Chapter 11

# What does “feasible” mean?

Imke Joormann, Martin Schmidt, Marc C. Steinbach,
Bernhard M. Willert

**Abstract** *The main goal of our efforts described in this book consists of solving the problem of validation of nominations in gas networks, i.e., deciding whether a feasible solution exists for a given set of boundary conditions represented by a nomination. However, it turns out that the meaning of “feasible” is not self-evident. This is due to a multitude of reasons, ranging from the accuracy of problem data over subtle differences between our models to tractability of optimization problems. In the current chapter we elaborate on these points and try to clarify precisely in what sense and how well the mathematical methodology presented can distinguish feasible from infeasible solutions.*

This chapter treats a number of topics addressing feasibility issues. Following the process from modeling the gas network and its operation to a solution in our models, we highlight the different stages where feasibility might be a concern. We start with Section 11.1 by discussing important differences between reality and our mathematical optimization models, as well as the practical interpretation of the results. Section 11.2 then addresses the interrelation of modeling accuracy and requirements on the availability and accuracy of model data. We then turn to the various mathematical difficulties concerning feasibility arising in our planning problems and how they are addressed by considering a hierarchy of abstract and computational models. This is discussed in Section 11.3 with a focus on the relations of these models to each other. We then present selected results comparing our nonlinear program (NLP) validation model with the commercial simulation package SIMONE in Section 11.4. In Section 11.5, we explain how infeasible results of the ValNLP (see Chapter 10) are postprocessed in order to give the user hints for possible reasons of the infeasibility. Finally, the last section is devoted to provably infeasible problem instances: Section 11.6 presents techniques for analyzing infeasibility in relaxed models.

## 11.1 ▪ Feasible network operation

“What is feasible network operation?” is one of the most fundamental questions of this book. This question needs to be answered in order to define the capacity of a gas network. It is thus behind every modeling and algorithmic decision that has been made. There is a precise (and trivial) answer/definition: a network operation is feasible if all technical, legal, and contractual conditions of the given gas network are satisfied. In this section we highlight several issues arising from this definition and give connections with the different parts of this book.

The first issue is that, naturally, the above definition implicitly includes a load flow situation and external conditions like temperatures, etc. In short-term planning, the load flow and external conditions must be known, or sufficiently accurate forecasts must be available, to generate suitable network operation plans. Since we concentrate on long-term planning in this book, the above data are not available in our context: we consider fictitious future *nominations*, where load flows and external conditions are assumed to be constant over a fixed time period. For the evaluation of capacities, a complete probability space of such nominations has to be generated using statistical data from the past (see Chapter 13) in combination with methods to include contractual data and forecasts for points for which no statistical data are available (see Chapter 14); see also Chapter 4 for a description of the approach currently used in practice.

An important second issue is that network operation is transient in practice, i.e., dynamic over time. The most detailed evaluation of capacities would thus consider transient computations. However, this would require us to replace the probability space of nominations with a probability space of dynamic load flow situations, each consisting of

▹ a dynamic profile of load flows and external conditions over the considered time period,
▹ an initial state of the system, and
▹ an optimization engine that can handle the transient computations.

It is entirely unclear whether it is possible to avoid the “curse of dimensionality” in such an attempt and whether sufficiently reliable probabilities can be generated from the data available. Moreover, the network sizes currently solvable by transient methods that handle discrete decisions are small (see, e.g., Domschke et al. (2011)); simulation and optimization without discrete decisions can currently handle larger sizes (see Ehrhardt and Steinbach (2005); Steinbach (2007)), but would need improvements as well. See also the discussion in Section 5.3.1.

As already discussed in Section 5.3.1, we therefore concentrate on stationary models and computations. While this is currently the only way to go, it has the drawback that certain temporal effects cannot fully be represented in stationary models. Examples are as follows:

▹ In contrast to a stationary model, entry and exit flows do not have to be balanced.
▹ It is sometimes possible to operate a turbo compressor with a working point left to its surgeline. This is realized by redirecting part of the compressor outflow back to its inlet, which results in a circular gas flow whose temperature is increased repeatedly; see Section 2.3.5.1. There is no adequate stationary model for this process: it requires an artificial relaxation of the stationary operating range of the turbo compressor, while the dynamic operating range is never violated in practice.
▹ Temporarily storing gas in a pipeline, the so-called linepack, cannot be represented in stationary models. It is, however, of great practical importance for gas network operation, since the pipes themselves can store a significant amount of gas and thus act as a large buffer to smoothen short-term demand peaks.

Consequently, stationary models are usually considered as being more conservative, since transient operation offers more flexibility. Moreover, a stationary network state is a special case of a transient state in which the load flows and external conditions do not change over time and the network reaches a steady state. However, it is possible that a sequence of nominations, whose corresponding stationary models are all feasible, cannot dynamically be operated. Usually, however, stationary planning provides practically reliable results. It is an important goal of future research to develop transient methods for short-term planning.

In order to focus our discussion of feasibility, we from now on consider the case of a stationary computation with a given nomination.

The third issue connected with the feasibility definition concerns the choice of the model for computations. All considered models are based on the physical laws, usually the Euler equations, which describe gas behavior and flow. However, as is obvious from the discussion in Chapter 2, several approximations and simplifications have to be made. For instance, the choice of one-dimensional (1d) Euler equations instead of a 3d fluid dynamical model is essential to be able to perform computations on networks larger than a few pipelines. However, the 1d Euler equations are usually considered as reasonably exact for practical computations on larger networks. Other model choices are based on computational reasons—this is discussed in detail in Chapters 6–10. Note that these facts imply that in practice there is nothing like a natural "master model" for the whole network from which one can then derive approximations.

As a consequence, any approach has to deal with the *model gap* between reality and the actually chosen model. This, of course, has implications on the definition of feasibility, which is with respect to the chosen model. This model can describe reality more or less accurately. In this book, we take the approach to define a base model, against which we validate all other models. Thus, we define a network operation as feasible if we can find a solution of the validation NLP (ValNLP, see Section 10.4) using computed/given discrete decisions that has a zero objective value. See Chapter 12 for more information of the performance with respect to this definition.

As a final issue with respect to feasibility, we mention the limited numerical accuracy with which all computations are performed. Although not easy in detail, the handling of this issue is standard.

## 11.2 ▪ Availability and accuracy of model data

Theoretically it would be possible to formulate a microscopic network model that accounts for the full physics of all the individual chemical species contained in natural gas and for the full three-dimensional geometry of the pipeline network, including special devices such as valves, compressors, control valves, filters, measurement instruments, etc.

Clearly, such an extremely detailed model would be of little practical use. First, it is impossible to gather reasonably accurate values of all the required physical and technical data: just think of the spatially varying heat conductance of the soil around the pipelines, or of technical network parameters that change slowly over time, because of usage conditions or wearout, like the roughness of inner pipe walls or the efficiency of compressors. Second, even if accurate and complete physical and technical data could be provided for (typically large) real-life networks, a numerical simulation or optimization matching the accuracy of such a model would be far beyond the capabilities of today's software and algorithms.

Fundamental considerations like these apply to most complex technical systems: modelers always face the difficulty of providing a *reasonable* degree of detail (or model accuracy) in the sense that the value of information drawn from the model by computational simulation and optimization outweighs the cost of setting up the model (complete with required data) plus the cost of performing the computations. In the case of gas networks, it turns out that

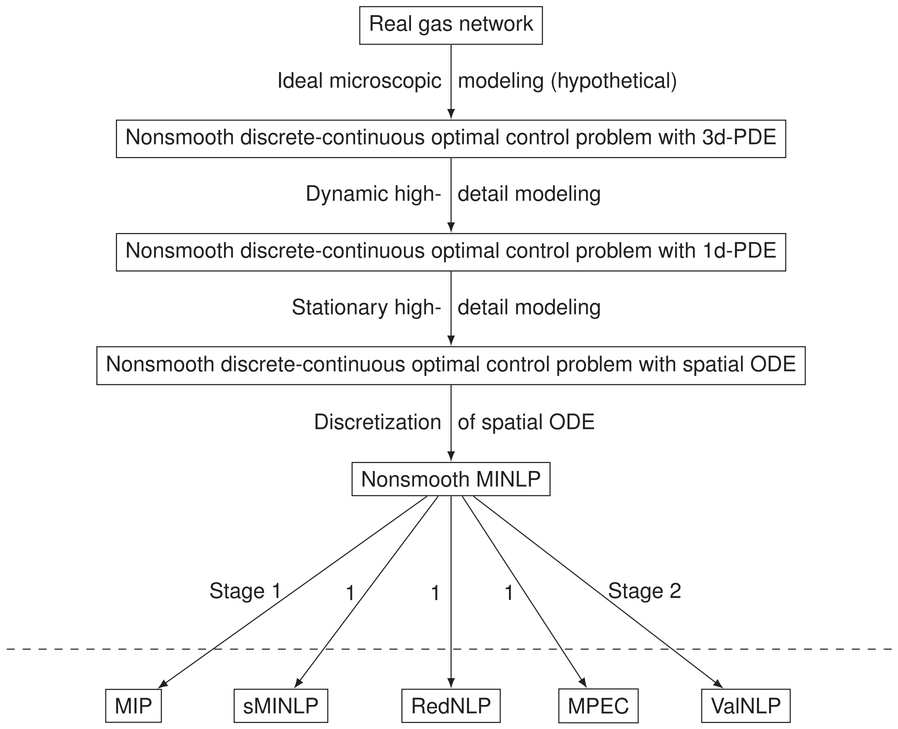

Real gas network
Ideal microscopic modeling (hypothetical)
Nonsmooth discrete-continuous optimal control problem with 3d-PDE
Dynamic high-detail modeling
Nonsmooth discrete-continuous optimal control problem with 1d-PDE
Stationary high-detail modeling
Nonsmooth discrete-continuous optimal control problem with spatial ODE
Discretization of spatial ODE
Nonsmooth MINLP
Stage 1
1
1
1
Stage 2
MIP
sMINLP
RedNLP
MPEC
ValNLP

**Figure 11.1.** *Overview of optimization models: abstract models are above the dashed line, computational models below the dashed line.*

▷ it is rather difficult to find a properly balanced degree of modeling detail;
▷ the degree of detail depends heavily on the intended use (on-line simulation, short-term planning, mid-term planning, ...) and on the algorithms used;
▷ even models used for on-line simulation are much coarser than the "ideal" microscopic model mentioned above;
▷ providing and maintaining the data of a real gas network requires a substantial effort even for relatively coarse models;
▷ the accuracy of results is typically limited by the accuracy of the data, not the accuracy of algorithms, even for relatively coarse models.

The discussion in this chapter is primarily focused on *deterministic* network data. All mid-term and long-term planning problems involve inherently *stochastic* data such as future supplies, discharges, and temperatures, which are the subject of Chapter 13.

## 11.3 ▪ How "feasible" are solutions of our models?

When we speak about feasibility of solutions, we have to consider both feasibility with respect to a certain model and feasibility with respect to reality. For a proper understanding, we need to take a closer look at the interrelations of the models in our hierarchy (see Figure 11.1). Our models basically need to address the following combination of difficult mathematical aspects:

▷ nonlinearity,
▷ nonconvexity,

- ▹ nonsmoothness,
- ▹ PDEs or ODEs for gas dynamics,
- ▹ continuous controls,
- ▹ discrete decisions.

All these aspects arise in the hypothetical microscopic network model mentioned before, which would have the form of a (nonlinear and nonconvex) nonsmooth discrete-continuous control problem with a system of gas dynamics PDEs in three spatial dimensions. All aspects still arise in a practical "reference" model, the next-lower level in Figure 11.1, which differs from the ideal model in that gas dynamics are only modeled in the pipes and only in one spatial dimension, and in that macroscopic technical models are used for all other network elements. As a side issue we would like to point out that even for such a transient one-dimensional model, theoretical results on existence and uniqueness of solutions for arbitrary networks are still unavailable to the best of our knowledge; for some of the most general studies see Colombo, Herty, and Sachers (2008), Colombo et al. (2009), and Gugat et al. (2012).

In the *stationary* case considered here, the 1d PDE actually reduces to a spatial ODE. Even in gas network *simulation* (see Králik et al. (1988); Záworka (1993)) one typically uses a priori discretizations of the PDEs (or ODEs), which in our context leads to a nonsmooth mixed-integer nonlinear program (MINLP) model, the lowest level in the upper part of Figure 11.1. This is still an abstract model, in the sense that we do not actually use it directly for computations. However, all our computational models are derived from it by further approximations, simplifications, and smoothings of the above mathematical difficulties; see lower part of Figure 11.1. (If applicable, further intermediate models in the derivation are described in the context of each computational model.) The first four computational models feature substantial simplifications to obtain tractable formulations of the overall discrete-continuous optimization problem; see Section 5.6 for an overview and Chapters 6–9 for details. In contrast, the validation NLP model ValNLP (Section 10.4) does not include discrete decisions but retains as much of the nonlinear physics as deemed necessary to check real-life feasibility with sufficient accuracy. For our purpose, this means that the degree of detail and the resulting solution accuracy are comparable to that of existing simulation packages (see Section 11.4 for a comparison), which have been trusted for years by practitioners to be reliable approximations of reality.

The fundamental difficulty in assessing real-life feasibility is that the different models in the hierarchy are not *quantitatively* comparable: although some of the differences are proper mathematical approximations for which error estimates exist (such as discretizations of the spatial ODE), other model differences just consist of different equations defined on different spaces (i.e., sets of variables) for which natural error measures do not even exist. The same point applies to comparisons of the four first-stage models with each other and with the ValNLP model: they all live on different spaces of variables, possibly including discrete variables, so that a natural difference measure for solutions is not clear. Thus, for comparing the results of our models with each other and for interpreting their implications in real life, we need to rely on experience with these models in practical application.

For these reasons, we consider a candidate solution from any of the four approaches as feasible with respect to our mathematical framework if fixing its discrete decisions and adding variables of the (finer) validation NLP model ultimately leads to a feasible point: a ValNLP solution with slack norm zero; see Section 10.3 and Section 5.6. Of course, there remains some unavoidable "grey area": a chance that an NLP-feasible point is slightly infeasible in real life. Such a grey area exists in simulation-based planning as well. Since the comparison with state-of-the-art simulation software is the best possible benchmark available to us, we basically try to produce decisions that are close to practically accepted simulation-based decisions.

If all four approaches fail to produce a candidate solution, due to divergence in the RedNLP and MPEC approaches or due to timeout in the MILP and sMINLP approaches, we cannot make a decision on feasibility and the runs are repeated with different parameter settings.

Note that the MILP and sMINLP approaches are actually capable of detecting *infeasibility* of their respective models, i.e., we may obtain a decisive *negative* answer to the feasibility question. This will be discussed in the last section of this chapter.

## 11.4 ▪ NLP validation vs. network simulation

The fundamental difficulty that our abstract and computational models are not quantitatively comparable (see Section 11.3) also applies to the relation of the ValNLP model with simulation models employed in commercial packages.

Ideally, the ValNLP model should reflect the real network behavior as closely as possible. In principle, this can be achieved by setting up an *inverse problem*, i.e., an optimization problem that estimates the model parameters from the mismatch between the given network model and measurements of the real system. However, measurements of *stationary* network states cannot be obtained during regular operation, and are therefore very expensive. On the other hand, to estimate parameters from *dynamic* measurements, we would have to develop a proper transient network model and to solve a corresponding dynamic inverse problem, both of which are way beyond the scope of the work described here. As an artificial reference substituting the real system we have hence chosen a particular simulation package, namely SIMONE version 5.73 [SIMONE], which is frequently used by Open Grid Europe GmbH (OGE) and other network operators.

In the following we shall demonstrate that, even when our NLP uses the model equations given in the SIMONE documentation, small but nonnegligible differences between the results of the NLP and SIMONE occur. These differences can be attributed to several factors:

- ▹ the overall numerical solution algorithms differ,
- ▹ differential equations are discretized using different schemes or grids,
- ▹ model variants like the pipe friction model may differ in minor aspects,
- ▹ data handling and the accuracy of data processing differ, and
- ▹ SIMONE possibly performs floating point computations in single precision arithmetic whereas all our computations are performed in double precision.

To test how well SIMONE and the various submodel variants of our NLP match, in particular the submodel variants used in ValNLP, we have performed a large number of comparisons on individual network elements. The tests cover all element types of Chapter 2, with multiple sets of design parameters and flow situations for each of them. An illustrative subset of these tests is discussed below.

For every comparison we fix all inflow quantities of the network element under consideration. Additionally, we fix all control quantities in case of active devices. It can be shown that all other quantities are thereby uniquely determined.

As it turns out, choosing suitable test sets of design parameters and flow situations is hard. Due to the nonlinearity of the model equations, small differences of intermediate values caused by the above-mentioned factors can get amplified during computation. Even worse, for the pipes one can create extreme cases with almost arbitrarily large (relative)

**Table 11.1.** *Parameters of test set for pipes.*

| Quantity | Tested values | Unit |
|---|---|---|
| $L$ | 0.01, 0.1, 0.9, 46.0 | km |
| $D$ | 150, 310, 405, 1185 | mm |
| $k$ | 0.006, 0.02, 0.1, 0.5 | mm |
| $h_{\text{out}}$ | −500, 0, 500 | m |
| $p_{\text{in}}$ | 3.9, 15.3, 53.8, 74.4 | bar |
| $T_{\text{in}}$ | 288.15, 298.15, 308.15, 318.15 | K |
| $Q_0$ | 50, 250, 500, 750 | $1000\text{Nm}^3/\text{h}$ |

differences in the final results. This happens, e.g., when considering the outflow pressure of very long pipes with small diameter.

We wish to perform a comparison that covers real-life network elements and flow situations as well as possible on the one hand and that avoids extreme cases on the other hand. In case of testing pipes we therefore proceed as follows. For the northern H-gas network of OGE (see Section 1.6), we take the distributions of pipe length, diameter, and roughness, and choose as test parameters the respective quantiles at 10%, 35%, 65%, and 90%. For the outlet elevation $h_{\text{out}}$, we choose the values −500 m, 0 m, and 500 m, the inlet elevation being fixed at 0 m. The resulting sets of individual parameters are given in Table 11.1. The entire test set is obtained by taking all 12 288 combinations of the individual pipe and flow parameters. Test sets of other network elements are generated in a similar manner. In addition to pipes, we will also address control valves here. More detailed results including all other element types are given in Schmidt, Steinbach, and Willert (2014).

For every type of network element, several quantities of the computed solutions are compared, such as outflow pressure and temperature. For every quantity $x$, we measure the absolute deviation between the NLP and SIMONE results $x_{\text{NLP}}$ and $x_{\text{Sim}}$, the relative deviation, and (if applicable) the relative deviation with respect to the quantity's change from inflow to outflow, $\Delta x = x_{\text{in}} - x_{\text{out}}$:

$$\begin{aligned} d_{\text{abs}}(x) &= |x_{\text{NLP}} - x_{\text{Sim}}|, \\ d_{\text{rel}}(x) &= \frac{|x_{\text{NLP}} - x_{\text{Sim}}|}{|x_{\text{Sim}}|}, \\ d_{\text{rel}}(\Delta x) &= \frac{|\Delta x_{\text{NLP}} - \Delta x_{\text{Sim}}|}{|\Delta x_{\text{Sim}}|}. \end{aligned}$$

If multiple model variants exist in both the NLP and SIMONE, several choices are compared. For instance, in the case of pipes we obtain four model variants by combining the AGA formula (10.4) and Papay's formula (10.5) for the compressibility factor $z(p,T)$ with two variants for the Euler equations: the approximation (10.25) and discretized ODE (10.20). Each of these four variants is used in all 12 288 test cases. Finally we exclude all test cases where one of the methods fails to produce a result and all combinations with an impossible outlet elevation, i.e., with $|h_{\text{out}}| > L$. Average deviations are then computed on the remaining test cases. In all instances, the settings (like compressibility factor or friction model) of the NLP and SIMONE are matched as closely as possible. Note that this will yield poor matchings in certain cases: while both the NLP and SIMONE offer the AGA formula and Papay's formula for the compressibility coefficient,

**Table 11.2.** *Average deviations of pipe outflow pressure (*bar*).*

| NLP model choice | $z(p,T)$ | Sample size | $d_{\text{abs}}(p_{\text{out}})$ | $d_{\text{rel}}(p_{\text{out}})$ | $d_{\text{rel}}(\Delta p)$ |
|---|---|---|---|---|---|
| Approximation | AGA | 2419 | 0.11 | 0.86% | 5.8% |
| Approximation | Papay | 2651 | 0.10 | 0.94% | 5.9% |
| ODE | AGA | 2658 | 0.046 | 0.31% | 3.2% |
| ODE | Papay | 2659 | 0.050 | 0.33% | 3.3% |

**Table 11.3.** *Average deviations of pipe outflow temperature (*K*).*

| NLP model choice | $z(p,T)$ | Sample size | $d_{\text{abs}}(T_{\text{out}})$ | $d_{\text{rel}}(T_{\text{out}})$ | $d_{\text{rel}}(\Delta T)$ |
|---|---|---|---|---|---|
| Approximation | AGA | 2419 | 1.1 | 0.38% | 16% |
| Approximation | Papay | 2651 | 1.0 | 0.35% | 15% |
| ODE | AGA | 2658 | 0.22 | 0.075% | 4.6% |
| ODE | Papay | 2659 | 0.15 | 0.051% | 2.9% |

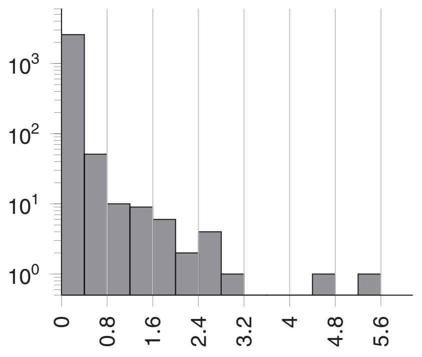
for instance, SIMONE always uses an ODE model for the pipe flow. The approximation model of our NLP (which is actually used in ValNLP) has no counterpart and is expected to be less accurate than the ODE model of SIMONE. In cases like this, the measured average deviations provide a reasonable indication of the absolute quality of the less accurate model.

The computed average deviations are listed in Table 11.2 and Table 11.3; Figure 11.2 displays logarithmically scaled histograms of the absolute deviations of pressure and temperature for the model variant with ODE discretization and Papay’s formula. The standard deviation of the distribution of absolute pressure differences is 0.24 and the leftmost bin contains 2574 of 2659 samples (97%). The standard deviation of the distribution of absolute temperature differences is 0.29 and the leftmost bin contains 2330 of 2659 samples (88%).

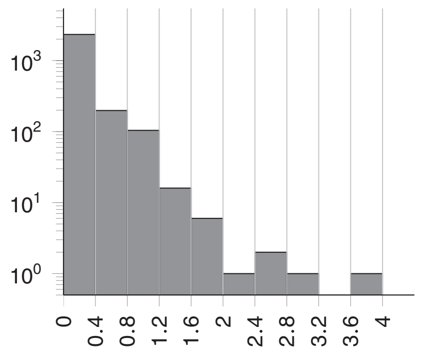
The deviations between the NLP and SIMONE with discretized ODEs are smaller than with the approximating model. This is to be expected, since the ODE discretization is more accurate than the approximation in both cases. Specifically, SIMONE applies an implicit integration method for the partial differential equations (10.15), based on algorithms presented in Králik et al. (1988) and Záworka (1993). Nevertheless, the absolute deviation of pressures of 0.1 bar using the approximation models is still sufficiently accurate for most of the planning tasks addressed in this book. In these cases, one can use the approximation models in order to achieve reduced computation times, as is being done in the ValNLP. If, however, this accuracy was expected to be insufficient, one could switch to the discretized ODE model.

Some pressure and temperature profiles along an exemplary pipe are plotted in Figure 11.3 and Figure 11.4. These graphs show the outflow pressure and temperature depending on the normal volumetric flow along the pipe for an inflow pressure of 74.4 bar at 318.15 K with a soil temperature of 284.15 K. The figures on the right show enlarged details of the most interesting area of the figures on the left, as indicated by the dashed box.

In Figure 11.3 we see that the ODE discretizations of the NLP and SIMONE agree quite well for the pressure profile while the less accurate approximation model of ValNLP is qualitatively similar, but with a larger absolute difference, as expected. This observation is in line with the average deviations given in Table 11.2 and Table 11.3.

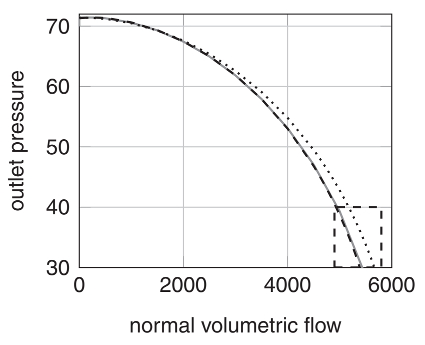


(a) Frequencies of $d_{\text{abs}}(p_{\text{out}})$ in bar.



(b) Frequencies of $d_{\text{abs}}(T_{\text{out}})$ in K.

**Figure 11.2.** *Logarithmic histograms for the tested pipes and the model choices "ODE discretization" and "Papay's formula."*



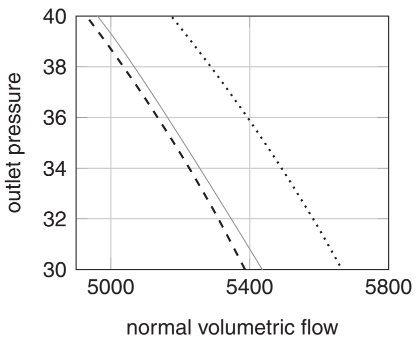


**Figure 11.3.** *Outlet pressure (*bar*) of a pipe ($L = 46$ km, $D = 1185$ mm, $k = 0.006$ mm, slope $s = 0.01$) vs. normal volumetric flow (*1000 $\text{Nm}^3/\text{h}$*), computed using Papay's formula in* SIMONE *v5.73 (——), ODE discretization (- - -), and ODE approximation (·····). Figure on the right is a zoom of the dashed frame in the left figure.*

In Figure 11.4 we see that the ODE discretizations of the NLP and SIMONE agree almost as well for the temperature profile, except for excessive deviations at very small flow values: here SIMONE yields drastically increasing temperatures that cannot be physically correct, which indicates numerical problems or a bug in the tested version. Spot tests suggest that this has been fixed in the release 5.83 of SIMONE. The approximation model of ValNLP is again qualitatively similar. It shows larger absolute differences, but only for medium and large flow values. At small flow values up to 400 000 $\text{Nm}^3/\text{h}$, some curvature information gets apparently lost when approximating the energy equation (10.17c), and the convex section of the temperature graph does not show up as with the ODE discretization. This could be a reason for the larger average deviations between the approximation model of ValNLP and SIMONE as reported in Table 11.3. Note, however, that even these larger average deviations of about 1 K lie within the range of data accuracy, since more accurate forecasts of environmental and soil temperatures required for planning processes are rarely available. Moreover, a small number of tests that we performed with two different versions of SIMONE indicate that the deviations between these two releases of the same software lie also in the same order of magnitude.

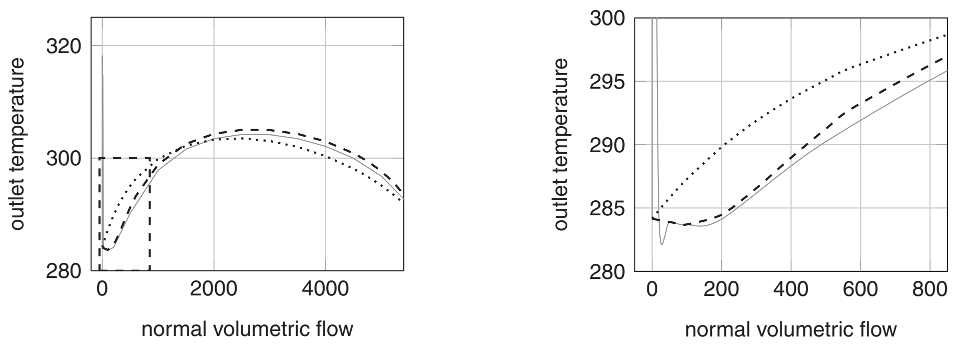


**Figure 11.4.** *Outlet temperature (K) of a pipe ($L = 46$ km, $D = 1185$ mm, $k = 0.006$ mm, slope $s = 0.01$) vs. normal volumetric flow (1000 Nm$^3$/h), computed using Papay's formula in SIMONE v5.73 (——), ODE discretization (- - -), and ODE approximation (·····). Figure on the right is a zoom of the dashed frame in the left figure.*

**Table 11.4.** *Average deviations of control valve outflow temperature (K).*

| $z(p,T)$ | Sample size | $d_{\text{abs}}(T_{\text{out}})$ | $d_{\text{rel}}(T_{\text{out}})$ | $d_{\text{rel}}(\Delta T)$ |
|---|---|---|---|---|
| AGA | 54 | 0.74 | 0.25% | 34% |
| Papay | 54 | 0.094 | 0.032% | 4.1% |

Let us finally consider control valves. Here we fix the inflow pressure, inflow temperature, flow value and controlled pressure reduction, and we compare outflow temperatures. Again, the compressibility factor can be modeled by the AGA formula (10.4) or Papay's formula (10.5). In both the NLP and SIMONE, the temperature loss is computed by the detailed model of Joule–Thomson; see (10.8). Note that there exists a model difference in the case of zero flow, where the “outflow” temperature is set to the soil temperature in SIMONE, whereas it is set to the inflow temperature in the NLP. Thus, we consider only test cases with nonzero flow. In total, there are 108 test cases, all of which produce results with both software packages. The resulting average deviations are listed in Table 11.4.

The differences between the AGA and Papay models are striking: despite the fact that all other quantities and model aspects are identical, the mean deviations with the AGA formula are about eight times as large as the values with Papay's formula. Moreover, the standard deviation of the absolute error for Papay's formula is less than $10^{-2}$ whereas the standard deviation for AGA is 0.63. This strongly indicates differences in the implementation of the AGA formula.

Although the pressure reduction is fixed, there also exists an average absolute deviation of $5 \times 10^{-5}$ bar between the NLP and SIMONE. The NLP reproduces the fixed value correctly; hence, we must conclude that in this case either the internal accuracy of SIMONE is lower or the API of SIMONE rounds output values to a precision of roughly $10^{-4}$ bar. The second explanation seems more likely.

In summary, the validation procedures that we have carried out confirm that our models actually match the level of detail and accuracy provided by SIMONE (and presumably also by other commercial simulation packages). However, the NLP results must always be expected to differ from simulation results even if the models are based on the same equations, since different algorithms and different implementations are involved. Moreover, there is no systematic way of eliminating the remaining differences since the source codes of the simulation packages are inaccessible. Our numerical experiments have shown that the same observations also apply to the comparison of different releases of SIMONE. Results are always roughly comparable, but often not exactly; see Willert (2014) and Schmidt, Steinbach, and Willert (2014).

## 11.5 ▪ The interpretation of ValNLP solutions

Our general workflow for solving the problem of validation of nominations consists of two stages (see Section 5.6). The decision approaches produce solution candidates with discrete decisions and an approximative description of the pressure-flow situation in the network. This discrete-continuous solution is used to fix the discrete decisions of all controllable network devices. In addition, the given solution is used to initialize the variables of the ValNLP model. For this, the situation can be divided into two distinct cases. First, there are variables (like pressure on vertices and flow on arcs) that appear both in the decision approaches and in the validation NLP. For these variables, the solution candidates are used directly to initialize the ValNLP variables. Second, the ValNLP model has additional variables that do not appear in the models of the decision approaches. These variables are initialized in a heuristic way depending on the given pressure and flow values of the solution candidate, the constraints of the ValNLP model, and suitable constants.

As it is described in Section 10.3, the original ValNLP is relaxed by using slack variables for a certain (sub)set of constraints, and the $\ell_1$-norm of the slack variable vector is minimized. When solving the relaxed ValNLP model, three different outcomes are possible:

1. The ValNLP model is solved to (local) optimality with a slack norm value of zero. In this case, we have provably found a feasible solution to the underlying nonsmooth and nonlinear mixed-integer feasibility problem.
2. The ValNLP model is not solved to (local) optimality.
3. The ValNLP model is solved to (local) optimality with a slack norm value larger than zero.

The last two cases require further consideration. If no (locally) optimal solution could be found at all, we can only state that the solution candidate could not be validated by the ValNLP model. This can happen for various reasons, such as improper starting points, poor problem scaling, ill-conditioning, etc.

If we have found a (locally) optimal solution with a slack norm value larger than zero, there is more information available. Obviously, we have *not* validated the given solution candidate from the decision approach. But in contrast to the second case, we can interpret the infeasibility by taking a closer look at the slack variables that do not vanish in the ValNLP solution. In practice, minimizing the $\ell_1$-norm of the constraint violation typically leads to a small number of nonvanishing slack variables, i.e., the set

$$\mathcal{N} := \{i \in \mathcal{E} \mid \sigma_i^+ > 0 \text{ or } \sigma_i^- > 0\} \cup \{j \in \mathcal{I} \mid \sigma_j^+ > 0\}$$

has small cardinality (see Section 11.6.1). The sets $\mathcal{E}$ and $\mathcal{I}$ denote the index sets of equality constraints and inequality constraints; see Chapter 10. With the indices in $\mathcal{N}$, we can then identify the network elements $u \in V$ or $a \in A$ at which the constraint violations appear. This gives the practitioner a first idea where reasons of infeasibility might be located. However, the slack variable *value* should be used for the infeasibility analysis, too. Unfortunately, the pure value does not give a reasonable tool for the analysis if we do not know the unit in which it should be interpreted. For instance, a slack variable value of 10 is of different importance if it is interpreted in bar or in MW. Finally, the situation becomes even more complicated by the fact that many constraints in the ValNLP model are reformulated in order to obtain better numerical behavior. For instance, the original constraint for the pressure loss at resistors (see 10.1.5),

$$p_u - p_v = \frac{8\zeta_a}{\pi^2 D_a^4} \frac{q_a|q_a|}{\rho_{a,\text{in}}},$$

might be reformulated as

$$\rho_{a,\text{in}}(p_u - p_v) = \frac{8\zeta_a}{\pi^2 D_a^4} q_a|q_a|,$$

to avoid the density variable in the denominator of the right-hand side. A relaxed version might then take the form

$$\rho_{a,\text{in}}(p_u - p_v) - \frac{8\zeta_a}{\pi^2 D_a^4} q_a|q_a| + \sigma^+ - \sigma^- = 0, \tag{11.1}$$

having the unit bar kg/m$^3$ which cannot be interpreted by a practitioner in a direct manner.

These considerations are the reason why we reinterpret the nonvanishing slack variable values in order to allow an infeasibility analysis based upon the ValNLP solution, as follows:

1. definition of a set of physical quantities together with units in which the ValNLP solution should be interpreted, e.g., pressure in bar and flow in kg/s;
2. identification of vertices and arcs whose constraints are not satisfied exactly;
3. reinterpretation of nonvanishing slack variables and reporting the reinterpreted values to the practitioner.

An intuitive way for the reinterpretation is to compute a slack value with the unit chosen in step 1 by rearranging the unrelaxed form of the violated constraint until a term with the appropriate unit is isolated. Relaxation variables are then reapplied and, based on the solution of the ValNLP, values for the relaxation variables are computed. In constraint (11.1), for instance, the recomputed values $\bar{\sigma}^\pm$ of the slack variables satisfy the equation

$$p_u^* - p_v^* - \frac{8\zeta_a}{\pi^2 D_a^4} \frac{q_a^*|q_a^*|}{\rho_{a,\text{in}}^*} + \bar{\sigma}^+ - \bar{\sigma}^- = 0,$$

yielding $\bar{\sigma}^\pm = (\sigma^\pm)^*/\rho_{a,\text{in}}^*$ (where superindices “$*$” denote values of the ValNLP solution). These slack values have the unit bar, which can be directly interpreted by a practitioner. In the following, this approach is referred to as the *direct approach*.

The direct approach has some significant drawbacks. First, it is not always possible to apply it to an arbitrarily chosen quantity in step 1. This is the case for implicitly stated constraints. Second, consider the ValNLP model of pipes. As it is also the case for other network elements, the model involves a *vector* of constraints, $c\colon \mathbb{R}^n \to \mathbb{R}^m$, i.e., $c(x) = (c_i(x))_{i=1}^m$. Thus, it might be the case that there are nonvanishing slack variables $\sigma_j^+$ or $\sigma_j^-$ for different constraints $c_j(x)$, $j \in \mathcal{J} \subset \{1,\dots,m\}$, of the pipe model. If we apply the direct approach, we obtain a reinterpreted slack value for every violated constraint $c_j(x)$, $j \in \mathcal{J}$, and all reinterpreted values can have different units. For the modeler of the problem, this information might be useful since it corresponds to a level of detail that the modeler can handle. However, this is not the case for a practitioner who has no knowledge about the concrete model formulation. Thus, the practitioner is interested in a *single* interpretable value for each network element that has a constraint violation. For instance, the practitioner might ask the question: "What is the error in terms of gas flow at the arc, i.e., how much more or less gas has to flow through it, in order to achieve a feasible solution?" For taking such requirements of practitioners into account, we do not apply the direct approach. Rather, we solve small optimization problems for every network element with nonvanishing slack variables in a post-processing step. In the following, we describe these optimization problems for arcs $a = (u, v)$. The concrete formulations for vertices are completely analogous.

Let $c_{a,\mathcal{E}}(x_a, x_u, x_v)$ and $c_{a,\mathcal{I}}(x_a, x_u, x_v)$ be the component model of the network element $a$ described in Chapter 10. That means, $c_{a,\mathcal{E}}$ and $c_{a,\mathcal{I}}$ are all constraints required to model the arc $a$ and $x_a, x_u$, and $x_v$ are the vectors of variables required by these constraints. Thus, the variable for the quantity selected in step 1 has to be contained in $x_a, x_u$, or $x_v$. Let $\hat{x} \in \mathbb{R}$ denote this single variable. With this notation, the post-processing optimization problem for arc $a$ reads

$$\begin{aligned}
\min_{x_a, x_u, x_v} \quad & |\hat{x} - \hat{x}^*| && (11.2a)\\
\text{s.t.} \quad & c_{a,\mathcal{E}}(x_a, x_u, x_v) = 0, && (11.2b)\\
& c_{a,\mathcal{I}}(x_a, x_u, x_v) \geq 0, && (11.2c)\\
& x_u - x_u^* = 0, \quad \text{except for } \hat{x}, && (11.2d)\\
& x_v - x_v^* = 0, \quad \text{except for } \hat{x}, && (11.2e)\\
& x_a - x_a^* = 0, \quad \text{except for } \hat{x}, && (11.2f)\\
& \hat{x} \in [\underline{\hat{x}}, \overline{\hat{x}}]. && (11.2g)
\end{aligned}$$

Thus, all variables of the arc and its incident nodes are fixed to the values defined by the ValNLP solution, except for the quantity $\hat{x}$ whose deviation to its value in the solution of the validation NLP is minimized. The constraints (11.2d) and (11.2e) fix the tail and head variables $x_u$ and $x_v$, respectively, except for the quantity represented by $\hat{x}$ if it is part of the node variable vectors. In analogy, constraint (11.2f) fixes the arc variables except for $\hat{x}$, if the latter is an arc variable. The objective consists of finding the minimum deviation of the variable $\hat{x}$ with respect to its value $\hat{x}^*$ in the ValNLP solution. For reasons of clarity, let us again take a look at problem (11.2) for the concrete case of $a$ being a pipe and the case in which $\hat{x}$ is chosen to be the flow $Q_{0,a}$ through the pipe. The constraints (11.2b), (11.2c), and simple bounds (11.2g) are the same as for the pipe in the ValNLP model and constraints (11.2d)–(11.2f) fix all variables of the post-processing problem to the values of the ValNLP solution except for the flow variable $Q_{0,a}$ that is part of $x_a$. Thus, if (11.2) is feasible, the value $Q_{0,a}^{**}$ in the solution of the post-processing problem has the minimal distance to the flow value $Q_{0,a}^*$ of the ValNLP solution. The answer to the question *"How much more or less gas has to flow through the pipe in order to achieve a feasible solution?"* is thus $|Q_{0,a}^{**} - Q_{0,a}^*|$.

Unfortunately, the approach of solving problems of type (11.2) in post-processing may fail when the infeasibility cannot be expressed in the quantity $\hat{x}$. For example, this may happen at a compressor group, when the working point of a compressor machine lies above *and* to the right of the feasible operating range in the characteristic diagram. Thus, the required compression ratio as well as the flow through the compressor are too high with respect to the capability of the machine. Obviously, the infeasibility cannot be measured in only *one* of the quantities compression ratio or flow. In such cases, the infeasibility of (11.2) has to be reported to the practitioner with the consequence that the situation has to be analyzed by inspection.

Nonetheless, the described approach turned out to be very useful in practice, since in most of the cases it is possible to give the practitioner an interpretable value of the constraint violations and their location in the network.

## 11.6 ▪ Analyzing infeasibility in a first stage model

In the previous section we have seen how to analyze near-feasible but not provably feasible solutions in the second stage, the ValNLP model. As described at the beginning of Section 11.5, we need to fix the discrete decisions to use this model. In the current section, we deal with the case where none of the first stage approaches can find a feasible discrete-continuous solution to start the ValNLP.

The mixed-integer linear program (MILP) (Chapter 6) and sMINLP (Chapter 7) models are solved by *global* methods: in exact arithmetic and with unlimited computing time, these methods either find a global minimum or they detect that no solution can exist. This offers the advantage of *proving* infeasibility of a nomination with respect to each of the two mixed-integer models—up to the chosen accuracy and if no timeout occurs, of course. In contrast to the RedNLP (Chapter 8) and MPEC (Chapter 9) approaches, which do not provide any information in case of divergence, we may thus obtain a decisive *negative* result.

Moreover, the MILP model is a *relaxation* of a certain intermediate MINLP model so that infeasibility can be proven for this abstract model whose accuracy is close to that of the nonsmooth MINLP (see Figure 11.1). Thus, we can actually detect infeasibility, again of course with some "gray area" that depends on the respective accuracy of the model.

The impossibility to find feasible discrete decisions can originate from different sources: First, the input data might be defective (as mentioned in Section 11.2) due to typing errors, outdated data, or misunderstandings in format. On the other side, there could be modeling and implementation issues. In the validation of a nomination, there might be too much flow to fit through the pipes (a "real," physical infeasibility). During the validation of a booking (see Section 3.2.1), we could encounter an infeasible nomination, but should the booking really be declined because of that? We might ask, "how far away" from feasibility is the nomination? Additionally, there is still the difference between the stationary and dynamic viewpoint (see Section 11.1), and we might be able to "repair" the nomination by pulling an interruptible contract (see Section 3.3.2) which was not incorporated from the beginning. In topology planning (see Section 15.4), we are explicitly asked how one can extend the network to turn a given infeasible nomination into a feasible one—a question for which an understanding of the location of the infeasibility is highly advantageous.

From a practitioner's perspective it is unsatisfactory to obtain a negative result without further explanation: it would be valuable to know why the problem is infeasible or, to pose it more positively, how it could be made feasible with—in some sense—minimal modifications.

There are different approaches how to analyze infeasibilities in models; for an overview in the LP case see Greenberg and Murphy (1991). The authors state

> An *isolation* means to find a portion of the model, preferably of minimal dimensions [sic], that contains the source of infeasibility. A *diagnosis* is an isolation that is meaningful—that is, consistent with the model's syntax, rather than an arbitrary portion that is as difficult to interpret as the original

problem. A diagnosis is *good* if it leads to the correction of the error in a reasonable amount of time.

Following this definition, our goal here is to develop a good diagnosis. Therefore, we present a “tool kit” with different problem-specific methods, which can give usable information not only on the model but the network. First, we could compute the smallest distance to feasibility (SDF) in a suitable but, in principle, arbitrary measure. These methods will include, on the one hand, changes of the network, and on the other hand, modifications of the nomination. The distance is determined by solving a relaxed version of the base model with an MILP solver in a black box fashion. The second approach uses “isolation”: We compute the smallest part of the network which is still infeasible. This is the concept of irreducible infeasible subsystems (IISs). In the following, we will present the different possibilities and practical implementations of these approaches.

As implied above, we have to choose a model on which our analyses can be based, and, since the ability to detect infeasibility should be given, we can use the MILP or the sMINLP. Although it is possible for most of the above-mentioned analyses to transfer the key ideas to other formulations for validation of nominations, we will present our models based on the MILP for ease of description.

## 11.6.1 ▪ Smallest distance to feasibility

For this approach, we ask how we can change as little of the problem as possible and obtain a feasible solution. This is done by introducing slack variables to the chosen validation of nominations model. These slacks will relax different aspects of the formulation, represented by certain constraints, proposing either changes of the network or modifications of the nomination. The aim, and therefore the objective function, is to minimize the deviance from the original model. To demonstrate the principle of the models and to illustrate the flexibility of different validation of nominations models, we will use the following generic model to describe the validation of nominations problem:

$$
\begin{array}{lll}
\min & f(q,p,r) & \\
\text{s.t.} & g(q,p,r) = a, & \text{(pipes)} \\
& h(q) = q^{\text{nom}}, & \text{(flow conservation)} \\
& k_{\text{active}}(q,p,r) \le c_1, & \text{(active elements)} \\
& k_{\text{passive}}(q,p,r) \le c_2, & \text{(further passive elements)} \\
& \underline{q} \le q \le \overline{q}, & \text{(flow)} \\
& \underline{p} \le p \le \overline{p}, & \text{(pressure)} \\
& \underline{r} \le r \le \overline{r}, & \text{(other variables)} \\
& r \in \mathbb{Z}^m \times \mathbb{R}^n, & \text{(partly binary)}
\end{array}
$$

where $f$, $g$, $h$, $k_{\text{active}}$, and $k_{\text{passive}}$ are functions, and $a$, $q^{\text{nom}}$, $c_1$, and $c_2$ are vectors with the associated right-hand sides and of appropriate dimension. As usual, $q$ and $p$ denote the flow and pressure, respectively, while $r$ is a wildcard for every other variable.

Throughout the next sections, we will present some exemplary computational results to give a better understanding of the strengths and weaknesses of the different models. These results are the means of 50 computations with different nominations, respectively, and were computed on an Intel i3 Dual-Core, 3.20 GHz, 8 GB with a time limit of 24 h and an $\ell_1$ objective function, unless otherwise stated. As a solver for the models, we used GUROBI 4.6. The examined nominations have no input data errors, but are "physically" infeasible, due to rather large flow amounts. The network in our numerical experiments corresponds to the H-gas north network (see Figure 1.5). It is medium-scale and consists of 31 sources, 129 sinks, 432 inner nodes, 452 pipes, 6 compressor stations, 23 control valves, 34 valves, and 9 resistors. The topology of the network is the same as that of the test set HN-SN used in Chapter 12.

**Selection of the objective function** As noted above, we are looking for a diagnosis of the infeasibility, ideally with a physical interpretation. For the network changing models, this is given by the concept of an *extension* (see also Section 15.4) of the network. An extension is a new or remodeled network element which helps realize a previously infeasible nomination. Herewith, every nonzero slack value corresponds to a network element which must be altered.

In the different models, the objective function is presented as an arbitrary norm of the slack variable vector $\sigma$. We have implemented and tested models using the $\ell_0$-"norm" ($\ell_0$ is not homogeneous and therefore not a norm by definition) and the $\ell_1$-norm; thus, we minimize the number of nonzero entries and the sum of the absolute values of the entries, respectively. It would be equally possible to use, for example, the $\ell_2$-norm, although the resulting model would no longer be linear. Anyway, in most cases the desired goal should be to minimize the number of affected elements and hence the usage of $\ell_0$. Unfortunately, the $\ell_0$-case is NP-hard to solve; see problem [MP5] in Garey and Johnson (1979), even for a pure LP. Therefore, we also investigated the $\ell_1$-norm, which leads to more easily solvable models and gives—in most cases—a decent alternative. Especially in the nomination modifying models, the $\ell_1$ solution is often very sparse. In a certain sense, the $\ell_1$-norm is also the best convex "relaxation" of the $\ell_0$-"norm": The $\ell_p$-norm, given by

$$||x||_p = \left(\sum |x_i|^p\right)^{1/p},$$

is nonconvex for $0 < p < 1$, but convex for $p \in [1, \infty]$, and it holds that $\lim_{p \searrow 0} ||x||_p = ||x||_0$. For a detailed discussion about the relation between $\ell_0$- and $\ell_1$-norms see Elad (2010). Besides, the $\ell_1$-norm has the additional property of minimizing the value of the slack variables (in contrast to the $\ell_0$-solution) which might be, depending on the application, a desired feature.

Note that (theoretically) the objective value is zero if (and only if) the original model has a feasible solution, since this means every entry in $\sigma$ is zero. Unfortunately though, it might be possible that this is not true in reality: First, in every model we have to cope with numerical difficulties, and second, when changing the bounds for pressure or flow variables, this leads to different linearizations of the pipe modeling in the MILP, since the bounds are used as outer supporting points (see Section 6.2). This problem could also lead to some other undesired effects: Since we work with a given model accuracy, wider bounds mean probably many more supporting points, which in turn leads to much bigger models. So even feasible nominations could become unsolvable or we observe seemingly contradictory solutions. It is therefore advisable to test the feasibility of a nomination first, and not depend on the characterization of the objective function value.

**Flow bounds** The first possibility for a network modification is a relaxation of the given flow bounds:

$$\begin{aligned} \min\quad & \|\sigma^q\| \\ \text{s.t.}\quad & g(q,p,r)=a, \\ & h(q)=q^{\text{nom}}, \\ & k_{\text{active}}(q,p,r)\le c_1, \\ & k_{\text{passive}}(q,p,r)\le c_2, \\ & \underline{q}-\sigma_1^q \le q \le \overline{q}+\sigma_2^q, \\ & \underline{p}\le p\le \overline{p}, \\ & \underline{r}\le r\le \overline{r}, \\ & r\in \mathbb{Z}^m\times\mathbb{R}^n, \\ & \sigma^q\ge 0. \end{aligned}$$

The objective function here is a suitable norm of the slack variable vector $\sigma^q$, where $\sigma^q$ is the concatenation of $\sigma_1^q$ and $\sigma_2^q$.

The physical interpretation of this model is, How much wider do the flow bounds need to be to operate the network? This model is fairly well solvable for our instances; unfortunately it is more or less useless in a real-world application. The explicit flow bounds are provided by rather wide technical capacities of pipes and simply not the limiting factor in a network, and therefore not responsible for the infeasibility, as the computational results emphasize: Out of 50 nominations, 50 models were still infeasible, with a mean solution time of 214.2 seconds. Thus, the only case in which this model might be useful is for detecting wrong input data.

**Pressure bounds** The next model relaxes the pressure bounds in a similar fashion:

$$\begin{aligned} \min\quad & \|\sigma^p\| \\ \text{s.t.}\quad & g(q,p,r)=a, \\ & h(q)=q^{\text{nom}}, \\ & k_{\text{active}}(q,p,r)\le c_1, \\ & k_{\text{passive}}(q,p,r)\le c_2, \\ & \underline{q}\le q\le \overline{q}, \\ & \underline{p}-\sigma_1^p \le p \le \overline{p}+\sigma_2^p, \\ & \underline{r}\le r\le \overline{r}, \\ & r\in \mathbb{Z}^m\times\mathbb{R}^n, \\ & \sigma^p\ge 0. \end{aligned}$$

Here, a "good" solution might be found, meaning that there is actually a pressure bound given for the validation of nominations that is too tight, since the pressure bounds are part of contracts. In this case, the interpretation of the solution would be asking the contracting party to adjust the requested pressure bound or, if the pressure is bounded by technical components, to extend their operating capacity.

In the computational experiments, 36 nominations were infeasible in the above model, 5 reached the time limit (24 hours), and 9 had a solution with a mean of 21.4 bar difference (in sum) to the original bounds and 2.6 affected nodes.

**Pipes** Another possibility is to target the pipes in the network:

$$\begin{aligned}
\min \quad & \|\sigma^\pi\| \\
\text{s.t.} \quad & \tilde{g}(q,p,r,\sigma^\pi) = a, \\
& h(q) = q^{\text{nom}}, \\
& k_{\text{active}}(q,p,r) \le c_1, \\
& k_{\text{active}}(q,p,r) \le c_2, \\
& \underline{q} \le q \le \overline{q}, \\
& \underline{p} \le p \le \overline{p}, \\
& \underline{r} \le r \le \overline{r}, \\
& r \in \mathbb{Z}^m \times \mathbb{R}^n, \\
& \sigma^\pi \ge 0,
\end{aligned}$$

where $\tilde{g}$ is a relaxed pipe modeling, depending on the slack variables $\sigma^\pi$. Here, the actual modeling details are a little more involved since we have some additional equations for the pipe description, as presented below. Preferably, our model will be easy to interpret, so the solution should be transferable to a network extension (e.g., a looped pipe or a different diameter of the pipe). For that reason, we allow the pressure drop on a pipe $a = (u,v)$ to be lower or higher than on the actually built pipe in the pressure-flow situation. Starting with Eq. (2.24),

$$p_u^2 e^{-S} - p_v^2 = \Lambda \phi(q_a) \frac{e^S - 1}{S} e^{-S},$$

we use

$$\begin{aligned}
p_u^2 e^{-S} - p_v^2 + \sigma_+^\pi - \sigma_-^\pi &= \Lambda \phi(q_a) \frac{e^S - 1}{S} e^{-S}, \\
\operatorname{sgn}\left(p_u^2 e^{-S} - p_v^2\right) &= \operatorname{sgn}\left(\Lambda \phi(q_a) \frac{e^S - 1}{S} e^{-S}\right).
\end{aligned}$$

Note the additional constraint, concerned with the flow direction (disregarding possible differences in height): We prevent the flow from going from the lower to the higher pressure node, since this could not be realized just by adding more pipes.

In the computational experiments, the solution of 3 nominations were aborted due to memory issues and 47 were solved to optimality, with an average of 5.9 affected pipes and 72.5 difference in the pressure drop in sum (measured in bar$^2$). The same models but with an $\ell_0$ objective function lead to the following comparison: only 8 nominations could be solved optimally, and the number of changed pipes is minimum 1, maximum 12, and average 4.4. The average computation time was 182.3 minutes (including nonoptimal solutions) and 128.1 minutes for the $\ell_1$ objective function.

**Combination** It is possible for all previously described models that they are still infeasible, with the real-world interpretation that it is just not enough to build new pipes alone, or to widen certain pressure bounds. To overcome this issue, we can combine the models, i.e., introduce slack variables on all mentioned conditions at once. This yields a model which allows corresponding violations at the same time and with directly transferable interpretation as the mentioned ones. Unfortunately, the models would become much harder to solve, so there must be some sort of trade-off. Of course, the size of the network plays a major role. In the computational experiments, all 50 instances were not solvable due to memory issues, although for networks smaller as the one considered here, the combination might be possible.

If the models are combined, one must incorporate weights for the different slack variables in the objective function—on the one hand, to actually weight the preference for the different violations (a pressure bound violation might be cheaper to eliminate than to replace a pipe) and on the other hand to even out the different orders of magnitude of the slack variables: Since, for example, flow and pressure, and thus their bounds and the slack variables, are measured in different units, but are represented in the model by real numbers, there should be some sort of a conversion in the penalty (e.g., a violation of 1 $m^3/s$ is not as bad as a violation of 1 bar).

**Active elements** The last model aimed at an addition of pipes, but an extension of the network could also mean new active elements. Therefore we introduce the following model; the idea here is to allow each nonpipe element (i.e., compressor groups, control valves, resistors, short cuts, and valves) in the network model to work as an ideal compressor or control valve (i.e., change the pressure by an arbitrary amount):

$$\begin{aligned}
\min \quad & \|\sigma^p\| \\
\text{s.t.} \quad & g(q,p,r) = a, \\
& h(q) = q^{\text{nom}}, \\
& k_{\text{active}}(q,p,r) \leq c_1 + \sigma^p, \\
& k_{\text{passive}}(q,p,r) \leq c_2, \\
& \underline{q} \leq q \leq \overline{q}, \\
& \underline{p} \leq p \leq \overline{p}, \\
& \underline{r} \leq r \leq \overline{r}, \\
& r \in \mathbb{Z}^m \times \mathbb{R}^n, \\
& \sigma^p \geq 0.
\end{aligned}$$

Again, this model should serve as a basic concept. The reason to choose the existing elements as a location for these ideal compressors and control valves is that existing elements are cheaper to extend than to build completely new elements. This remodeling is done by adding a slack variable to the particular delimiting pressure relation. For short cuts this is (6.5), for valves Eq. (6.11c) and Eq. (6.11d), for resistors either (6.8) or (6.10). Compressor groups and control valves are handled by relaxing the bypass valve (see Section 5.1.6 and Section 5.1.7, respectively).

For instance, the modeling of a valve $a = (u,v)$ becomes

$$\begin{aligned}
(\overline{p}_v - \underline{p}_u)s_a + p_v - p_u - \sigma_c^p &\leq \overline{p}_v - \underline{p}_u, \\
(\overline{p}_u - \underline{p}_v)s_a + p_u - p_v - \sigma_r^p &\leq \overline{p}_u - \underline{p}_v,
\end{aligned}$$

with slack variables $\sigma_c^p$ for the compressor and $\sigma_r^p$ for the control valve, put in the respective correct position.

In the computational experiments, 25 nominations remained infeasible, 1 reached the time limit with a feasible solution, and 24 were solved optimally with an average of 3.1 ideal compressors and 1.7 ideal control valves.

**Flow amount** So far, we have considered models that would induce changes of the network; we are now switching to nomination modifying models. An application for these could arise after an unsuccessful validation of a booking: As mentioned earlier, we would ask “how infeasible” a nomination is. Thus, the next model tries to find the “closest smaller” feasible nomination with respect to the given one, by allowing a smaller supply and demand at the entries and exits, respectively:

$$\begin{array}{lll}
\min & \|q^{\mathrm{nom}} - \sigma^q\| & \\
\text{s.t.} & g(q,p,r) = a, & \\
& h(q) = \sigma^q, & \\
& k_{\mathrm{active}}(q,p,r) \le c_1, & \\
& k_{\mathrm{passive}}(q,p,r) \le c_2, & \\
& \underline{q} \le q \le \overline{q}, & \\
& \underline{p} \le p \le \overline{p}, & \\
& \underline{r} \le r \le \overline{r}, & \\
& r \in \mathbb{Z}^m \times \mathbb{R}^n, & \\
& 0 \le \sigma^q_u \le q^{\mathrm{nom}}_u & \text{for all } u \in V_+, \\
& q^{\mathrm{nom}}_u \le \sigma^q_u \le 0 & \text{for all } u \in V_-, \\
& \sigma^q_u = 0 & \text{for all } u \in V_0.
\end{array}$$

The objective function value—in relation to the total supplied flow amount—should give a pretty good idea of how to answer our question.

Of the 50 nominations, 48 were solved to optimality, while 2 were aborted due to memory limits. In the minimum case, 0.8% of the original flow amount fed into the network had to be reduced (“almost feasible”), in the maximum case 25.1% (“highly infeasible”), with an average of thereby affected nodes of 9.1.

**Supplier** The studied nominations are potential situations with limits given by contracts (see Section 3.2). These contracts might contain additional agreements which permit the network operator to manipulate the incoming flows. With the next model, we allow the flow at entries to be reallocated to other entries:

$$\begin{array}{lll}
\min & \|\sigma^q - q^{\mathrm{nom}}\| & \\
\text{s.t.} & g(q,p,r) = a, & \\
& h(q) = \sigma^q, & \\
& k_{\mathrm{active}}(q,p,r) \le c_1, & \\
& k_{\mathrm{passive}}(q,p,r) \le c_2, & \\
& \underline{q} \le q \le \overline{q}, & \\
& \underline{p} \le p \le \overline{p}, & \\
& \underline{r} \le r \le \overline{r}, & \\
& r \in \mathbb{Z}^m \times \mathbb{R}^n, & \\
& \sigma^q_u \ge 0 & \text{for all } u \in V_+, \\
& \sigma^q_u = q^{\mathrm{nom}}_u & \text{for all } u \in V_- \cup V_0.
\end{array}$$

In our numerical experiments, 3 nominations were infeasible, 2 reached a limit, and 45 were solved to optimality with an average of 1139.6 relocated flow units (1000 $\mathrm{Nm}^3/\mathrm{h}$) and 19.3 affected nodes.

There are still other models imaginable, built in a similar fashion, which can be used to analyze infeasibility. The presented ones should demonstrate that it is possible to design models tailored to particular needs and interests (e.g., the aforementioned infeasibility in the validation of a booking). So, when deciding which model to use, the first question should be, What will the obtained information be used for? Otherwise, it might be a good idea to compare the solutions from several models. It is possible (and likely) that different models have ambiguous solutions (i.e., indicating other parts of the network as the source of infeasibility), hence, we cannot speak of "the" bottleneck. However, if multiple models highlight the same part, an actual congestion for the nomination was presumably found (see also Section 4.2.3.5).

### 11.6.2 ▪ Infeasible subsystems

The next basic approach is to search for a subsystem as small as possible of the corresponding MILP which is still infeasible. This is the concept of an IIS of the given MILP; see Chinneck (2008) for more details. The computation of IISs for MILPs is done in a straightforward way by removing constraints of the MILPs while maintaining infeasibility, with certain selection rules; see Guieu and Chinneck (1999) for details. This algorithm is implemented in most of the commercial MILP solvers, e.g., GUROBI and CPLEX.

What is left open after the computation is that, in fact, one wants an infeasible subnetwork, not a subsystem of the MILP. Thus, we identify network elements by their describing constraints (where one element can have more than one constraint). In the next step, we build the desired infeasible subnetwork by including every element in the network for which at least one constraint or an associated variable (e.g., the flow variable on a pipe) is part of the IIS. One of the properties of this subnetwork is that it is connected (see Joormann (2013)), hence, this network can be processed further to obtain more detailed information. By doing so, we gain a considerable limitation of the search area for manual search (e.g., for defective data) and, more generally, this method could be combined with the experience of network planners, or the SDF models in Section 11.6.1. The problem is that if the IIS is large, there is almost no informative value.

Although it is not always possible for MILPs in the considered dimensions to compute an *irreducible* infeasible subsystem within reasonable time, one can get a sufficiently good reducible infeasible subsystem (IS) with the same advantages as an IIS. The disadvantage, however, in contrast to the SDF models, is that there is no statement implied on how to resolve the infeasibility, or what the "reason" for the infeasibility is.

For our numerical experiments, we could compute 20 IISs; 30 nominations reached the time limit. The contained constraints translate to a minimum of 7 edges, 8 nodes and a maximum of 468 edges, 446 nodes (compared to 524 edges and 592 nodes in the original network).

This approach is not easily transferred to a basic model other than the MILP: In any model with a nonlinear part, it is nontrivial to compute an IIS, e.g., today's local solvers could report a model to be infeasible, but by adding one constraint, the solver may find a feasible point and the status turns to feasible.

### 11.6.3 ▪ Identifying bottlenecks of the network

The identification of a transport congestion or bottleneck of the network would be highly beneficial for a number of reasons, e.g., for a readily given extension decision (see Section 15.4). Furthermore, bottlenecks are used in the generation of worst-case scenarios (see Section 4.2.3). Unfortunately, mathematically defining a bottleneck of a gas network is not straightforward, let alone computing it.

The standard network approach of computing a min-cut does not work here, as the computational experiments for the flow bound model (Section 11.6.1) emphasize. Besides, the problem of both presented approaches, i.e., SDF models and IISs, is the dependence on the nomination, while we are actually looking for a bottleneck of the *network*. For an arbitrary model, one always needs a right-hand side and that is precisely the nomination. We will now sketch different attempts to break this dependence and we will see that it is not easily overcome.

Intuitively, we could try to use only lower and upper bounds instead of a fixed nomination. The problem with this approach is that for real-world data, the network is not so strict, meaning that, from our experience, we always end up with a feasible model.

In a second attempt, we could perform the presented analyses for multiple nominations and compare the results. But, unfortunately, there are most likely no overlappings of the affected elements, especially due to ambiguous solutions.

Instead of one nomination, we could try to regard several nominations at once (e.g., representatives of temperature classes). The first shortcoming of this idea is that it is not applicable for IISs, and for SDF with nominations $q_1^{\text{nom}}, \dots, q_N^{\text{nom}}$, there would arise a system

$$\begin{aligned}
g(q_i, p_i, r_i) &= a && \text{for all } i = 1, \dots, N,\\
h(q_i) &= q_i^{\text{nom}} && \text{for all } i = 1, \dots, N,\\
k_{\text{active}}(q_i, p_i, r_i) &\le c_1 && \text{for all } i = 1, \dots, N,\\
k_{\text{passive}}(q_i, p_i, r_i) &\le c_2 && \text{for all } i = 1, \dots, N,\\
\underline{q} \le q_i &\le \overline{q} && \text{for all } i = 1, \dots, N,\\
\underline{p} \le p_i &\le \overline{p} && \text{for all } i = 1, \dots, N,\\
\underline{r} \le r_i &\le \overline{r} && \text{for all } i = 1, \dots, N,\\
r_i &\in \mathbb{Z}^m \times \mathbb{R}^n && \text{for all } i = 1, \dots, N,
\end{aligned}$$

with copies of the constraints for every nomination (note that $q_i$, $p_i$, and $r_i$ are still vectors). This is problematic because there already is a little luck involved if the elementary models can be solved, let alone a model $N$ times the size.

Instead, it may be worth looking at the following empirical approach. We start with a given set of nominations (feasible and infeasible) and ask how their status changes if the network elements are modified successively (scaling up or down the pressure interval at a fixed node, changing a pipe diameter, etc.). Then, we count how many infeasible nominations became feasible and vice versa. After such computations we could state that if a smaller (pressure, diameter, etc.) value does not lead to many more infeasible nominations, then the examined element is *not* a bottleneck. Similarly, if a larger value does not lead to many more feasible nominations, the element is *not* a bottleneck (at least not alone). Finally, if a larger value does lead to many more feasible nominations, the corresponding element *is* a bottleneck.

For this last approach a lot of individual computations need to be performed. However, since these are only ordinary validations of a nomination, they can be carried out much faster than SDF computations (see Section 11.6.1 as an indication for SDF computational times and Section 12.2 for validation of nomination computations). The computational effort can be justified by the independence of only a single nomination, although it should be clear that some sort of a preselection of the regarded elements is necessary; for a real-world network it is not possible to observe every element individually (nor every combination of these).