} = a_i \qquad \forall i = 1, \dots, n, \tag{4.12}$$

$$b_vp_{v,t_i} - b_up_{u,t_i} + \frac{c_{u,i}}{p_{u,t_i}} + \frac{c_{v,i}}{p_{v,t_i}} = 0 \qquad \forall i = 1, \dots, n. \tag{4.13}$$

The system (4.12)-(4.13) admits an explicit solution. We can solve the system by calculating the zero set of a polynomial of degree three. This implies that the system always has a solution (even at least one real solution), but it does not guarantee that the solution is physically possible (i.e., negative or imaginary pressure values).

Lemma 4.2.2. *For a pipe $a = (u, v) \in A_{pi}$ and timesteps $t_1, \dots, t_n$, and initial pressure values p_{u,t_0} and p_{v,t_0} the nonlinear system* (4.3)-(4.4) *has a solution $p^*_{u,t_1}, \dots, p^*_{u,t_n} \in [\underline{p}_u, \overline{p}_u]$ and $p^*_{v,t_1}, \dots, p^*_{v,t_n} \in [\underline{p}_v, \overline{p}_v]$ if for every $i = 1, \dots, n$ the polynomial*

$$\begin{aligned} P_i(x) := &(b_u + b_v)x^3 \\ &- (a_ib_u + 2\,a_ib_v)x^2 \\ &+ \left({a_i}^2 b_v - c_{u,i} + c_{v,i} \right) x \\ &+ a_ic_{u,i} \end{aligned}$$

has a real-valued zero $p^*_i \in [\underline{p}_u, \overline{p}_u]$ and additionally $a_i - p^*_i \in [\underline{p}_v, \overline{p}_v]$ holds.

Proof. We extend the momentum equation (4.13)

$$- b_u{p_{u,t_i}}^2 p_{v,t_i} + b_vp_{u,t_i}{p_{v,t_i}}^2 + c_{v,i}p_{u,t_i} + c_{u,i}p_{v,t_i} = 0. \tag{4.14}$$

Additionally, we rearrange the continuity equation (4.12) for timestep t_i

$$p_{v,t_i} = a_i - p_{u,t_i}, \tag{4.15}$$

and can then substitute p_{v,t_i} by equation (4.15) into (4.14), and get a polynomial in p_{u,t_i}:

$$\begin{aligned}
&(a_i - p_{u,t_i})^2 b_v p_{u,t_i} - (a_i - p_{u,t_i}) b_u {p_{u,t_i}}^2 + c_{v,i} p_{u,t_i} + (a_i - p_{u,t_i}) c_{u,i} \\
&\quad = (b_u + b_v) {p_{u,t_i}}^3 - (a_i b_u + 2\, a_i b_v) {p_{u,t_i}}^2 + \big({a_i}^2 b_v - c_{u,i} + c_{v,i}\big) p_{u,t_i} + a_i c_{u,i} \\
&\quad =: P(p_{u,t_i}).
\end{aligned}$$

The polynomial P has degree three, so we can find the zeros of P explicitly via Cardano's method (e.g. [Bosch, 2013]). If one of the solutions is in $p_{u,t_i} \in [\underline{p}_u, \overline{p}_u]$ for which holds that $a_i - p_{u,t_i} = p_{v,t_i} \in [\underline{p}_v, \overline{p}_v]$, then p_{u,t_i} and p_{v,t_i} are valid solutions for the subsystem. If this holds for every $i = 1, \ldots, n$ we have found a solution for the complete system. $\square$

Lemma 4.2.2 shows that for each timestep there are possibly three different solutions to the system (2.15)-(2.16). The following example shows that we can find flow and pressure values as well as length and diameter to construct a single pipe instance in which more than one solution is physically possible.

Example 4.2.3. *Consider the pipe $a = (u, v)$ with the following parameters*

$$\begin{aligned}
s_a &= 0, \\
L_a &= 14.4\, km, \\
D_a &= 39\, cm, \\
k_a &= 0.1\, mm, \\
\rightsquigarrow \lambda_a &\approx 0.14.
\end{aligned}$$

We consider a scenario with the following values:

$$\begin{aligned}
t_0 &= 0\, s, \\
t_1 &= 3600\, s, \\
p_{u,t_0} &= 45\, bar, \\
p_{v,t_0} &= 13.61\, bar, \\
q^{nom}_{u,t_1} &= 62\, kg/s, \\
q^{nom}_{v,t_1} &= 60\, kg/s.
\end{aligned}$$

With these values, the polynomial from Lemma 4.2.2 has the three solutions (rounded to two decimal places)

$$\begin{aligned}
x_0 &\approx -5.60, \\
x_1 &\approx 53.28, \\
x_2 &\approx 57.09.
\end{aligned}$$

Even for moderate pressure bounds, i.e., 1 bar as lower bound and 100 bar as upper bound, the system (2.15)-(2.16) *has two solutions within these bounds*

$$p_{u,t_1} \approx 53.28\,\textit{bar},$$
$$p_{v,t_1} \approx 16.57\,\textit{bar},$$

as well as

$$p_{u,t_1} \approx 57.09\,\textit{bar},$$
$$p_{v,t_1} \approx 12.76\,\textit{bar}.$$

For the single pipe case, we can easily calculate the exact solutions, but we want to find out something about the convergence properties of the iterative method 1. The plan is now to define the iteration as a fixed point iteration and then use the Banach fixed-point theorem to prove convergence. We start with a recapitulation of the Banach fixed-point theorem (see for example [Amann and Escher, 1998]).

Theorem 4.2.4 (Banach Fixed-Point Theorem)**.** *Let X be a non-empty complete metric space and $d_X(\cdot,\cdot)$ be the corresponding metric. Let $f : X \to X$ be a* contraction *on X, i.e., a function s.t. there exists a constant $L < 1$ s.t. for all points $x, y \in X$ it holds that*

$$d_X(f(x), f(y)) \leq L d_X(x, y).$$

Then f has a unique *fixed-point $x^* \in X$ and the iteration $x_n = f(x_{n-1})$ converges to x^* for all start values $x_0 \in X$.*

Proof. See [Amann and Escher, 1998]. □

To apply the theorem to our case, we first construct the actual iteration function. Consequently, we apply the iteration schema to the simplified system (4.5)-(4.6). This yields yet another system that we can explicitly solve for $p^{(k)}_{u,t_i}$ and $p^{(k)}_{v,t_i}$.

$$p^{(k)}_{u,t_i} + p^{(k)}_{v,t_i} = a_i \quad \forall i = 1, \dots, n, \tag{4.16}$$
$$b_v p^{(k)}_{v,t_i} - b_u p^{(k)}_{u,t_i} + \frac{c_{u,i}}{p^{(k-1)}_{u,t_i}} + \frac{c_{v,i}}{p^{(k-1)}_{v,t_i}} = 0 \quad \forall i = 1, \dots, n. \tag{4.17}$$

For fixed values of $p^{(k-1)}_{u,t_i}$ and $p^{(k-1)}_{v,t_i}$ the system (4.16)-(4.17) is linear and has a block diagonal structure. Each block ((4.16) and (4.17) for a *single* timestep) admits a unique solution. Thus the complete system (for all timesteps) also admits a unique solution.

Remark 4.2.5. *The solution for the linear system* (4.16)-(4.17) *is*

$$p^{(k)}_{u,t_i} = \frac{1}{2}\left(a_i b_v + \frac{c_{u,i}}{p^{(k-1)}_{u,t_i}} + \frac{c_{v,i}}{p^{(k-1)}_{v,t_i}}\right),$$
$$p^{(k)}_{v,t_i} = \frac{1}{2}\left(a_i b_u - \frac{c_{u,i}}{p^{(k-1)}_{u,t_i}} - \frac{c_{v,i}}{p^{(k-1)}_{v,t_i}}\right).$$

Proof. Use Gaussian elimination and the fact that $b_u + b_v = 2$ (see definition (4.7)). Also

$$a_i - \frac{a_i b_u}{2} = \frac{2a_i - a_i b_u}{2} = \frac{(b_u + b_v)a_i - a_i b_u}{2} = \frac{a_i b_v}{2}.$$

□

From remark 4.2.5 we can explicitly construct the iteration function that we need for Theorem 4.2.4. Now we want to define a function f_{pi} that maps a solution

$$p^{(k)} = (p_{u,t_1}^{(k)}, \dots, p_{u,t_n}^{(k)}, p_{v,t_1}^{(k)}, \dots, p_{v,t_n}^{(k)}) \in \mathbb{R}^{2n}$$

in the k-th iteration step to the next iterate $p^{(k+1)} \in \mathbb{R}^{2n}$. Therefore we define f_{pi} in terms of its components. We can construct them directly from 4.2.5. For any index $i = 1, \dots n$ we can define the two component functions

$$\begin{aligned} f_{2i-1}(x_1, \dots, x_{2n}) &:= \frac{1}{2}\left(a_i b_v + \frac{c_{u,i}}{x_{2i-1}} + \frac{c_{v,i}}{x_{2i}}\right), \\ f_{2i}(x_1, \dots, x_{2n}) &:= \frac{1}{2}\left(a_i b_u - \frac{c_{u,i}}{x_{2i-1}} - \frac{c_{v,i}}{x_{2i}}\right). \end{aligned}$$

This yields the iteration function

$$f_{pi}(x_1, \dots, x_{2n}) := \begin{pmatrix} f_1(x_1, \dots, x_{2n}) \\ \vdots \\ f_{2n}(x_1, \dots, x_{2n}) \end{pmatrix}.$$

The way we constructed the system (4.16)-(4.17) and f_{pi} implies that any solution p^* to the system (4.3)-(4.4) is a fixed point of the function f_{pi}.

Theorem 4.2.6. *Suppose there exists a solution*

$$p^* = (p_{u,t_1}^*, \dots, p_{u,t_n}^*, p_{v,t_1}^*, \dots, p_{v,t_n}^*) \in [\underline{p}, \overline{p}]^{2n}$$

for the nonlinear system (4.3)-(4.4) *and for every point* $p \in [\underline{p}, \overline{p}]^{2n}$ *it holds that*

$$f_{pi}(p) \in [\underline{p}, \overline{p}]^{2n}.$$

If the constant

$$l := \left(\max_{i=1,\dots,n} \frac{|c_{u,i}| + |c_{v,i}|}{\underline{p}^2}\right) \tag{4.18}$$

is less than 1, *(i.e.,* $l < 1$*) then the iteration* $p^{(k)} = f_{pi}(p^{(k-1)})$ *converges to* p^* *for every* $p^{(0)} \in [\underline{p}, \overline{p}]^{2n}$.

Proof. We want to show that f is a contraction. We can estimate Lipschitz parameters for the component functions f_{2i-1} and f_{2i} on the interval $[\underline{p}, \overline{p}]$. For this let $x_1, \dots, x_{2n}, y_1, \dots, y_{2n} \in [\underline{p}, \overline{p}]$. Then for all $i = 1, \dots, n$ the following inequalities hold for the functions f_{2i-1}:

$$\begin{aligned}
&|f_{2i-1}(x_1, \dots, x_{2n}) - f_{2i-1}(y_1, \dots, y_2n)| \\
&\quad = \frac{1}{2}\left|\frac{c_{u,i}}{x_{2i-1}} + \frac{c_{v,i}}{x_{2i}} - \frac{c_{u,i}}{y_{2i-1}} - \frac{c_{v,i}}{y_{2i}}\right| \\
&\quad = \frac{1}{2}\left|\frac{c_{u,i}y_{2i-1} - c_{u,i}x_{2i-1}}{x_{2i-1}y_{2i-1}} + \frac{c_{v,i}y_{2i} - c_{v,i}x_{2i}}{x_{2i}y_{2i}}\right| \\
&\quad \le \frac{1}{2}\left|\frac{c_{u,i}(y_{2i-1} - x_{2i-1})}{x_{2i-1}y_{2i-1}}\right| + \left|\frac{c_{v,i}(y_{2i} - x_{2i})}{x_{2i}y_{2i}}\right| \\
&\quad \le \frac{1}{2}\left(\frac{|c_{u,i}|}{\underline{p}^2}|x_{2i-1} - y_{2i-1}| + \frac{|c_{v,i}|}{\underline{p}^2}|x_{2i} - y_{2i}|\right) \\
&\quad \le \frac{|c_{u,i}| + |c_{v,i}|}{\underline{p}^2}\left\|(x_1, \dots, x_{2n}) - (y_1, \dots, y_{2n})\right\|_\infty .
\end{aligned}$$

The same bound can be proven for the functions f_{2i} in a similar way. In total we can then estimate the Lipschitz parameter for f:

$$\begin{aligned}
&\|f(x_1, \dots, x_{2n}) - f(y_1, \dots, y_{2n})\|_\infty \\
&\quad \le \underbrace{\left(\max_{i=1,\dots,n}\frac{|c_{u,i}| + |c_{v,i}|}{\underline{p}^2}\right)}_{l:=}\|(x_1, \dots, x_{2n}) - (y_1, \dots, y_{2n})\|_\infty .
\end{aligned}$$

If the parameter l is less than 1, f is a contraction. Together with $X = [\underline{p}, \overline{p}]^{2n}$ and $d(x, y) = \|x - y\|_\infty$ we can use Theorem 4.2.4 to show that the iteration converges. ☐

The result of Theorem 4.2.6 is useful in two different ways: The estimate we can show is only influenced by the pipe parameters, a minimum pressure estimate and the flow demands at the end of the pipe and in particular *independent* on the initial pressure values p_{u,t_0}, p_{v,t_0}. Thus we can make relatively general statements about convergence for a large class of scenarios. Therefore we can solve (4.18) for the length L_a. This can then serve as a bound on the length of a pipe for which algorithm 1 is guaranteed to converge. For a horizontal pipe and a single timestep we can calculate the bound via the following formula:

$$L_{\max}(D) = \frac{\underline{p}^2\pi^2 D^5}{4\lambda_a R_s T z} \cdot \frac{1}{(q_{u,t_1}^{nom})^2 + (q_{v,t_1}^{nom})^2}. \tag{4.19}$$

An exemplary result is shown in Figure 4.1.

Due to the high exponent of D in (4.19), it is not surprising, that the possible length of a pipe rapidly increases with increasing diameter. In pipes with a large diameter, there is much less friction.

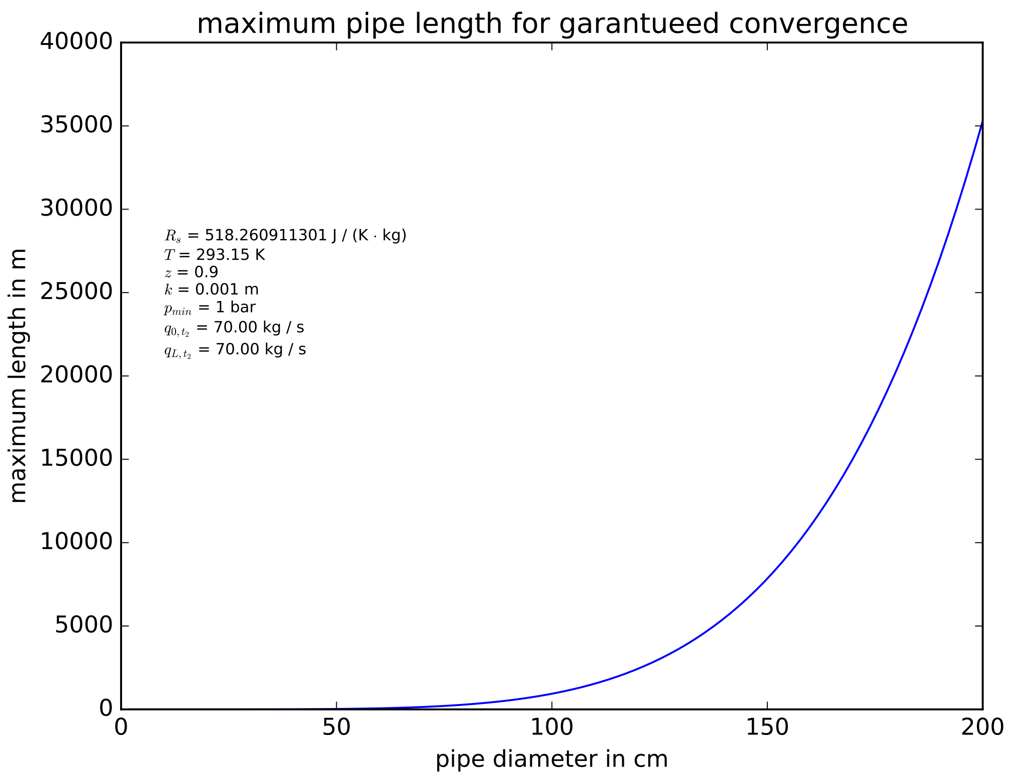


Figure 4.1.: Maximum length L of a pipe as a function of the diameter D s.t. Algorithm 1 is guaranteed to convergence.
The physical parameters used for the calculation are given in the figure.

	minimum	maximum	average	median
Length (km)	0.01	173.66	3.99	0.66
Diameter (cm)	15	210	41.41	30
Roughness (mm)	0.008	1	0.098	0.1

Table 4.1.: Average, Median and Extreme Values of Pipe Parameters from GasLib-11, GasLib-24, GasLib-40, GasLib-134, GasLib-135, GasLib-582 and GasLib-4197 Networks.

We calculated minimum, maximum, median and average for pipe parameters from the GasLib networks. The results are listed in Table 4.1.

Table C.1 in Appendix C shows Lipschitz parameters for different combinations of the values from Table 4.1 with a minimum pressure estimate of 1 bar. We only include combinations of length, diameter, and roughness that admit a stationary solution for a flow of 70 kg/s with pressure values between 1 bar and 100 bar. For example, a pipe that is 100km long and has a diameter of 15cm cannot transport 70 kg/s of gas with a maximum pressure of 100 bar.

We can see that the diameter of the pipe has the biggest influence, and thus we can only guarantee convergence for very short pipes or pipes with a relatively large diameter. Another insight of the formula (4.19) is that the pressure estimate has a quadratic influence on the Lipschitz constant, thus having good lower bounds on possible pressure values is important if we want to guarantee convergence. Overall one of the implications of Theorem 4.2.6 is that the algorithm shows vastly better convergence behaviour in high-pressure scenarios.

4.3. Properties of the Linear System for Paths

By coupling several pipes on a path, we can examine a more extensive linear system. Ideally we should be able to prove convergence theorems for pipes on a path, but unfortunately, no approach has worked for us so far. Still, we want to share the insights into the structure of the nonlinear system, that we gathered.

We define the path P_n on n nodes as: Let $P_n = (V, A)$ where $V = \{0, \dots, n\}$ and $A = A_{pi} = \{(i-1, i) \mid i \in V, i > 0\}$.

We are dealing with a very specific case of the system here. We only examine a *single timestep* on a path with *only horizontal* pipes. In order to make the nonlinear systems less overloaded with parameters and indices, we introduce a slightly simplified notation for this special case. Having at most one ingoing and one outgoing pipe per node results in the coupling conditions

$$q^{out}_{(i-1,i),t} = q^{in}_{(i,i+1),t} \quad \forall i \in V : 0 < i < n. \tag{4.20}$$

From these conditions we can introduce a single flow variable *per node*, which we will refer to as $q_{i,t}$, where $q_{0,t} = q^{in}_{(0,1),t}$ and $q_{n,t} = q^{out}_{(n-1,n),t}$.

So for paths on a single timestep $[t_0, t_1]$ with $\Delta_t = t_1 - t_0$ we we define the following constants

$$\begin{aligned} c_i &= \frac{2R_s T z \Delta_t}{L_{(i,i+1)} A_{(i,i+1)}}, \\ d_i &= p_{i,t_0} + p_{i+1,t_0}, \\ e_i &= \frac{\lambda_{(i,i+1)} L_{(i,i+1)}}{4D_{(i,i+1)} A_{(i,i+1)}}. \end{aligned}$$

The unsimplified equations are thus

$$\begin{aligned} p_{i,t_1} + p_{i+1,t_1} + c_i(q_{i+1,t_1} - q_{i,t_1}) - d_i &= 0, \\ p_{i+1,t_1} - p_{i,t_1} + e_i \left(\frac{|q_{i,t_1}|\, q_{i,t_1}}{p_{i,t_1}} + \frac{|q_{i+1,t_1}|\, q_{i+1,t_1}}{p_{i+1,t_1}} \right) &= 0. \end{aligned}$$

With $v_i^{(k)} = \frac{\left|q_{i,t_1}^{(k)}\right|}{p_{i,t_1}^{(k)}}$ The iteration scheme results in the following system:

$$p_{i,t_1} + p_{i+1,t_1} + c_i(q_{i+1,t_1} - q_{i,t_1}) - d_i = 0, \tag{4.21}$$

$$p_{i+1,t_1} - p_{i,t_1} + e_i \left(v_i^{(k)} q_{i,t_1} + v_{i+1}^{(k)} q_{i+1,t_1} \right) = 0. \tag{4.22}$$

By adding (4.21) to (4.22) and subtracting (4.22) from (4.21) we end up with

$$2p_{i,t_1} + \left(-c_i - e_i v_i^{(k)}\right) q_{i,t_1} + \left(c_i - e_i v_{i+1}^{(k)}\right) q_{i+1,t_1} - d_i = 0, \tag{4.23}$$

$$2p_{i+1,t_1} + \left(-c_i + e_i v_i^{(k)}\right) q_{i,t_1} + \left(c_i + e_i v_{i+1}^{(k)}\right) q_{i+1,t_1} - d_i = 0. \tag{4.24}$$

Equation (4.24) for pipe $(i, i+1)$ can be subtracted from equation (4.23) for pipe $(i+1, i+2)$ which results in the tridiagonal system

$$\begin{aligned} & q_{0,t_1} && = q_{0,t_1}^{nom}, \\ & 2p_{0,t_1} + \left(-c_0 - e_0 v_0^{(k)}\right) q_{0,t_1} + \left(c_1 - e_1 v_1^{(k)}\right) q_{1,t_1} && = d_0, \\ & \left(-e_i v_i^{(k)} + c_i\right) q_{i,t_1} + \left(v_{i+1}^{(k)} (e_i - e_{i+1}) - c_i - c_{i+1}\right) q_{i+1,t_1}, && \\ & \quad \left(-e_{i+1} v_{i+2}^{(k)} + c_{i+1}\right) q_{i+2,t_1} && = d_{i+1} - d_i \; \forall i \in V, \\ & q_{n,t_1} && = q_{n,t_1}^{nom}. \end{aligned}$$

The simplified system can then be solved with the tridiagonal-matrix-algorithm to speed up the solution method.

5. Computational Results

In this chapter, we want to compare both the speed and the accuracy of the different solution approaches that we introduced in the chapters 3 and 4. We test different approximations for the pipe flow system (2.15)-(2.16). The networks and the corresponding transient scenarios are described in Appendix B.

Before we start to present the computational results, we discuss how to obtain an initial state for the demand scenarios.

5.1. Calculating an Initial State

We need a valid initial state for the transient system, so we chose to calculate a so-called stationary state (also called "steady state") for the network. The pressure and flow in a steady state system are fixed in such a way that the network can supply the flow demands for an unlimited amount of time (see [Koch et al., 2015, 1.2]). In a stationary calculation, we want to find flow variables q_a for $a \in A$ and pressure values p_u for nodes $u \in V$.

For our transient scenario, we want to have an initial state from which we can both increase or decrease pressure variables. Therefore we need to find a stationary solution which enables this. As a consequence, we decided to use an objective function that maximizes the minimal distance of the pressure variables to their pressure bounds, i.e.,

$$\text{maximize } \min_{u \in V} \min \left(p_u - \underline{p}_u, \overline{p}_u - p_u \right). \tag{5.1}$$

[Humpola et al., 2015] describe a constraint programming formulation for the existence of stationary flow in passive pipe networks. By adding short cut constraints and the objective function (5.1) (which maximizes pressure slack), we can extend their formulation to our use-case. For a pipe $a = (u, v) \in A_{pi}$ we then define the following parameters

$$\begin{aligned}
\Lambda_a &= \frac{\lambda_a R_s T z L_a}{A_a^2 D_a}, \\
S_a &= \frac{2 g s_a L_a}{R_s T z}, \\
\alpha_a &= \begin{cases} \Lambda_a \frac{e^{S_a} - 1}{S_a}, & \text{if } S_a \neq 0 \\ \Lambda_a & \text{otherwise} \end{cases}, \\
\beta_a &= e_a^S.
\end{aligned}$$

A stationary solution for a passive network (all elements are open or in bypass mode) can then be obtained by solving the following program:

$$\text{maximize} \min_{u\in V} \min\left(p_u - \underline{p}_u, \overline{p}_u - p_u\right) \tag{5.2}$$

$$\text{s.t.} \sum_{a\in\delta^+(u)} q_a - \sum_{a\in\delta^-(u)} q_a = q^{nom}_{u,t_0} \qquad \forall u \in V, \tag{5.3}$$

$$p_v^2 - \beta_a p_u^2 - \alpha_a |q_a| q_a = 0 \qquad \forall a = (u,v) \in A_{pi}, \tag{5.4}$$

$$p_u - p_v = 0 \qquad \forall a = (u,v) \in A \setminus A_{pi}, \tag{5.5}$$

$$\underline{p}_u \leq p_u \leq \overline{p}_u \qquad \forall u \in V, \tag{5.6}$$

$$\underline{q}_a \leq q_a \leq \overline{q}_a \qquad \forall a \in A. \tag{5.7}$$

We then perform the stationary calculations on a passive network (i.e., short cut constraints for active elements) with a minimum pressure $\underline{p}_u$ of 1 bar and a maximum pressure $\overline{p}_u$ of 100 bar for each node $u \in V$. On further details on how to do stationary calculations see [Koch et al., 2015], which contains extensive material on this and closely related topics (nomination-validation, gas network capacities, etc.).

A solution to the stationary problem (5.2)-(5.7) then gives pressure values p_u for each node $u \in V$ and thus defines the initial conditions $p_{u,t_0} = p_u$ for the transient scenario.

5.2. Convergence Results for Iterative Velocity Approximation

The purpose of this section is to explore convergence properties of Algorithm 1 experimentally. We created a few simple test instances and also tested on publicly available networks. Details and visual representations of the networks can be found in Appendix B.

For some of the more complex instances we did not manage to find an explicit solution to the problem, so we have to find another way to measure the quality of an approximate solution to the system (2.15)-(2.16). A common approach is to measure how well a solution in each iteration "fulfills" the constraints, i.e., we measure the values that we get by plugging in a solution of the iteration scheme into the original discretized equations using the correct nonlinear velocity in the friction term. What then remains is essentially the value of the constraint violation. [Kelley, 1995] calls this the nonlinear residual.

As an example let us look at some pipe $a = (u,v) \in A_{pi}$, then the solution after the k-th iteration yields some pressure and flow values $p^{(k)}_{u,t_i}$, $p^{(k)}_{v,t_i}$, $(q^{in})^{(k)}_{a,t_i}$ and $(q^{out})^{(k)}_{a,t_i}$. The residual for the momentum equation for this pipe at timestep t_i is defined as:

$$\begin{aligned} r_{a,t_i} := &\left(\frac{gsL}{2R_sTz} + 1\right) p^{(k)}_{v,t_i} + \left(\frac{gsL}{2R_sTz} - 1\right) p^{(k)}_{u,t_i} \\ &+ \frac{\lambda R_s T z L_a}{4 D_a A_a^2} \left(\frac{\left|(q^{in}_{a,t_i})^{(k)}\right|}{p^{(k)}_{u,t_i}} (q^{in}_{a,t_i})^{(k)} + \frac{\left|(q^{out}_{a,t_i})^{(k)}\right|}{p^{(k)}_{v,t_i}} (q^{out}_{a,t_i})^{(k)} \right). \end{aligned} \tag{5.8}$$

The residuals for the other equations (flow conservation and discretized continuity equation) are less interesting because both these equations are linear and linear system solvers tend to return solutions with a negligible residual for linear constraints. In our experiments, we do compare all of the residuals, but we have found that the residual of the discretized momentum equation always yields the highest residual. To estimate how well Algorithm 1 performs, we measure the maximum of all the residuals of the momentum equation

$$r_{\max} = \max_{a \in A_{pi}, t_i \in \mathcal{T}} |r_{a,t_i}|. \tag{5.9}$$

5.2.1. Pipes

In the previous sections, we have proved that under certain circumstances the iteration described in algorithm 1 converges on a single pipe (see Theorem 4.2.6). This theorem only gives an upper bound on the Lipschitz constant, so we are interested in cases in which these bounds are particularly bad, and we want to see if the iteration still converges. Due to lemma 4.2.1 we only need to look at scenarios with one timestep, as each timestep is independent of the one before.

So we tested the iteration on three different pipes, an "average pipe" (i.e., average values from pipes in the GasLib, compare Table 4.1) with 3.99 km length 41.41 cm diameter and a roughness of 0.098 mm, with a Lipschitz constant of 253.88 (compare Table C.1 in Appendix C) and a "hard pipe" with 3.99 km length, 30 cm diameter and roughness 1 mm which yields a Lipschitz constant of 2402.59 (again compare Table C.1 in Appendix C), as well as the pipe defined by the parameters in Example 4.2.3.

For the special case of single pipes we do not need to measure the residual, but can instead give the difference to the explicit solution (see Lemma 4.2.2). We only simulate one timestep on a single pipe $a = (u, v)$, so the only variables that we need to find are p_{u,t_1} and p_{v,t_1}. The continuity equation (2.15) gives a linear relation between p_{u,t_1}, and p_{v,t_1}. So for an exact solution p^*_{u,t_1}, p^*_{v,t_1} and an intermediate solution $p^{(k)}_{u,t_1}, p^{(k)}_{v,t_1}$ it suffices to measure

$$d_a^{(k)} := \left| p^*_{u,t_1} - p^{(k)}_{u,t_1} \right|.$$

Figures C.1 to C.6 in Appendix C show the results of the calculations. Each figure additionally contains the parameters used in the iteration (the pressure at the start node, the pressure at end the node, the inflow, the outflow and the start values for the iteration, as well as, the values of the exact solution).

It is interesting to see that for all different start values that we tried (even for negative ones) the algorithm converges on the average as well as on the hard pipe. What becomes evident from the plot C.5 is that intermediate solutions in the iteration process might lie outside of the designated pressure bounds for the pipe. If the iteration is included within a linear program in which there are bounds on pressure and flow this might lead to the infeasibility of intermediate programs. Thus it makes sense to remove these bounds on the variables for the iteration itself.

Figure C.6 shows the result for a pipe, where several solutions are feasible. Initially, the distance to a solution increases until it reaches a tipping point at the 13-th iteration. We suspect that this is related to the fact that two solutions of the nonlinear system are close together (see Example 4.2.3).

All in all, convergence on a single pipe works well for pipes with realistic values, even for instances with large Lipschitz constants.

5.2.2. Instances with multiple pipes

For instances with multiple pipes we are not able to calculate an exact solution, so we plot the maximum residual $r_{\max}$ for each intermediate solution and additionally plot the maximum difference between subsequent iterations. Details on the instances and the corresponding scenarios can be found in Appendix B. The plots for the results can be seen in Figures C.7 to C.15c which are in Appendix C.

The initial conditions (i.e., pressure values p_{u,t_0} for all $u \in V$) are calculated by calculating a stationary flow in the passive network (see Appendix B for the details). In all of the calculations, the number of iterations varies, as the iteration was performed until there was a cycle in the sequence of solutions, i.e., the solution did not change any more, or a previous solution was reached again. The residual is measured for the discretized momentum equation (2.16) and is measured in Pascal. The model is implemented using the Pyomo 5.5.1 Python package [Hart et al., 2017]. The resulting series of linear programs are then solved with CPLEX 12.8 [IBM ILOG CPLEX Division, 2017]. The initial values for the iterative method are obtained by extending the flow and pressure values from a stationary calculation to all timesteps, i.e., if for each node $u \in V$ the pressure values p_u and for each arc $a \in A$ the flow values q_a are obtained from the stationary calculation, we set

$$p^{(0)}_{u,t_i} = p_u \qquad \forall u \in V, \quad \forall i = 1, \dots, n, \tag{5.10}$$

$$(q^{in})^{(0)}_{a,t_i} = q_a \qquad \forall a \in A_{pi}, \quad \forall i = 1, \dots, n, \tag{5.11}$$

$$(q^{out})^{(0)}_{a,t_i} = q_a \qquad \forall a \in A_{pi}, \quad \forall i = 1, \dots, n, \tag{5.12}$$

$$q^{(0)}_{a,t_i} = q_a \qquad \forall a \in A \setminus A_{pi}, \quad \forall i = 1, \dots, n. \tag{5.13}$$

We divide the test set into two different groups. The first group consists of networks designed by ourselves (`Path`, `Star`, `Cycle`, and `Tree`). We call these networks the "artificial instances". The other set of instances consists of networks from [Schmidt et al., 2017] and are therefore called "GasLib" instances. The GasLib instances contain the networks GasLib-11, GasLib-24, GasLib-40, GasLib-134 and GasLib-135.

Artificial Instances

Figures C.7 to C.10 in Appendix C show the convergence properties on the artificial networks. We observe that in small networks already after few iterations, the residual is in an acceptable range (less than $10^{-5} Pa = 10^{-10} bar$). In the cycle-free networks (`Path`, `Star`, `Tree`) all the pressure variables we can clearly see that the iteration converges, the difference between iterations becomes less than 10^{-9}. As a comparison [Stolwijk and Mehrmann, 2018] use an absolute tolerance of 10^{-3} as a stopping criterion for Newton's method.

What we also see is that for `Cycle`, the results do have an acceptable residual (less than $10^{-6}Pa$), but the iteration cycles between two solutions that are further apart than for cycle-free networks ($1.27 \cdot 10^{-6}$ compared to $1 \cdot 10^{-9}$). We suspect that there are two different solutions to the discretized system, that are very close together, so the iteration might not be able to converge. (We cannot prove this, as we do not have a symbolic solution for the discretized system).

GasLib Instances

GasLib-11, GasLib-24 (Figures C.11 to C.13b) in Appendix C start to cycle between solutions that are relatively close together (difference less than 0.3) but the residual of these solutions is higher than in the artificial instances. It is important to note that all of these networks do contain cycles.

The results for GasLib-40 are considerably worse than for the smaller networks. Again the solution cycle, but the difference is larger than 1. The residual is also larger than for all of the smaller networks.

Even though the GasLib-134 is the second largest network in our calculations, the results are better than for the smaller instances. A residual less than 10^{-5} can be achieved, and subsequent solutions are as close as $2.87 \cdot 10^{-7}$. The fact that the results are better than for the other instances might be influenced by the structure of the network. In contrast to the other GasLib instances, GasLib-134 does not contain any cycles. We suspect that the better performance of the iterative method is related to this.

The GasLib-135 instance (Figures C.15a and C.15c) shows the worst behaviour. The best residual that is achieved reached 65.34 is also relatively high compared to all the other instances. Particularly bad is that the iteration cycles between solutions that are far apart, the difference does not go below 5.54 even after 287 iterations.

5.3. Performance Results

We compare piecewise linear approximation and iterative velocity approximation to generic nonlinear solution approaches that are implemented by BARON [Tawarmalani and Sahinidis, 2005] and SCIP [Gleixner et al., 2018]. We use SCIP 6.0 which is compiled to use IPOPT [Wächter and Biegler, 2006] and CPLEX [IBM ILOG CPLEX Division, 2017].

For a piecewise linear approximation, replacing the friction term is described in 3.6. We test all of the PWL formulations with three discretization points in each dimension. The CCLOG 3.3.1 formulation is additionally tested with five and seven discretization points. The discretization points are equally spaced between the lower and the upper bound of the variable (the first point is equal to the lower bound and the last point equal to the upper bound). Both the INC and the CCLOG formulation use the J_1-triangulation to triangulate the domain. For the other formulations, we use a Delaunay triangulation generated by qdelaunay from qhull [Barber et al., 1996].

The third approach is to apply iterative velocity approximation. We have seen in section 5.2, that Algorithm 1 does not always converge and the residual can be bounded.

So instead of formulating a stopping criterion based on the residual or the difference between subsequent solutions, we choose to fix the number of iterations for the iterative method 1. From section 5.2 we conclude that ten iterations yield a good trade-off between solution time and solution accuracy.

Even though we do not have an objective to optimize, all of the models are solved by optimization software, because we want to test solution behaviour when we embed the approximations into an optimization problem. Therefore we define a *constant* objective function and hence solve a feasibility problem. The linear models are solved with CPLEX 12.8 [IBM ILOG CPLEX Division, 2017]. The computations were performed on Intel Xeon E3-1245 v3 CPU with 3.40GHz and 32 GB RAM. We run all calculations with a time limit of one hour (3600 seconds) and implement the different solution methods with the Pyomo Python package [Hart et al., 2017].

For a passive network $G = (V, A)$ with a transient scenario we work with a system composed of the constraints introduced in chapter 2:

$$\begin{aligned}
&\exists p, q^{in}, q^{out}, q && && (5.14)\\
\text{s.t. }& \text{discretized continuity equation (2.15)} & \forall a \in A_{pi}, & \quad \forall t \in \mathcal{T}, & (5.15)\\
& \text{discretized momentum equation (2.16)} & \forall a \in A_{pi}, & \quad \forall t \in \mathcal{T}, & (5.16)\\
& \text{short cut constraint (2.18)} & \forall a \in A \setminus A_{pi}, & \quad \forall t \in \mathcal{T}, & (5.17)\\
& \text{flow conservation constraint (2.3)} & \forall u \in V, & \quad \forall t \in \mathcal{T}, & (5.18)\\
& \text{flow bounds (2.1)} & \forall a \in A, & \quad \forall t \in \mathcal{T}, & (5.19)\\
& \text{pressure bounds (2.1)} & \forall a \in A, & \quad \forall t \in \mathcal{T}. & (5.20)
\end{aligned}$$

Each of the pressure and flow variable comes with an upper and a lower bound. These bounds tend to be far apart, so in order to achieve an acceptable approximation with the PWL models, we either need very fine discretizations (i.e., many points) or have very good bounds on the variable s.t. a coarse discretization still yields good results. The problem with fine discretizations is that the size of the formulations grows fast because the number of triangles in the discretization increases quadratically in the number of discretization points. So to get better bounds on the variables, we first solve the nonlinear system with a nonlinear solver and then apply new tighter bounds to the variables. To tighten the bounds we choose to reduce the lower bound for each variable by 10% of its value in the nonlinear solution value and the upper bound is increased by 10 % of its nonlinear solution value.

Our results are then obtained by solving the instances with the updated variable bounds. We then give the maximum nonlinear residual of each solution (see equation (5.9)). To give some relation between the solutions from different solution methods, we include a comparison with the result of the nonlinear solution process. Therefore we calculate the maximum of the relative difference

$$\Delta(x, y) = \frac{|x - y|}{\max(|x|, |y|)} \tag{5.21}$$

for each variable to the nonlinear solution

$$\left(\text{with solution values } p^{nl}_{u,t_i}, q^{nl}_{a,t_i}, (q^{in}_{a,t_i})^{nl}, (q^{out}_{a,t_i})^{nl}\right),$$

Constant	value
T	$10°C$
R_s	0.52 kJ/(kg · K)
z	0.9
ρ_0	0.78 kg/m^3
g	9.81 m/s^2

Table 5.1.: Physical Quantities for Transient Calculations

i.e.,

$$\Delta_{max} := \max_{x \in p_{u,t_i}, q_{a,t_i}, q^{in}_{a,t_i}, q^{out}_{a,t_i}} \Delta(x, x^{nl}). \tag{5.22}$$

We also include the solution time for each method.

The physical parameters used in the calculations can be found in table 5.1.

The results for all test scenarios are listed in the tables D.2 to D.10 in Appendix D.

5.3.1. Artificial Networks

The artificial networks (`Path`, `Star`, `Cycle`, `Tree`) are very small (less than ten pipes), so the corresponding optimization problems are also rather small, and the instances can easily be solved within less than a minute. The details of the solutions are summarized in tables D.2 to D.5 in Appendix D.

Solving the nonlinear problem with improved bounds is always the fastest approach for the small instances. Also in all of the instances, the residual of the nonlinear solution is the smallest by a large factor of at least 10^6. The nonlinear solution with improved bounds can improve the nonlinear residuals in three out of the four instances, and the solution is always extremely close to the solution without improved bounds. NL-SCIP solves the `Path`, `Cycle` and `Tree` instances with tightened bounds one magnitude faster than NL-BARON. The `Star` instance can be solved in the same magnitude as NL-BARON, but it shows similar behaviour to ITERATE, i.e., the solution is almost 10% different to the solution of NL-0. The residual of the SCIP solution is always among the smallest.

Let us examine the iterative velocity approximation results. In section 5.2 we have already seen that the solutions from the algorithm are acceptable w.r.t. their nonlinear residual. In the `Path`, `Tree` and `Cycle` instances the solutions are always extremely close to the nonlinear solution (less than 10^{-8}%), but in the `Star` instance, there is a difference of 9.76% which is almost the maximum difference we can see due to the improved bounds. This leads us to believe that the solution of the `Star` instance might not be unique. Solution times are always worse than the nonlinear solver.

The PWL models have the worst residuals, but the solutions still are acceptable, they never differ more than 1% to the nonlinear solution. We can also verify that increasing the number of discretization points always improves the nonlinear residual, which is exactly what we expect. Formulations which use the same domain discretization yield the same results, which is also what we expect. The solution times of all the models correlate with their formulation size (compare Table 3.1) formulations with fewer binary variables can be solved faster than the others, CCLOG and DLOG seem to perform best. This corresponds to the findings in [Vielma et al., 2010]. The number of discretization points heavily influences solution time and considerably slows down the solution process.

5.3.2. GasLib

The GasLib networks do not only contain pipes as network elements but include active elements also. With the system (5.14)-(5.20), we can calculate a solution on passive versions of each GasLib network. The GasLib networks have a more structure than the artificial instances and are generally bigger, so results on the GasLib instances are a little more interesting because the GasLib networks are a lot more realistic. We analyze the results of the tables D.6 to D.10.

In all of the networks improving bounds greatly speeds up the nonlinear solver. The nonlinear solution with improved bounds does not yield consistent results w.r.t the nonlinear residual. In two cases there is an improvement, in two other cases it gets worse, and for GasLib-134 the problem is marked as "infeasible". We are not able to provide an explanation for this behaviour. GasLib-134 can be solved by SCIP within 5.35 seconds which is almost 5 times faster than BARON. SCIP is not able to solve GasLib-40 and GasLib-135 within the time limit.

Iterative velocity approximation produces worse results than the nonlinear solver and is always slower than the nonlinear solver with improved bounds. The relative difference is always within 0.01% of the nonlinear result. But iterative velocity approximation can find acceptable approximations in less time than the nonlinear solver without needing improved bounds. The difference in the quality of solutions is acceptable, and the solution time is much faster.

If we look at the PWL models, we can see that CCLOG is always the fastest, in the three largest instances all other formulations trigger a timeout. Unfortunately increasing the number of discretization points is not feasible for even moderately sized instances like GasLib-24. In every instance, iterative velocity approximation performs better than all of the PWL models.

6. Conclusion and Further Ideas

We have introduced different ways to find approximations to the solution of the discretized nonlinear system for gas flow in pipelines (2.15)-(2.16) and compared them to current methods to solve MINLPs.

We employed different MIP-formulations of piecewise linear functions to approximate nonlinear function here, but with our approach, none of them give promising results. There are several problems that we see with our approach: First, the discretization for the pipe flow includes nonlinearities in two variables. Therefore piecewise linear formulations already increase in size compared to discretizations that only involve nonlinearities in one variable. Additionally, we do not yet have a good approach to improve the variable bounds, to reduce the number of discretization points. Solving the nonlinear problem first obviously defeats the purpose of removing the nonlinearities. We have also seen that for nonlinear problems without integer variables, the nonlinear solvers outperform piecewise linear formulations even when we use small discretizations.

Other researchers had better results by using a piecewise linear formulations as relaxations that enclose the nonlinear function. For further details see [Burlacu et al., 2017b].

Iterative velocity approximation can be used to approximate solutions for the pipe flow model on passive networks. We were only able to prove convergence on a single pipe, so there is no theoretical foundation for convergence on bigger networks. The convergence is also only guaranteed with very restrictive conditions. Maybe some methods to improve global convergence that are used for traditional iterative methods for systems of nonlinear equations could be applied (see [Kelley, 1995]). Commercial nonlinear solvers are faster than our implementation of iterative velocity approximation if tight variable bounds can be applied a priori. If these bounds are not available, iterative velocity approximation can give good results faster than the nonlinear solvers without tighter bounds. Rough approximations can thus be calculated with only a few iterations of Algorithm 1.

In each iteration of Algorithm 1 we only solve a linear system of equations. Therefore we suspect that the solution speed of iterative velocity approximation could be severely improved by implementing the iterative method with some fast sparse linear algebra package instead of going through a linear programming solver.

Ideally we could compare the results obtained by our solution methods to real-world measurements, but unfortunately, our data does not have pressure and flow measurements to which we could compare our results. So in the context of real gas networks, we can not estimate how well our approaches perform. With real-world data, we could also compare how well different discretizations of the momentum equation (2.12) perform. For example, you can discretize (2.12) in a way that only quadratic terms in single variables are part of the discretization. You can then approximate the quadratic terms with piecewise linear functions in only one variable (see above).

For future research, it would be interesting to prove further convergence theorems for iterative velocity approximation on classes of networks (e.g., on trees). Also, iterative velocity approximation needs to be incorporated into the optimization with active networks to asses performance in an optimization setting. Especially the choice of objective function might yield interesting results, e.g., how is the convergence of the method affected if the number of switching operations for active elements is to be minimized.

Appendices

A. Additional Data and Sketches

Product	Final Energy Consumption (1000 toe)
All products	216447.3
Natural gas	52415.9
Solid fuels	10163.2
Liquified petroleum gas (LPG)	1778.8
Gasoline (without bio components)	17230.3
Aviation gasoline	10.4
Other kerosene	6.1
Kerosene type jet fuel (without bio components)	9383.3
Gas/diesel oil (without bio components)	52232.2
Total fuel oil	855.3
Petroleum coke	116.9
Electrical energy	44486.4

Table A.1.: Final energy consumption in Germany in 2016. The amount of energy consumed is given in 1000 tonnes of oil equvialent (1000 toe).
Source: [Eurostat, 2018]

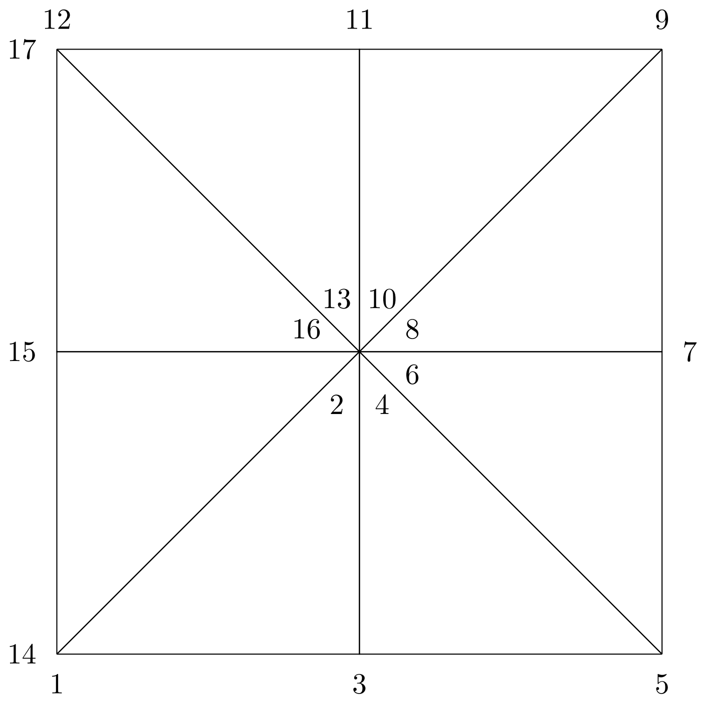


Figure A.1.: Vertex Traversal for J_1 Triangulation, from the lower left to the upper left corner.
The numbers 1 to 17 give the required vertex order to adhere to the ordering properties 3.5.1

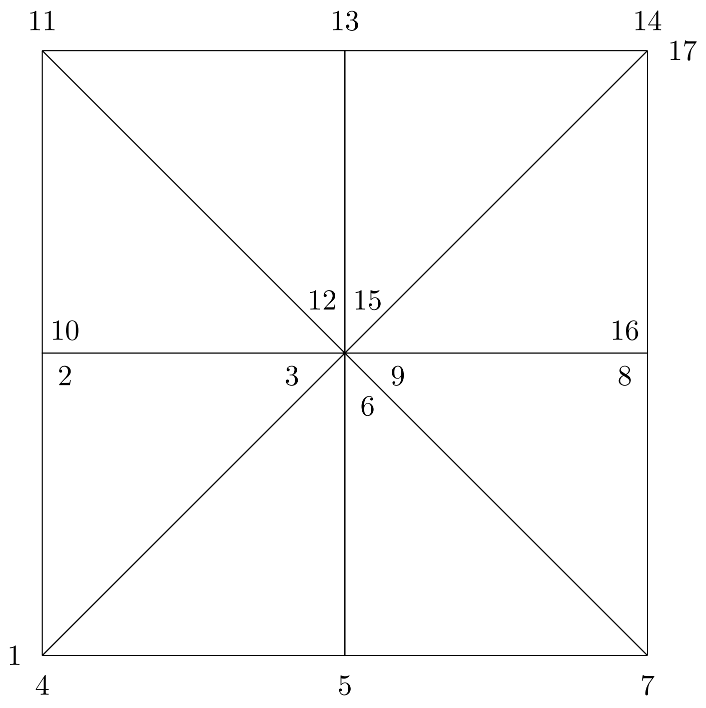


Figure A.2.: Vertex Traversal for J_1 Triangulation, from the lower left to the upper right corner.
The numbers 1 to 17 give the required vertex order to adhere to the ordering properties 3.5.1

B. Test Networks and Scenarios

A test network is a digraph $G = (V, A)$ as defined in 2.1. Each of the networks additionally comes with what we call a transient scenario. A transient scenario describes a sequence of flow demands over a specified time horizon. In our case each scenario has a total length of 5 hours (18000 seconds) and is divided into five time intervals of 1 hour each. Thus the set of time points is $\{t_i = i \cdot 3600 \mid i = 0, \ldots, 5\}$. We always construct a scenario from an initial state and a final state. The initial (final) state fixes flow demands q^{nom}_{u,t_0} (q^{nom}_{u,t_5}) for each of the nodes $u \in V$. For the intermediate demands at each timestep, we interpolate the start demands and the flow demands with an affine linear function, i.e.,

$$q^{nom}_{u,t_i} = \frac{i}{5} \cdot (q^{nom}_{u,t_5} - q^{nom}_{u,t_0}) + q^{nom}_{u,t_0}.$$

It is common to give the flow demands at the nodes as volumetric flow Q (in $1000 \cdot m^3/h$) (see for example [Schmidt et al., 2017]). Therefore we also give the demands as volumetric flow Q^{nom}, a conversion to mass flow (which is needed in the physical calculations) can be achieved via the following formula

$$q^{nom} = \rho_0 Q^{nom}, \tag{B.1}$$

where ρ_0 is normal density, i.e., density under normal conditions (1 bar pressure and 0° temperature). A short overview of the number of elements in each of the networks can be gathered from table B.1.

We created a set of artificial networks to test the algorithm on simple network topologies.

All of the GasLib networks, visualizations and the demand nomination files are available at `http://gaslib.zib.de`.

network	$\lvert V_+\rvert$	$\lvert V_-\rvert$	$\lvert V_0\rvert$	$\lvert A_{pi}\rvert$	$\lvert A_{rs}\rvert$	$\lvert A_{va}\rvert$	$\lvert A_{cv}\rvert$	$\lvert A_{sc}\rvert$	$\lvert A_{cs}\rvert$
`Path`	1	1	3	4	0	0	0	0	0
`Star`	3	3	1	6	0	0	0	0	0
`Cycle`	1	1	2	4	0	0	0	0	0
`Tree`	1	4	2	6	0	0	0	0	0
GasLib-11	3	3	5	8	0	1	0	0	2
GasLib-24	3	5	16	19	1	0	1	1	3
GasLib-40	3	29	8	39	0	0	0	0	6
GasLib-134	3	45	86	86	0	0	1	45	1
GasLib-135	6	99	30	141	0	0	0	0	29

Table B.1.: Number of Network Elements in the Test Networks

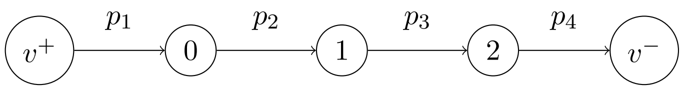


Figure B.1.: Schematic View of `Path`

Pipe	Length (km)	Diameter (cm)	Roughness (mm)
p_1	173.66	210	1
p_2	3.99	41.41	0.098
p_3	0.66	30	0.01
p_4	0.01	15	0.008

Table B.2.: Pipes in `Path`

B.1. Path

B.1.1. The Network

In a path network pipes are simply coupled in series. Thus the system for a path has simple coupling conditions (see subsection 4.3). In our particular instance we connect one entry node (v^+) with one exit node (v^-) via four pipes and thus we have three additional internal nodes. We want to test the behaviour of the iterative method 1 on a simple topology where we combine pipes with extremely different parameters. The pipes are combinations of values from the GasLib, see table 4.1. A schematic of the `Path` instance can be seen in figure B.1. The detailed parameters for the pipes are given in table B.2.

B.1.2. The Scenario

We are testing an unbalanced flow scenario. We start with balanced demands

$$Q^{nom}_{v^+,t_0} = 300 \cdot 1000 \cdot \frac{m^3}{h},$$
$$Q^{nom}_{v^-,t_0} = -300 \cdot 1000 \cdot \frac{m^3}{h}.$$

and reduce the demands to

$$Q^{nom}_{v^+,t_5} = 270 \cdot 1000 \cdot \frac{m^3}{h},$$
$$Q^{nom}_{v^-,t_5} = 260 \cdot 1000 \cdot \frac{m^3}{h}.$$

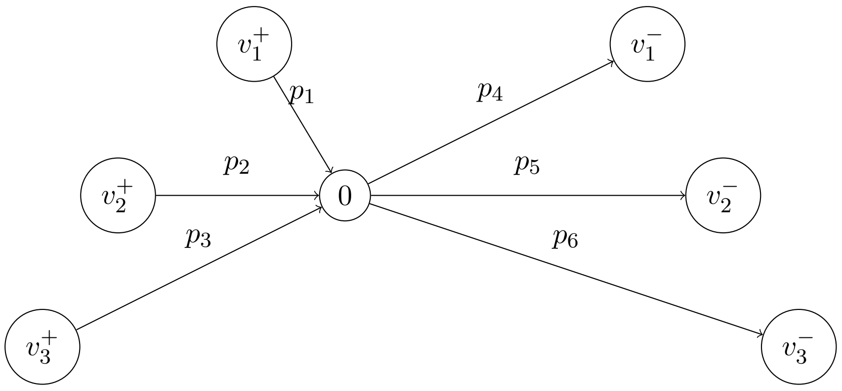


Figure B.2.: Schematic View of `Star`

Pipe	Length (km)	Diameter (cm)	Roughness (mm)
p_1	1	41.41	0.1
p_2	2	41.41	0.1
p_3	3	41.41	0.1
p_4	4	41.41	0.1
p_5	5	41.41	0.1
p_6	6	41.41	0.1

Table B.3.: Pipes in `Star`

B.2. Star

B.2.1. The Network

A star-shaped network yields the simplest coupling conditions. Only a single coupling constraint is added at the center of the star. We want to know how such a simple constraint influences the solution process of algorithm 1. We created a star-shaped network with seven nodes and six pipes. In the network there are three entry nodes (v_1^+, v_2^+, v_3^+) and three exit nodes (v_1^-, v_2^-, v_3^-). The pipes were chosen to have the median roughness of the GasLib, and the average diameter of the GasLib. Different pipe lengths have been chosen so in the stationary calculation, the pressure at each entry (exit) is different from the pressure at the other entries (exits). The pipe lengths are chosen to be greater than the median and to be of the same magnitude as the average length in the GasLib. The exact values can be gathered from table B.3. A visualization is given in figure B.2.

B.2.2. The Scenario

We start with equal demands at each boundary node

$$Q^{nom}_{v_i^+,t_0} = 300 \cdot 1000 \cdot \frac{m^3}{h} \qquad i \in \{1,2,3\}\,,$$
$$Q^{nom}_{v_i^-,t_0} = -300 \cdot 1000 \cdot \frac{m^3}{h} \qquad i \in \{1,2,3\}\,.$$

The final state is balanced with a 10% reduction of the initial demands:

$$Q^{nom}_{v_i^+,t_0} = 270 \cdot 1000 \cdot \frac{m^3}{h} \qquad i \in \{1,2,3\}\,,$$
$$Q^{nom}_{v_i^-,t_0} = -270 \cdot 1000 \cdot \frac{m^3}{h} \qquad i \in \{1,2,3\}\,.$$

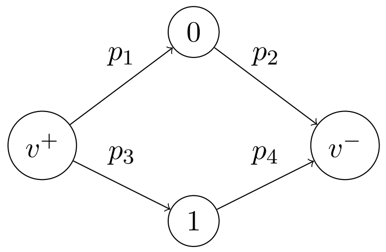


Figure B.3.: Schematic View of `Cycle`

Pipe	Length (km)	Diameter (cm)	Roughness (mm)
p_1	4	30	0.1
p_2	4	30	0.1
p_3	3	30	0.1
p_4	3	30	0.1

Table B.4.: Pipes in `Cycle`

B.3. Cycle

B.3.1. The Network

By creating a cycle with 4 nodes we want to see what happens to the iterative method 1 if the coupling conditions between pipes introduce cyclic dependencies between variables. The cycle has one entry (v^+) and one exit node (v^-) connected through four pipes. For the pipes we chose to use the median diameter and the median roughness from the GasLib (see 4.1), the pipe length is of the same magnitude as the average length of pipes in the GasLib. The pipes differ in length so the two paths from the source node to the sink node are different. The exact pipe configurations are listed in table B.4.

B.3.2. The Scenario

We test a balanced scenario. We start with

$$\begin{aligned} Q^{nom}_{v^+,t_0} &= 300 \cdot 1000 \cdot \frac{m^3}{h}, \\ Q^{nom}_{v^-,t_0} &= -300 \cdot 1000 \cdot \frac{m^3}{h}, \end{aligned}$$

and reduce the demands by 10% to

$$\begin{aligned} Q^{nom}_{v^+,t_5} &= 270 \cdot 1000 \cdot \frac{m^3}{h}, \\ Q^{nom}_{v^-,t_5} &= 270 \cdot 1000 \cdot \frac{m^3}{h}. \end{aligned}$$

Pipe	Length (km)	Diameter (cm)	Roughness (mm)
p_1	1	40	0.1
p_2	2	40	0.1
p_3	3	40	0.1
p_4	4	40	0.1
p_5	5	40	0.1
p_6	6	40	0.1

Table B.5.: Pipes in `Tree`

B.4. Tree

B.4.1. The Network

The `Tree` network tests behaviour on a small tree-shaped network. The network contains one entry node v^+ and four exit nodes v_1^-, v_1^-, v_1^- and v_4^-. The parameters for the six pipes in the `Tree` network are given in table B.5.

B.4.2. The Scenario

For the tree we chose a more complex scenario, again the initial state is balanced with demands

$$\begin{aligned}
Q^{nom}_{v^+,t_0} &= 300 \cdot 1000 \cdot \frac{m^3}{h}, \\
Q^{nom}_{v_i^-,t_0} &= -75 \cdot 1000 \cdot \frac{m^3}{h}, \quad i \in \{1,2,3,4\},
\end{aligned}$$

and the final state

$$\begin{aligned}
Q^{nom}_{v^+,t_0} &= 270 \cdot 1000 \cdot \frac{m^3}{h}, \\
Q^{nom}_{v_1^-,t_0} &= -30 \cdot 1000 \cdot \frac{m^3}{h}, \\
Q^{nom}_{v_2^-,t_0} &= -100 \cdot 1000 \cdot \frac{m^3}{h}, \\
Q^{nom}_{v_3^-,t_0} &= -10 \cdot 1000 \cdot \frac{m^3}{h}, \\
Q^{nom}_{v_4^-,t_0} &= -100 \cdot 1000 \cdot \frac{m^3}{h}.
\end{aligned}$$

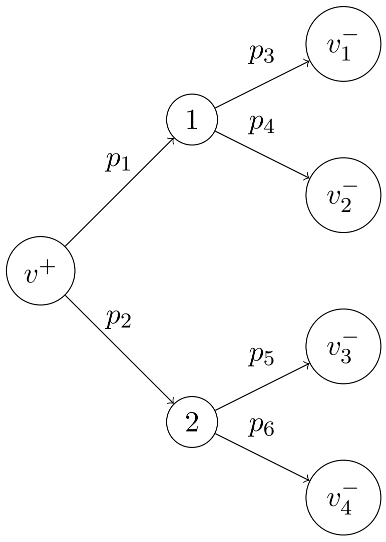


Figure B.4.: Schematic View of `Tree`

B.5. GasLib-11

B.5.1. The Network

GasLib-11 is a test network from [Schmidt et al., 2017].

B.5.2. The Scenario

The initial state is taken from the test scenario `GasLib-11.scn` from [Schmidt et al., 2017]. The final state is a 10% reduction of all the demands given in the initial state.

B.6. Gaslib-24

B.6.1. The Network

GasLib-24 is a test network from [Schmidt et al., 2017].

B.6.2. The Scenario

The initial state is taken from the test scenario `GasLib-24.scn` from [Schmidt et al., 2017]. The final state is a 10% reduction of all the demands given in the initial state.

B.7. Gaslib-40

B.7.1. The Network

GasLib-40 is a test network from [Schmidt et al., 2017].

B.7.2. The Scenario

The initial state is taken from the test scenario `GasLib-40.scn` from [Schmidt et al., 2017]. The final state is a reduction of the demands from the initial state to 20% of their original value.

B.8. GasLib-134

B.8.1. The Network

GasLib-134 is a network from [Schmidt et al., 2017] and is based on real network data.

B.8.2. The Scenario

The initial state is taken from the scenario `2012-08-14.scn` and the final state is taken from the scenario `2012-08-15.scn`. Both scenario files are part of the nominations for GasLib-134.

B.9. Gaslib-135

B.9.1. The Network

GasLib-135 is a test network from [Schmidt et al., 2017].

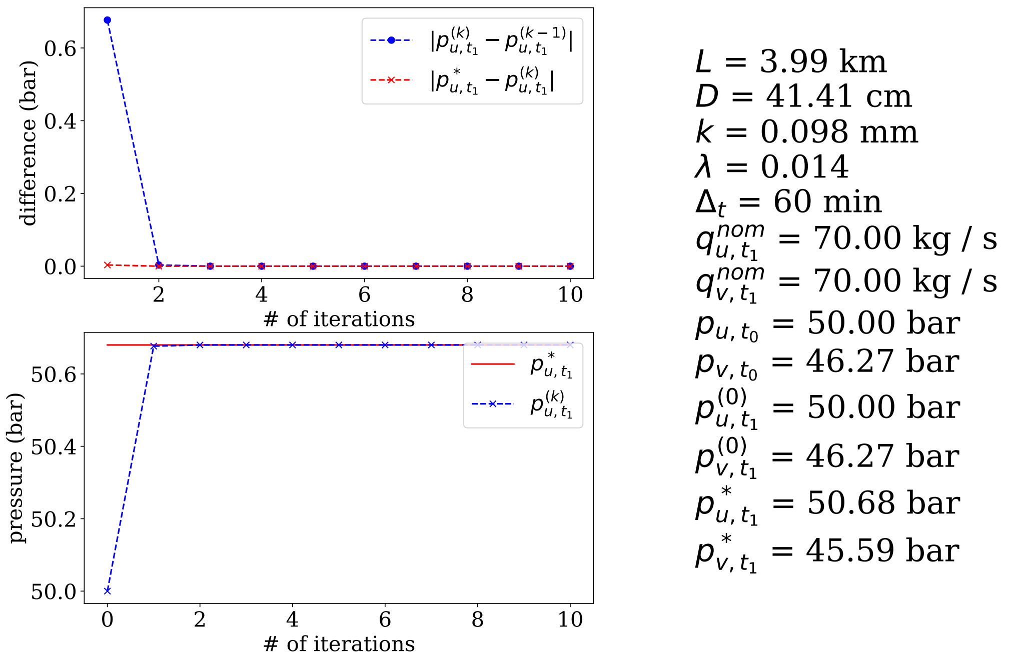
B.9.2. The Scenario

The initial state is taken from the test scenario `GasLib-135` from [Schmidt et al., 2017]. The final state is a 10% reduction of all the demands given in the initial state.

C. Convergence Results

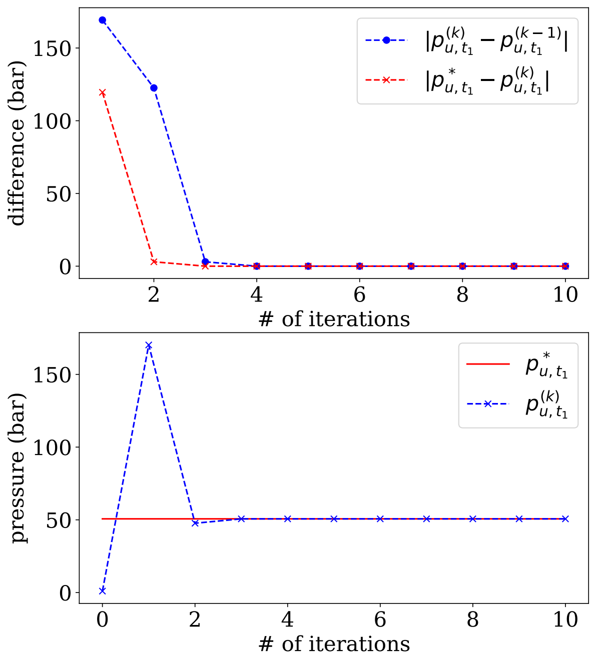
Length (km)	Diameter (cm)	Roughness (mm)	l
0.01	15	0.008	76.26
0.01	15	0.1	127.48
0.01	15	1	237.27
0.01	30	0.008	2.11
0.01	30	0.1	3.41
0.01	30	1	6.02
0.01	41.41	0.008	0.40
0.01	41.41	0.1	0.64
0.01	41.41	1	1.10
0.01	210	0.008	0.00
0.01	210	0.1	0.00
0.01	210	1	0.00
0.66	30	0.008	139.41
0.66	30	0.1	225.25
0.66	30	1	397.42
0.66	41.41	0.008	26.37
0.66	41.41	0.1	42.00
0.66	41.41	1	72.49
0.66	210	0.008	0.01
0.66	210	0.1	0.01
0.66	210	1	0.01
3.99	30	0.008	842.82
3.99	30	0.1	1361.75
3.99	30	1	2402.59
3.99	41.41	0.008	159.40
3.99	41.41	0.1	253.88
3.99	41.41	1	438.26
3.99	210	0.008	0.04
3.99	210	0.1	0.06
3.99	210	1	0.09
173.66	210	0.008	1.61
173.66	210	0.1	2.41
173.66	210	1	3.81

Table C.1.: Estimate on the lipschitz parameter l rounded to 2 decimal places for different pipes with Theorem 4.2.6.



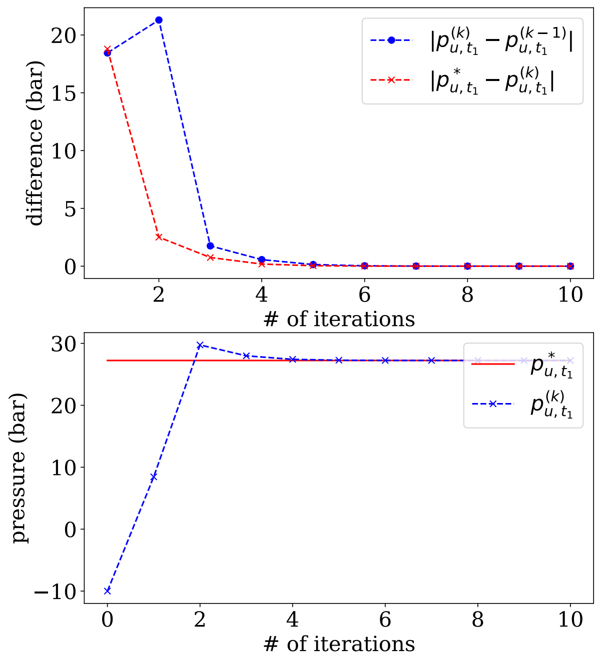
$L = 3.99$ km
$D = 41.41$ cm
$k = 0.098$ mm
$\lambda = 0.014$
$\Delta_t = 60$ min
$q^{nom}_{u,t_1} = 70.00$ kg / s
$q^{nom}_{v,t_1} = 70.00$ kg / s
$p_{u,t_0} = 50.00$ bar
$p_{v,t_0} = 46.27$ bar
$p^{(0)}_{u,t_1} = 50.00$ bar
$p^{(0)}_{v,t_1} = 46.27$ bar
$p^{*}_{u,t_1} = 50.68$ bar
$p^{*}_{v,t_1} = 45.59$ bar

Figure C.1.: Iterative velocity approximation on a pipe (u, v) of length L, diameter D, and integral roughness k. The figure includes the difference between start and end time Δ_t, the demand values q^{nom}_{u,t_1} at u and q^{nom}_{v,t_1} at v, the pressure at the first time point (i.e., the initial state) p_{u,t_0} at u and p_{v,t_0} at v. We start the iteration with the pressure values $p^{(0)}_{u,t_1}$ for u and $p^{(0)}_{v,t_1}$ for v. p^{*}_{u,t_1} and p^{*}_{v,t_1} are pressure values in an exact solution at u and v respectively (rounded to two decimal places). The calculations are performed with the physical parameters from Table 5.1. We show the absolute difference between subsequent solutions $\left|p^{(k)}_{u,t_1} - p^{(k-1)}_{u,t_1}\right|$ and the progression of the pressure values $p^{(k)}_{u,t_1}$.
Here we treat the "average" pipe (refer to Subsection 5.2.1) and start with values close to the solution.



$L = 3.99$ km
$D = 41.41$ cm
$k = 0.098$ mm
$\lambda = 0.014$
$\Delta_t = 60$ min
$q_{u,t_1}^{nom} = 70.00$ kg / s
$q_{v,t_1}^{nom} = 70.00$ kg / s
$p_{u,t_0} = 50.00$ bar
$p_{v,t_0} = 46.27$ bar
$p_{u,t_1}^{(0)} = 1.00$ bar
$p_{v,t_1}^{(0)} = 1.00$ bar
$p_{u,t_1}^{*} = 50.68$ bar
$p_{v,t_1}^{*} = 45.59$ bar

Figure C.2.: Iterative velocity approximation on a pipe (u, v) of length L, diameter D, and integral roughness k. The figure includes the difference between start and end time Δ_t, the demand values q_{u,t_1}^{nom} at u and q_{v,t_1}^{nom} at v, the pressure at the first time point (i.e., the initial state) p_{u,t_0} at u and p_{v,t_0} at v. We start the iteration with the pressure values $p_{u,t_1}^{(0)}$ for u and $p_{v,t_1}^{(0)}$ for v. p_{u,t_1}^{*} and p_{v,t_1}^{*} are pressure values in an exact solution at u and v respectively (rounded to two decimal places). The calculations are performed with the physical parameters from Table 5.1. We show the absolute difference between subsequent solutions $\left|p_{u,t_1}^{(k)} - p_{u,t_1}^{(k-1)}\right|$ and the progression of the pressure values $p_{u,t_1}^{(k)}$.
Here we see iterations on the "average" pipe (refer to 5.2.1) and start with *small* pressure values.



L = 3.99 km
D = 41.41 cm
k = 0.098 mm
λ = 0.014
Δ_t = 60 min
q^{nom}_{u,t_1} = 70.00 kg / s
q^{nom}_{v,t_1} = 70.00 kg / s
p_{u,t_0} = 25.00 bar
p_{v,t_0} = 16.31 bar
$p^{(0)}_{u,t_1}$ = -10.00 bar
$p^{(0)}_{v,t_1}$ = -10.00 bar
p^{*}_{u,t_1} = 27.23 bar
p^{*}_{v,t_1} = 14.08 bar

Figure C.3.: Iterative velocity approximation on a pipe (u, v) of length L, diameter D, and integral roughness k. The figure includes the difference between start and end time Δ_t, the demand values q^{nom}_{u,t_1} at u and q^{nom}_{v,t_1} at v, the pressure at the first time point (i.e., the initial state) p_{u,t_0} at u and p_{v,t_0} at v. We start the iteration with the pressure values $p^{(0)}_{u,t_1}$ for u and $p^{(0)}_{v,t_1}$ for v. p^{*}_{u,t_1} and p^{*}_{v,t_1} are pressure values in an exact solution at u and v respectively (rounded to two decimal places). The calculations are performed with the physical parameters from Table 5.1. We show the absolute difference between subsequent solutions $\left|p^{(k)}_{u,t_1} - p^{(k-1)}_{u,t_1}\right|$ and the progression of the pressure values $p^{(k)}_{u,t_1}$.
Here we see iterations on the "average" pipe (refer to 5.2.1) and start with *negative* pressure values.

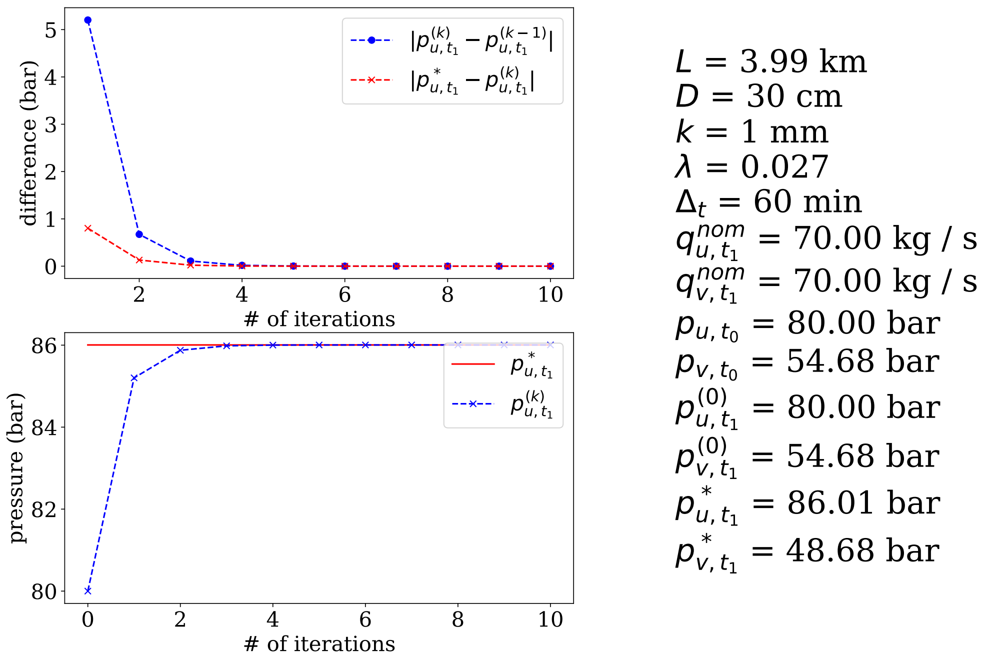


Figure C.4.: Iterative velocity approximation on a pipe (u, v) of length L, diameter D, and integral roughness k. The figure includes the difference between start and end time Δ_t, the demand values q^{nom}_{u,t_1} at u and q^{nom}_{v,t_1} at v, the pressure at the first time point (i.e., the initial state) p_{u,t_0} at u and p_{v,t_0} at v. We start the iteration with the pressure values $p^{(0)}_{u,t_1}$ for u and $p^{(0)}_{v,t_1}$ for v. p^{*}_{u,t_1} and p^{*}_{v,t_1} are pressure values in an exact solution at u and v respectively (rounded to two decimal places). The calculations are performed with the physical parameters from Table 5.1. We show the absolute difference between subsequent solutions $\left|p^{(k)}_{u,t_1} - p^{(k-1)}_{u,t_1}\right|$ and the progression of the pressure values $p^{(k)}_{u,t_1}$.
Here we see iterations on the "hard" pipe (refer to 5.2.1) and start with pressure values close to the solution.

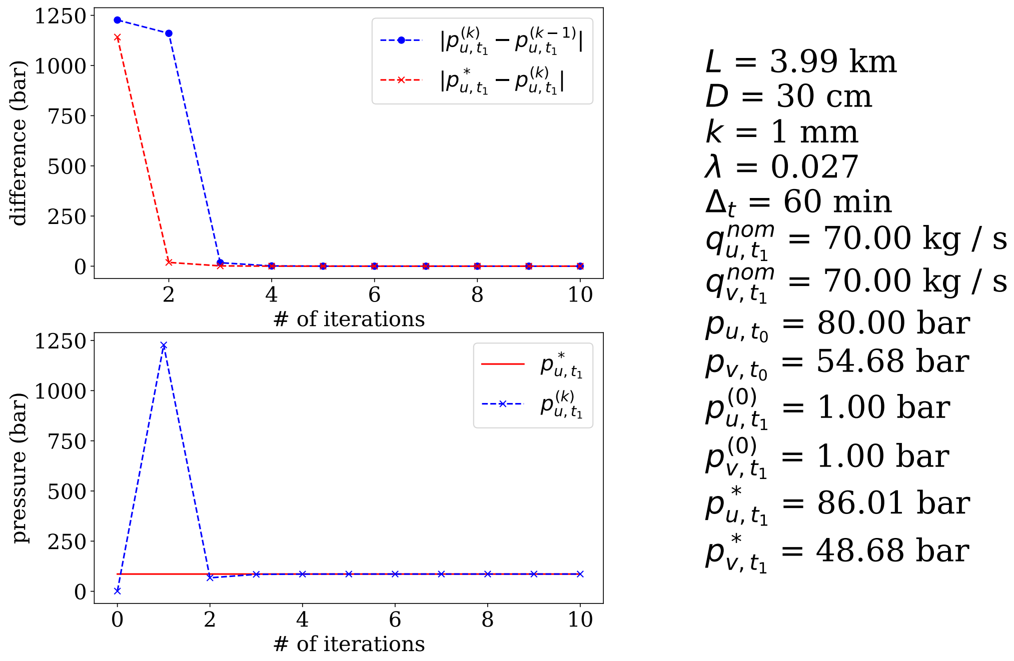
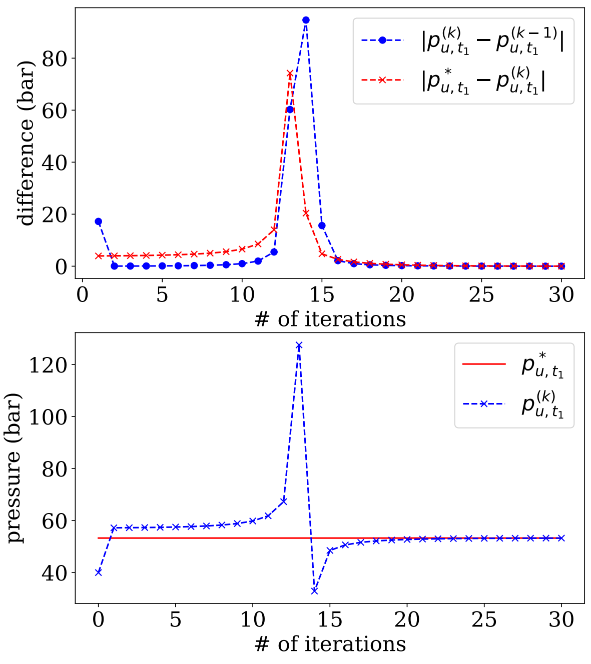


Figure C.5.: Iterative velocity approximation on a pipe (u, v) of length L, diameter D, and integral roughness k. The figure includes the difference between start and end time Δ_t, the demand values q_{u,t_1}^{nom} at u and q_{v,t_1}^{nom} at v, the pressure at the first time point (i.e., the initial state) p_{u,t_0} at u and p_{v,t_0} at v. We start the iteration with the pressure values $p_{u,t_1}^{(0)}$ for u and $p_{v,t_1}^{(0)}$ for v. p_{u,t_1}^{*} and p_{v,t_1}^{*} are pressure values in an exact solution at u and v respectively (rounded to two decimal places). The calculations are performed with the physical parameters from Table 5.1. We show the absolute difference between subsequent solutions $\left|p_{u,t_1}^{(k)} - p_{u,t_1}^{(k-1)}\right|$ and the progression of the pressure values $p_{u,t_1}^{(k)}$.
Here we see iterations on the "hard" pipe (refer to 5.2.1) and start with small pressure values.



L = 14.8 km
D = 39 cm
k = 0.1 mm
λ = 0.014
Δ_t = 60 min
q_{u,t_1}^{nom} = 62.00 kg / s
q_{v,t_1}^{nom} = 60.00 kg / s
p_{u,t_0} = 45.00 bar
p_{v,t_0} = 14.09 bar
$p_{u,t_1}^{(0)}$ = 40.00 bar
$p_{v,t_1}^{(0)}$ = 14.09 bar
p_{u,t_1}^{*} = 53.28 bar
p_{v,t_1}^{*} = 16.57 bar

Figure C.6.: Iterative velocity approximation on a pipe (u, v) of length L, diameter D, and integral roughness k. The figure includes the difference between start and end time Δ_t, the demand values q_{u,t_1}^{nom} at u and q_{v,t_1}^{nom} at v, the pressure at the first time point (i.e., the initial state) p_{u,t_0} at u and p_{v,t_0} at v. We start the iteration with the pressure values $p_{u,t_1}^{(0)}$ for u and $p_{v,t_1}^{(0)}$ for v. p_{u,t_1}^{*} and p_{v,t_1}^{*} are pressure values in an exact solution at u and v respectively (rounded to two decimal places). The calculations are performed with the physical parameters from Table 5.1. We show the absolute difference between subsequent solutions $\left|p_{u,t_1}^{(k)} - p_{u,t_1}^{(k-1)}\right|$ and the progression of the pressure values $p_{u,t_1}^{(k)}$.
Here we see iterations on the pipe from Example 4.2.3, which has multiple physically feasible solutions for the pipe flow system.

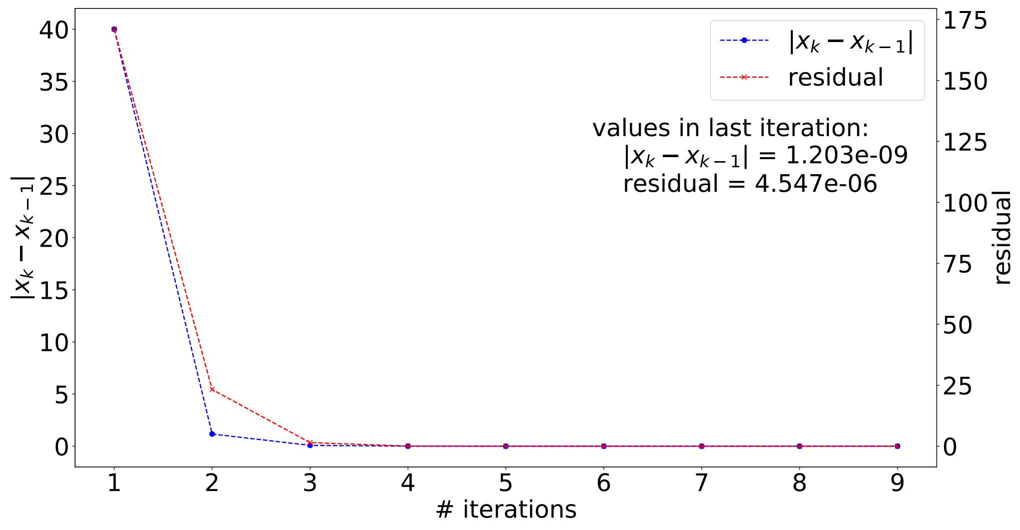


Figure C.7.: Iterative velocity approximation on `Path`.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.

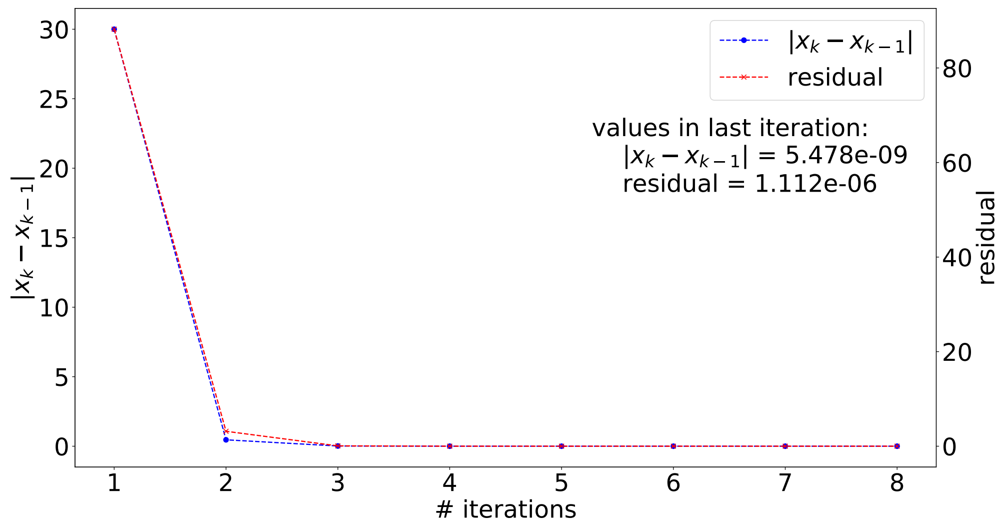


Figure C.8.: Iterative velocity approximation on `Star`.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.

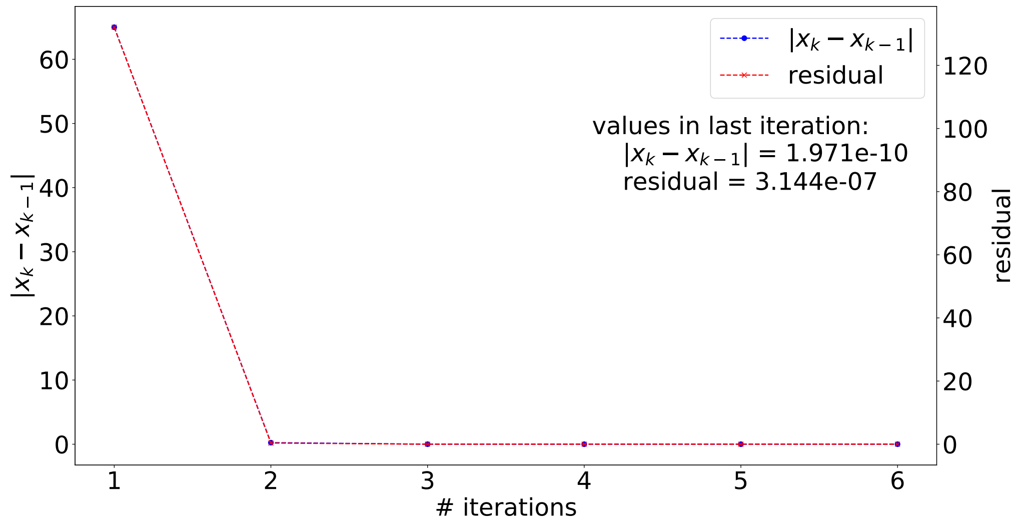


Figure C.9.: Iterative velocity approximation on `Tree`.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_{\infty}$.
The physical parameters used for the calculations are specified in Table 5.1.

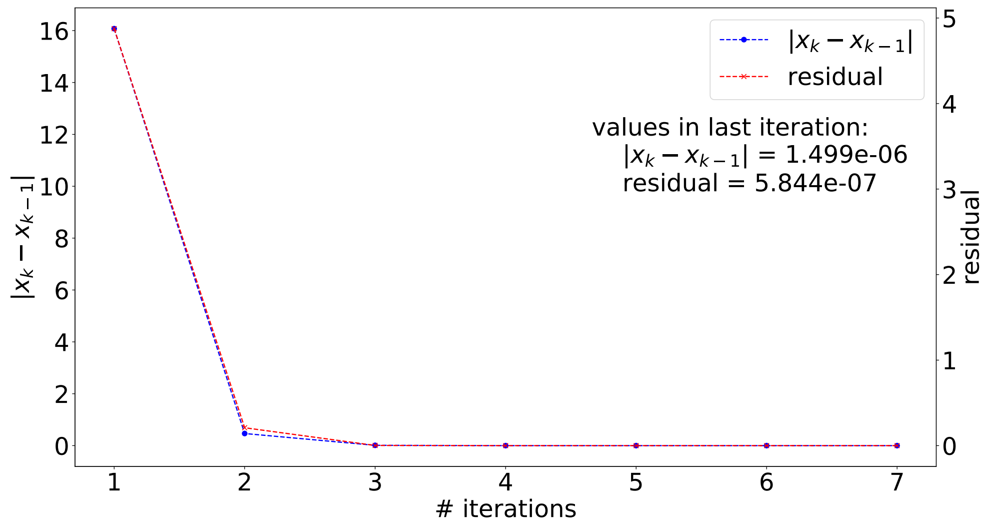


Figure C.10.: Iterative velocity approximation on `Cycle`.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.

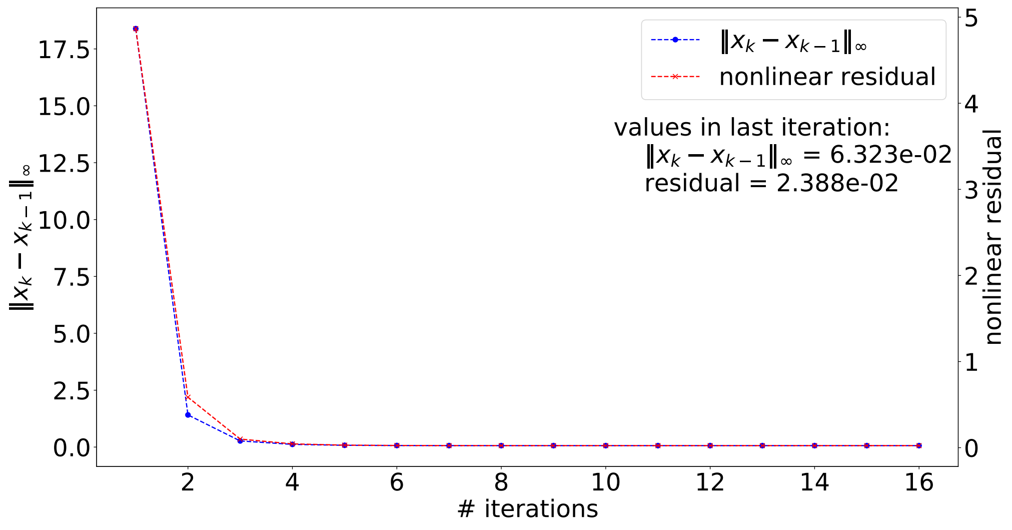


Figure C.11.: Iterative velocity approximation on GasLib-11.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.

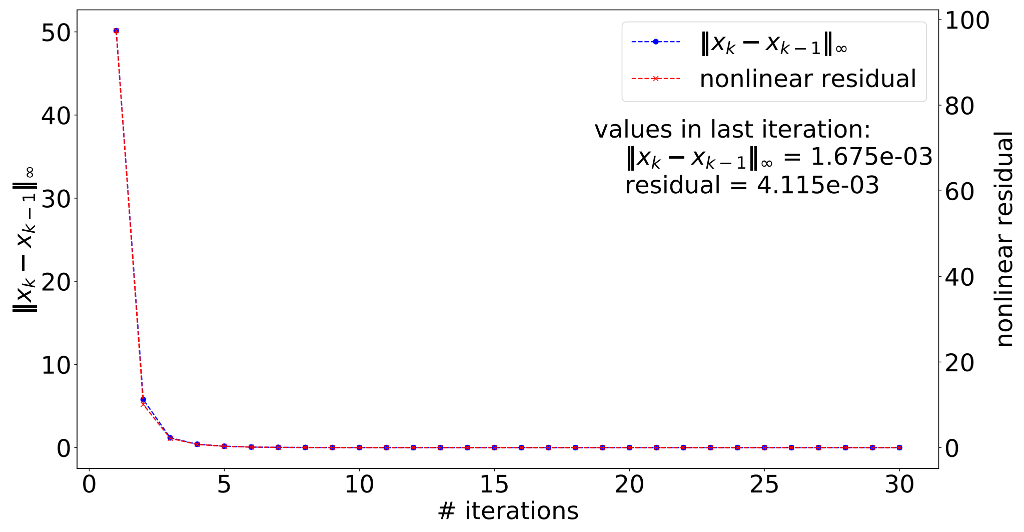
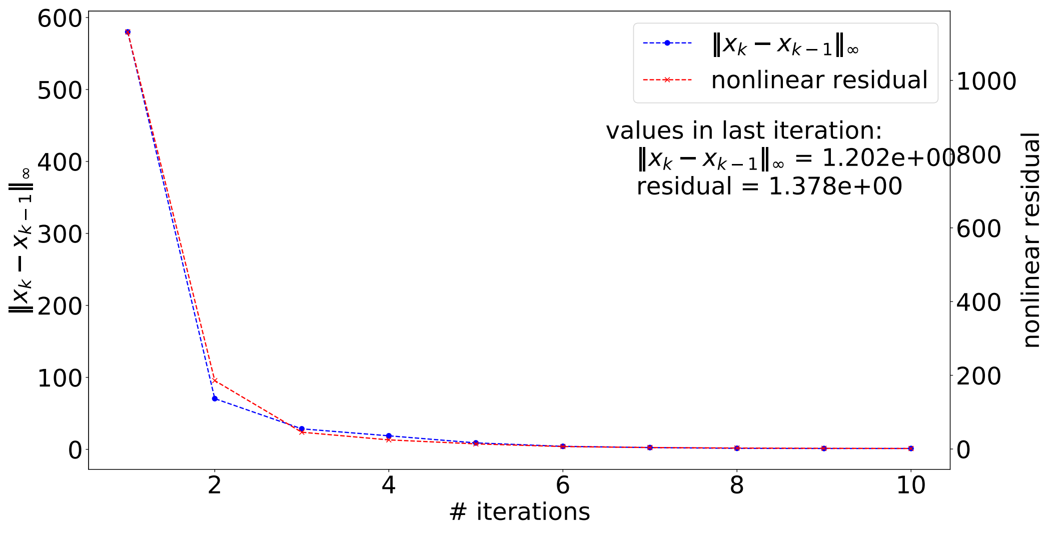
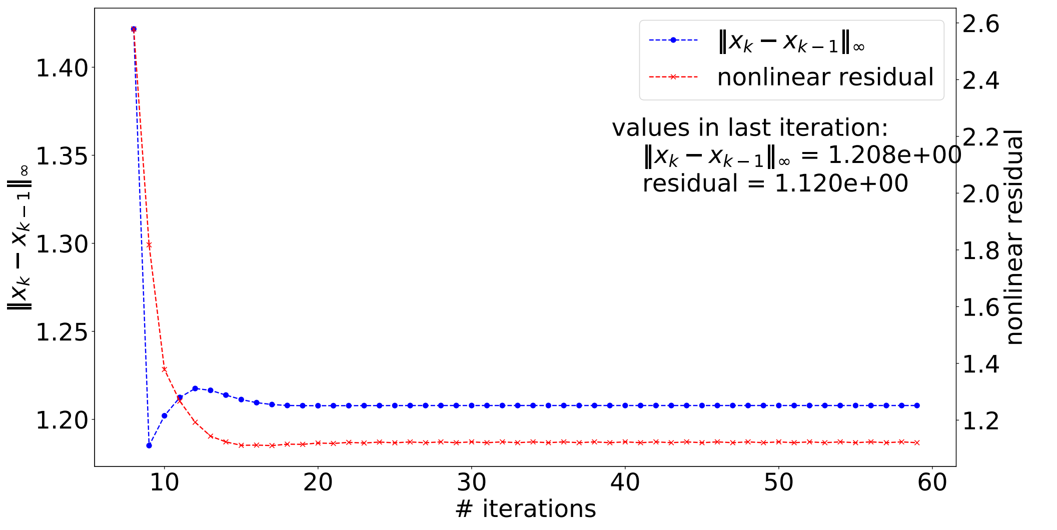


Figure C.12.: Iterative velocity approximation on GasLib-24.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.

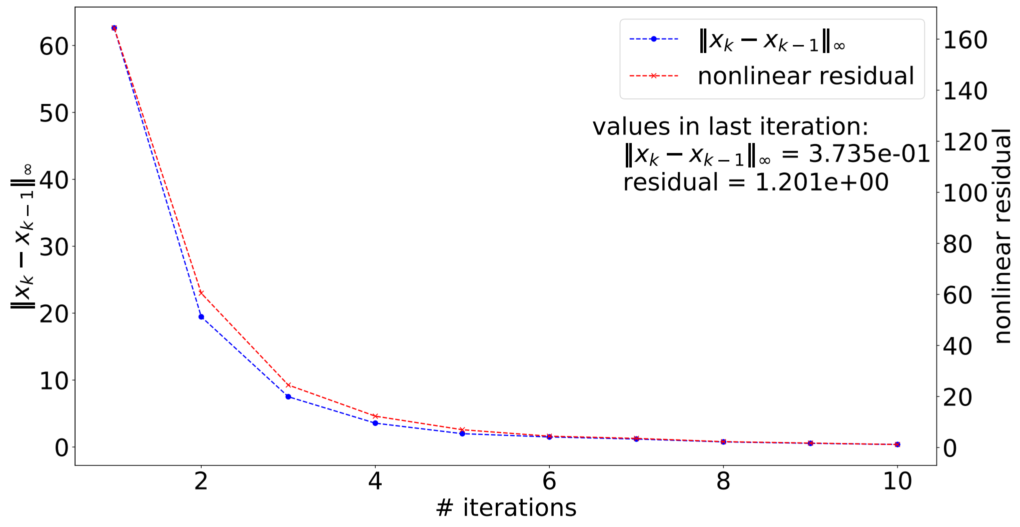


(a) First 10 iterations

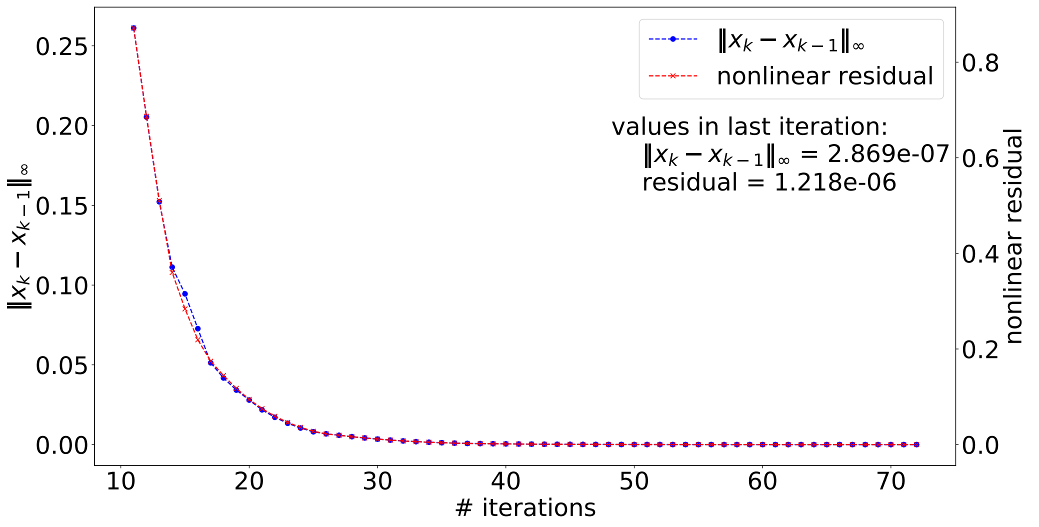


(b) Iterations 7 to 59

Figure C.13.: Iterative velocity approximation on GasLib-40.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.

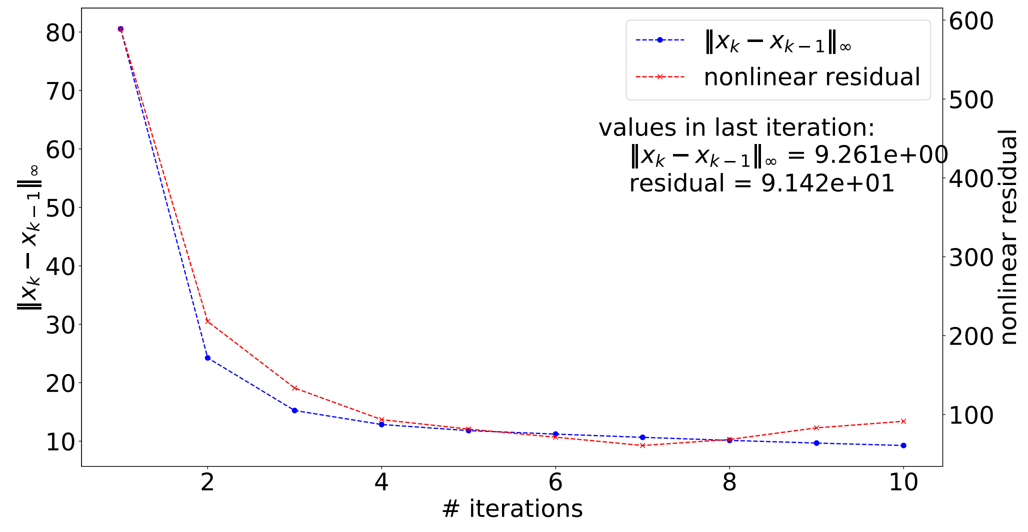


(a) First 10 iterations

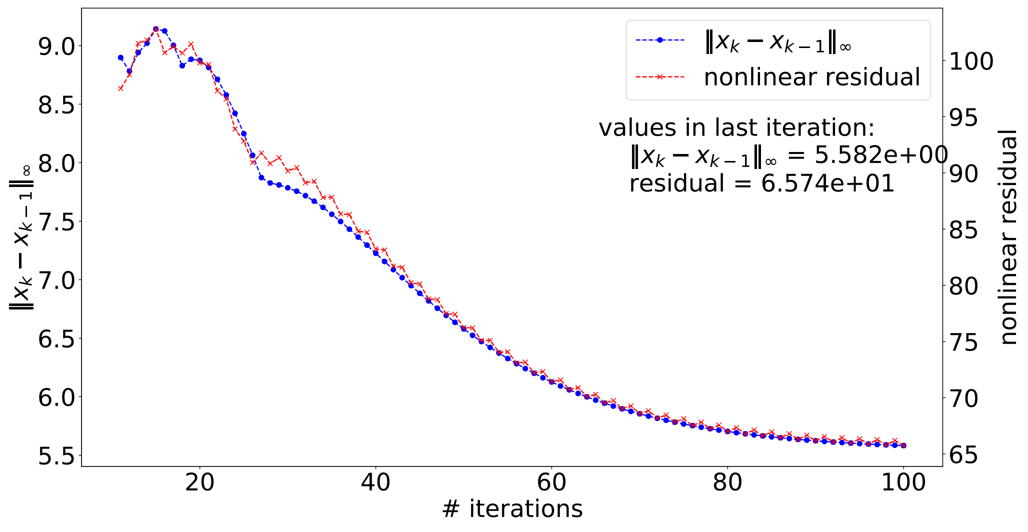


(b) Iterations 10 to 72

Figure C.14.: Iterative velocity approximation on GasLib-134.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.

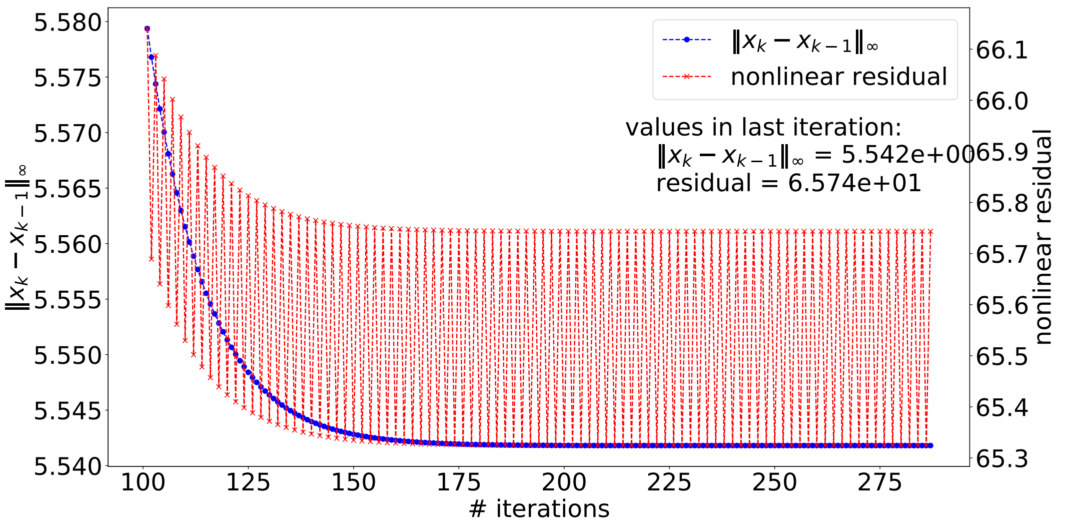


(a) First 10 iterations



(b) Iterations 10 to 100

Figure C.15.: Iterative velocity approximation on GasLib-135.
The iteration is performed until the iteration starts to cycle.
We display the maximum nonlinear residual $r_{\max}$ for each iteration, as well as the maximum difference between subsequent iterations $\left\|x^{(k)} - x^{(k-1)}\right\|_\infty$.
The physical parameters used for the calculations are specified in Table 5.1.



(c) Iterations 100 to 287

Figure C.15.: Iterative Velocity Approximation on GasLib-135 cont.

D. Computational Data

Abbreviation	Explanation
NL-0	nonlinear calculation using BARON
	without tighter variable bounds
NL-BARON	nonlinear calculation using BARON
	with tighter variable bounds applied
NL-SCIP	nonlinear calculation using SCIP
	with tighter variable bounds applied
CC-k	CC formulation (see section 3.3)
	with tighter variable bounds applied
	k discretization points
CCLOG-k	CCLOG formulation (see subsection 3.3.1)
	with tighter variable bounds applied
	k discretization points
DCC-k	DCC formulation (see section 3.2)
	with tighter variable bounds applied
	k discretization points
DLOG-k	DLOG formulation (see subsection 3.2.1)
	with tighter variable bounds applied
	k discretization points
INC-k	INC formulation (see section 3.5)
	with tighter variable bounds applied
	k discretization points
MC-k	MC formulation (see section 3.4)
	with tighter variable bounds applied
	k discretization points
ITERATE	iterative velocity approximation (see chapter 4)
	without tighter variable bounds applied

Table D.1.: Abbreviations of Solution Methods

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	2.81e-10	0.00e+00	0.78
NL-BARON	3.34e-10	4.36e-14	0.49
NL-SCIP	3.30e-09	2.69e-13	0.05
ITERATE	4.55e-06	8.09e-11	2.37
CC-3	8.87e+00	1.34e-03	5.35
CCLOG-3	8.37e+00	1.31e-03	1.12
CCLOG-5	2.23e+00	3.57e-04	2.52
CCLOG-7	9.41e-01	1.51e-04	6.74
DCC-3	8.87e+00	1.34e-03	7.02
DLOG-3	8.87e+00	1.34e-03	1.05
INC-3	8.37e+00	1.31e-03	4.71
MC-3	8.87e+00	1.34e-03	1.23

Table D.2.: Comparison of Approximation Methods for Transient Flow Calculations on the `Path` Network

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	4.55e-11	0.00e+00	0.57
NL-BARON	3.81e-11	7.80e-15	0.62
NL-SCIP	9.57e-09	9.76e-01	0.72
ITERATE	1.11e-06	9.76e-01	2.75
CC-3	5.02e+01	7.04e-03	5.64
CCLOG-3	7.18e+01	1.00e-02	2.70
CCLOG-5	1.46e+01	2.05e-03	1.66
CCLOG-7	6.01e+00	8.49e-04	18.87
DCC-3	5.02e+01	7.04e-03	7.54
DLOG-3	5.02e+01	7.04e-03	0.68
INC-3	7.18e+01	1.00e-02	1.96
MC-3	5.02e+01	7.04e-03	2.20

Table D.3.: Comparison of Approximation Methods for Transient Flow Calculations on the `Star` Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	2.80e-06	0.00e+00	0.51
NL-BARON	2.88e-06	8.05e-11	0.41
NL-SCIP	2.18e-11	7.24e-11	0.06
ITERATE	3.14e-07	9.45e-11	3.36
CC-3	2.27e+00	3.42e-04	6.37
CCLOG-3	2.28e+00	3.43e-04	1.27
CCLOG-5	4.27e-01	5.51e-05	5.47
CCLOG-7	2.38e-01	3.56e-05	56.50
DCC-3	2.27e+00	3.42e-04	11.34
DLOG-3	2.27e+00	3.42e-04	5.57
INC-3	2.28e+00	3.43e-04	1.83
MC-3	2.27e+00	3.42e-04	9.33

Table D.4.: Comparison of Approximation Methods for Transient Flow Calculations on the `Tree` Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	8.14e-11	0.00e+00	0.49
NL-BARON	7.36e-07	2.74e-10	0.40
NL-SCIP	4.79e-11	7.14e-15	0.05
ITERATE	5.03e-06	2.24e-10	1.97
CC-3	2.56e-01	5.51e-04	8.91
CCLOG-3	2.52e-01	5.45e-04	2.23
CCLOG-5	6.71e-02	1.46e-04	10.41
CCLOG-7	3.02e-02	6.62e-05	17.96
DCC-3	2.56e-01	5.51e-04	8.77
DLOG-3	2.56e-01	5.51e-04	8.30
INC-3	2.52e-01	5.45e-04	5.95
MC-3	2.56e-01	5.51e-04	15.52

Table D.5.: Comparison of Approximation Methods for Transient Flow Calculations on the `Cycle` Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	2.07e-07	0.00e+00	0.62
NL-BARON	6.44e-06	4.91e-09	0.29
NL-SCIP	4.84e-08	1.74e-10	0.19
ITERATE	2.39e-01	4.23e-08	2.51
CC-3	2.66e-01	5.85e-04	9.89
CCLOG-3	2.71e-01	5.92e-04	0.95
CCLOG-5	7.12e-02	1.62e-04	17.36
CCLOG-7	3.06e-02	6.84e-05	42.66
DCC-3	2.66e-01	5.85e-04	14.83
DLOG-3	2.66e-01	5.85e-04	6.43
INC-3	2.71e-01	5.92e-04	6.37
MC-3	2.66e-01	5.85e-04	234.02

Table D.6.: Comparison of Approximation Methods for Transient Flow Calculations on the GasLib-11 Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	4.02e-09	0.00e+00	1.00
NL-BARON	1.17e-09	3.13e-13	0.46
NL-SCIP	1.48e-09	6.78e-14	1.43
ITERATE	1.07e-02	5.93e-06	6.02
CC-3	timeout	-	3600
CCLOG-3	5.22e+00	1.60e-04	7.26
CCLOG-5	timeout	-	3600
CCLOG-7	timeout	-	3600
DCC-3	timeout	-	3600
DLOG-3	timeout	-	3600
INC-3	5.22e+00	1.60e-04	1159.25
MC-3	timeout	-	3600

Table D.7.: Comparison of Approximation Methods for Transient Flow Calculations on the GasLib-24 Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	1.88e-09	0.00e+00	6.65
NL-BARON	1.16e-06	4.04e-11	0.87
NL-SCIP	timeout	-	3600
ITERATE	2.60e+00	3.30e-05	12.89
CC-3	timeout	-	3600
CCLOG-3	2.58e+02	6.29e-02	452.96
CCLOG-5	timeout	-	3600
CCLOG-7	timeout	-	3600
DCC-3	timeout	-	3600
DLOG-3	timeout	-	3600
INC-3	timeout	-	3600
MC-3	timeout	-	3600

Table D.8.: Comparison of Approximation Methods for Transient Flow Calculations on the GasLib-40 Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	3.98e-06	0.00e+00	984.74
NL-BARON	infeasible	-	23.94
NL-SCIP	1.31e-10	3.43e-10	5.35
ITERATE	1.20e+00	3.28e-05	38.88
CC-3	timeout	-	3600
CCLOG-3	timeout	-	3600
CCLOG-5	timeout	-	3600
CCLOG-7	timeout	-	3600
DCC-3	timeout	-	3600
DLOG-3	timeout	-	3600
INC-3	timeout	-	3600
MC-3	timeout	-	3600

Table D.9.: Comparison of Approximation Methods for Transient Flow Calculations on the GasLib-134 Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

Solution Method	$r_{\max}$	$\Delta_{\max}$	Time (s)
NL-0	1.70e-06	0.00e+00	372.56
NL-BARON	5.51e-09	7.47e-09	7.12
NL-SCIP	timeout	-	3600
ITERATE	9.14e+01	4.03e-04	71.40
CC-3	timeout	-	3600
CCLOG-3	timeout	-	3600
CCLOG-5	timeout	-	3600
CCLOG-7	timeout	-	3600
DCC-3	timeout	-	3600
DLOG-3	timeout	-	3600
INC-3	timeout	-	3600
MC-3	timeout	-	3600

Table D.10.: Comparison of Approximation Methods for Transient Flow Calculations on the GasLib-135 Network.
We include the maximal nonlinear residual ($r_{\max}$), the relative difference to the solution NL-0 ($\Delta_{\max}$), and the solution time.

E. Deutsche Zusammenfassung

Erdgas ist eine der meistgenutzten Energiequellen in Deutschland. Daher müssen große Mengen an Gas von Produzenten zu Verbrauchern transportiert werden.

In dieser Arbeit vergleichen wir verschiedene Algorithmen um Fluss- und Druckwerte in einem Gasnetzwerk zu berechnen. Im Unterschied zu sog. stationären Rechnungen, bei welchen eine konstante Kapazität über einen unbegrenzent Zeitraum nachgefragt wird, beschäftigen wir uns mit transienten Problemen. Gegeben sind ein Gasnetzwerk, sowie Gaszufuhr- und Gasentnahmewerte über einen spezifizierten Zeitraum. Passend zu diesen Bedarfswerten wollen wir Druck- und Flusswerte im Netzwerk berechnen, sodass die gegebenen Bedarfe von der Versorgung gedeckt werden können. Die Schwierigkeit darin besteht in dem physikalischen Verhalten von Gas. Der Gasfluss wird beschrieben durch die sogenannten Euler-Gleichungen. Diese bestehen aus einem System von nichtlinearen partiellen Differentialgleichungen.

Wir beschäftigen uns ausschließlich mit Rohrnetzwerken (d.h. andere Elemente wie z.B. Kompressoren werden vernachlässigt).

Die Arbeit beginnt mit einer Einführung in das Thema und einem Literaturüberblick zu bisherigen Linearisierungsverfahren im Gaskontext.

Im zweiten Kapitel der Arbeit führen wir die Notation sowie die physikalischen Beziehungen für Gasfluss in Rohren ein. Wir besprechen die Euler Gleichungen, und leiten eine Vereinfachung der Eulergleichungen ab (das sog. “friction dominated” Modell) und wir berechnen, basierend auf dem implizite Box-Verfahren, die Diskretisierung die wir im Rest der Arbeit benutzen werden.

Das dritte Kapitel führt (auf gemischt-ganzzahliger Programmierung basierende) Formulierungen für stückweise lineare Funktionen ein. Wir erklären wie diese stückweise linearen Formulierungen zur Approximation der Nichtlenaritäten in den Gasflussgleichungen eingesetzt werden kann.

In Kapitel 4 besprechen wir eine eigene iterative Methode zur Lösung der Gasflussgleichungen. Die Methode basiert auf der Linearisierung der Impulsgleichung, indem wir die absolute Gasflussgeschwindigkeit fixieren. Diese Methode nennen wir “iterative velocity approximation” (iterative Geschwindigkeitsapproximation). Wir beweisen Konvergenz des Verfahrens auf einzelnen Rohren.

Im fünften Kapitel testen wir das Konvergenzverhalten von “iterative velocity approximation” auf komplexeren Netzwerken (d.h. gekoppelte Rohre). Des Weiteren vergleichen wir die verschiedenen Ansätze zur Lösung des Gasflusssystems. Dazu vergleichen wir Lösungsgeschwindigkeit und die Güte der Lösung (durch Messung der nichtlinearn Residuums) auf neun verschiedenen Netzwerken. Wir betrachten generische Lösungsverfahren zur Lösung nichtlinearer Programme wie sie von den Lösern SCIP und BARON eingesetzt werden, sowie alle stückweise Linearisierungen aus dem dritten Kapitel und das “iterative

velocity approximation" Verfahren.

Im letzten Kapitel bewerten wir unsere Ergebnisse und geben einen Ausblick in weitere Forschungsideen.

List of Figures

List of Tables

Bibliography

[Amann and Escher, 1998] Amann, H. and Escher, J. (1998). *Analysis I.* Basel: Birkhäuser.

[Barber et al., 1996] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483.

[Bosch, 2013] Bosch, S. (2013). *Algebra.* Berlin: Springer Spektrum.

[Brouwer et al., 2011] Brouwer, J., Gasser, I., and Herty, M. (2011). Gas pipeline models revisited: Model hierarchies, nonisothermal models, and simulations of networks. *Multiscale Modeling & Simulation*, 9:601–623.

[Burlacu et al., 2017a] Burlacu, R., Egger, H., Groß, M., Martin, A., Pfetsch, M., Schewe, L., Sirvent, M., and Skutella, M. (2017a). Maximizing the storage capacity of gas networks: a global minlp approach. `https://opus4.kobv.de/opus4-trr154/files/221/max_storage_capacity-opus.pdf`. [Preprint; accessed 2018-12-06].

[Burlacu et al., 2017b] Burlacu, R., Geißler, B., and Schewe, L. (2017b). Solving Mixed-Integer Nonlinear Programs using Adaptively Refined Mixed-Integer Linear Programs. `http://www.optimization-online.org/DB_HTML/2017/05/6029.html`. [Preprint; accessed 2018-12-06].

[Correa-Posada and Sánchez-Martín, 2014] Correa-Posada, C. M. and Sánchez-Martín, P. (2014). Gas Network Optimization: A comparison of Piecewise Linear Models. `http://www.optimization-online.org/DB_HTML/2014/10/4580.html`. [Preprint; accessed 2018-12-06].

[Domschke et al., 2011] Domschke, P., Geißler, B., Kolb, O., Lang, J., Martin, A., and Morsi, A. (2011). Combination of nonlinear and linear optimization of transient gas networks. *INFORMS J. Comput.*, 23(4):605–617.

[Domschke et al., 2017] Domschke, P., Hiller, B., Lang, J., and Tischendorf, C. (2017). Modellierung von Gasnetzwerken: Eine übersicht. `https://opus4.kobv.de/opus4-trr154/files/191/TR2017-DomschkeHillerLangTischendorf-mDeckblatt.pdf`. [Preprint; accessed 2018-12-06].

[Ehrhardt and Steinbach, 2005] Ehrhardt, K. and Steinbach, M. C. (2005). Nonlinear optimization in gas networks. In Bock, H. G., Phu, H. X., Kostina, E., and Rannacher, R., editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 139–148, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Eurostat, 2018] Eurostat (2018). Final energy consumption by product. `https://ec.europa.eu/eurostat/tgm/refreshTableAction.do?tab=table&plugin=1&pcode=ten00095&language=en`. [Online; accessed 2018-12-06].

[Finnemore and Franzini, 2002] Finnemore, E. J. and Franzini, J. B. (2002). *Fluid mechanics with engineering applications*, volume 10. McGraw-Hill New York.

[Fügenschuh et al., 2015] Fügenschuh, A., Geißler, B., Gollmer, R., Morsi, A., Pfetsch, M. E., Rövekamp, J., Schmidt, M., Spreckelsen, K., and Steinbach, M. C. (2015). Physical and technical fundamentals of gas networks. In *Evaluating gas network capacities*, pages 17–43. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: Mathematical Optimization Society.

[Fügenschuh et al., 2009] Fügenschuh, A., Geißler, B., Martin, A., and Morsi, A. (2009). The transport pde and mixed-integer linear programming. In Barnhart, C., Clausen, U., Lauther, U., and Möhring, R. H., editors, *Models and Algorithms for Optimization in Logistics*, number 09261 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[Geißler et al., 2011] Geißler, B., Kolb, O., Lang, J., Leugering, G., Martin, A., and Morsi, A. (2011). Mixed integer linear models for the optimization of dynamical transport networks. *Mathematical Methods of Operations Research*, 73(3):339.

[Geißler et al., 2012] Geißler, B., Martin, A., Morsi, A., and Schewe, L. (2012). Using Piecewise Linear Functions for Solving MINLPs. In Lee J, L. S., editor, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 287–314. Springer Science+Business Media, New York.

[Geißler et al., 2015] Geißler, B., Martin, A., Morsi, A., and Schewe, L. (2015). The MILP-relaxation approach. In *Evaluating gas network capacities*, pages 103–122. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: Mathematical Optimization Society.

[Gleixner et al., 2018] Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R. L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M. E., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J. M., Walter, M., Wegscheider, F., Witt, J. T., and Witzig, J. (2018). The SCIP Optimization Suite 6.0. Technical report, Optimization Online.

[Hart et al., 2017] Hart, W. E., Laird, C. D., Watson, J.-P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L., and Siirola, J. D. (2017). *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, second edition.

[Hart et al., 2011] Hart, W. E., Watson, J.-P., and Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260.

[Hennings, 2018] Hennings, F. (2018). Benefits and limitations of simplified transient gas flow formulations. In *Operations Research Proceedings 2017*, volume Operations Research Proceedings, pages 231 – 237.

[Huchette and Vielma, 2017a] Huchette, J. and Vielma, J. P. (2017a). A mixed-integer branching approach for very small formulations of disjunctive constraints. [Preprint; accessed 2018-12-06].

[Huchette and Vielma, 2017b] Huchette, J. and Vielma, J. P. (2017b). Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. [Preprint; accessed 2018-12-06].

[Humpola et al., 2015] Humpola, J., Fügenschuh, A., Hiller, B., Koch, T., Lehmann, T., Lenz, R., Schwarz, R., and Schweiger, J. (2015). The specialized MINLP approach. In *Evaluating gas network capacities*, pages 123–143. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: Mathematical Optimization Society.

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

[IBM ILOG CPLEX Division, 2017] IBM ILOG CPLEX Division (2017). CPLEX User's Manual. `https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/Optimization_Studio/topics/PLUGINS_ROOT/ilog.odms.studio.help/pdf/usrcplex.pdf`. [Online; accessed 2018-12-05].

[Jeroslow, 1987] Jeroslow, R. G. (1987). Representability in mixed integer programming. I: Characterization results. *Discrete Appl. Math.*, 17:223–243.

[Jeroslow and Lowe, 1984] Jeroslow, R. G. and Lowe, J. K. (1984). Modelling with integer variables. *Math. Program. Study*, 22:167–184.

[Jones et al., 01] Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. `http://www.scipy.org/`. [Online; accessed 2018-12-06].

[Kelley, 1995] Kelley, C. (1995). *Iterative Methods for Linear and Nonlinear Equations.* Society for Industrial and Applied Mathematics.

[Knuth, 2011] Knuth, D. E. (2011). *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1.* Pearson Education India.

[Koch et al., 2015] Koch, T., Hiller, B., Pfetsch, M., and Schewe, L., editors (2015). *Evaluating gas network capacities.* Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: Mathematical Optimization Society.

[Kolb et al., 2010] Kolb, O., Lang, J., and Bales, P. (2010). An implicit box scheme for subsonic compressible flow with dissipative source term. *Numerical Algorithms*, 53(2):293–307.

[Králik et al., 1988] Králik, J., Stiegler, P., Vostry, Z., and Závorka, J. (1988). Dynamic modeling of large-scale networks with application to gas distribution. *Studies in Automation and Control.*

[Lowe, 1984] Lowe, J. K. (1984). *Modelling with Integer Variables.* PhD thesis, Georgia Institute of Technology.

[Markowitz and Manne, 1957] Markowitz, H. M. and Manne, A. S. (1957). On the solution of discrete programming problems. *Econometrica: journal of the Econometric Society*, pages 84–110.

[Martin et al., 2006] Martin, A., Möller, M., and Moritz, S. (2006). Mixed Integer Models for the Stationary Case of Gas Network Optimization. *Mathematical Programming*, 105(2):563–582.

[Meyer, 1976] Meyer, R. R. (1976). Mixed integer minimization models for piecewise-linear functions of a single variable. *Discrete Math.*, 16:163–171.

[Moritz, 2007] Moritz, S. (2007). *A mixed integer approach for the transient case of gas network optimization.* Darmstadt: Univ. Darmstadt, Fachbereich Mathematik (Diss.).

[Nikuradse, 1950] Nikuradse, J. (1950). Laws of flow in rough pipes. *National Advisory Committee for Aeronautics Washington.*

[Papay, 1968] Papay, J. (1968). A termelés technológiai paraméterek változása a gáztelepek muvelése során. *OGIL MUSZ, Tud, Kuzl., Budapest*, pages 267–273.

[Pratt and Wilson, 1984] Pratt, K. and Wilson, J. (1984). Optimisation of the operation of gas transmission systems. *Transactions of the Institute of Measurement and Control*, 6(4):261–269.

[Ríos-Mercado and Borraz-Sánchez, 2015] Ríos-Mercado, R. Z. and Borraz-Sánchez, C. (2015). Optimization problems in natural gas transportation systems: A state-of-the-art review. *Applied Energy*, 147:536 – 555.

[Sahinidis, 2017] Sahinidis, N. V. (2017). *BARON 17.10.16: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual. `http://www.minlp.com/downloads/docs/baron%20manual.pdf` [Online; accessed 2018-12-06].

[Saleh, 2002] Saleh, J. (2002). *Fluid flow handbook.* McGraw-Hill Professional.

[Schewe et al., 2015] Schewe, L., Koch, T., Martin, A., and Pfetsch, M. E. (2015). Mathematical optimization for evaluating gas network capacities. In *Evaluating gas network capacities*, pages 87–102. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM); Philadelphia, PA: Mathematical Optimization Society.

[Schmidt et al., 2017] Schmidt, M., Aßmann, D., Burlacu, R., Humpola, J., Joormann, I., Kanelakis, N., Koch, T., Oucherif, D., Pfetsch, M. E., Schewe, L., Schwarz, R., and

Sirvent, M. (2017). GasLib – A Library of Gas Network Instances. *Data*, 2(4):article 40.

[Stolwijk and Mehrmann, 2018] Stolwijk, J. J. and Mehrmann, V. (2018). Error analysis and model adaptivity for flows in gas networks. *Analele Universitatii" Ovidius" Constanta-Seria Matematica*, 26(2):231–266.

[Tawarmalani and Sahinidis, 2005] Tawarmalani, M. and Sahinidis, N. V. (2005). A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249.

[The Sage Developers, 2018] The Sage Developers (2018). *SageMath, the Sage Mathematics Software System (Version 8.2)*. `"http://www.sagemath.org"`. [Online; accessed 2018-12-06].

[Todd, 1977] Todd, M. J. (1977). Union jack triangulations. In KARAMARDIAN, S., editor, *Fixed Points*, pages 315 – 336. Academic Press.

[van der Hoeven, 2004] van der Hoeven, T. (2004). *Math in Gas and the art of linearization*. PhD thesis, Rijksuniversiteit Groningen.

[Vielma et al., 2010] Vielma, J. P., Ahmed, S., and Nemhauser, G. (2010). Mixed-integer models for nonseparable piecewise linear optimization: unifying framework and extensions. *Operations Research*, 58:303–315.

[Vielma and Nemhauser, 2008] Vielma, J. P. and Nemhauser, G. (2008). Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. In Lodi, A., Panconesi, A., and Rinaldi, G., editors, *Proceedings of the 13th Conference on Integer Programming and Combinatorial Optimization (IPCO 2008)*, volume 5035 of *Lecture Notes in Computer Science*, pages 199–213.

[Vielma and Nemhauser, 2011] Vielma, J. P. and Nemhauser, G. (2011). Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128:49–72.

[Wächter and Biegler, 2006] Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

[Wilson, 1998] Wilson, D. L. (1998). *Polyhedral methods for piecewise-linear functions.* PhD thesis, University of Kentucky.