Energy Conversion and Management: X 24 (2024) 100690

Contents lists available at ScienceDirect

Energy Conversion and Management: X

journal homepage: www.sciencedirect.com/journal/energy-conversion-and-management-x

An object-oriented implementation of a recursive "quantum network" solver and its application to district heating networks

Cornelia Blanke *, Malick Kane

School of Engineering and Architecture (HEIA-FR), Energy Institute, HES-SO University of Applied Sciences and Arts of Western Switzerland, Boulevard de Pérolles 80, CH-1700 Fribourg, Switzerland

ARTICLE INFO

Keywords:
District heating
Thermal network
Energy system modeling
Thermo-hydraulic simulation
Object-oriented design
Recursive solver

ABSTRACT

With the increasing importance of energy efficiency and sustainability, the demand for high-performance district heating networks is also on the rise. As traditional engineering methods were often no longer sufficient, several packages for numerical simulation have evolved. Many of them offer high accuracy at the cost of considerable manual preparation and computational effort. However, there is still no user-friendly software that is equally suitable for simplified studies in the early project phase, for the rapid optimisation of multiple concepts and for subsequent detailed planning and dimensioning. For this reason and in cooperation with some companies from the energy sector, we had conceived a novel, highly flexible modular approach, the so-called "quantum networks", where all parts of the district heating network are appropriately abstracted into quantum elements. Now we present our recent implementation of this model in an object-oriented C++ library. Starting from a generalised base class and making use of the concepts of inheritance and polymorphism, the idea of different levels of detail for the same element type is directly realised and can always be further refined. In addition, as all elements are derived from the same base class, they all share the same outer appearance and can thus be easily combined or interchanged. Based on these prerequisites, a dedicated thermo-hydraulic solver has then been developed. Thanks to its recursive design requiring neither outer iterations nor matrix inversions, it proves to be extremely fast and is therefore suitable for rapid design or optimisation studies in which a vast number of configurations has to be computed. To conclude this part of the work, the developed C++ library was benchmarked on two use cases covering a full year of operation that can now be computed in approximately one second of runtime.

1. Introduction

In many buildings around the world, their inhabitants' need for heat and/or sanitary hot water is fulfilled by different kinds of local heat generators (e. g. ovens, electrical heaters, local heat pumps). Those systems do not only take valuable space in the buildings, but often they are also not the most cost and/or energy efficient way of heat generation. In contrast to those local installations, in a district heating network (DHN) a big amount of heat is generated by a central power plant and then distributed through a piping system to a multitude of users.

This fundamental idea of DHNs is quite old. Already the Romans were using hot thermal water not only for their famous baths but also to heat their buildings. Later on, similar methods were likewise adopted in different regions of the world having access to hot springs (e. g. Iceland or Chaudes-Aigues, France). The first extended commercial DHNs of the modern age were built in the United States around 150 years ago and later replicated in European cities. In contrast to the use of naturally hot water, they relied on coal-fired steam distributed in concrete ducts [1].

The idea of this first generation of DHNs was optimised by the use of steel pipes and replacing the steam by pressurized water still having a temperature above 100 °C (second generation, around 1930–1970). The third generation that is nowadays widely implemented still works similarly, but thanks to progress in the thermal efficiency of buildings, water at lower temperatures between 70–95 °C can be used. Thereby various ways of heating up the water are possible: fossil fuels like coal, oil and gas, but also biomass and waste incineration, and occasionally geothermal heat.

The newest generations of DHNs are currently under development with some first realisations. Often the circulating fluid is still water but at low (fourth generation, 30–70 °C [2]) or very low (fifth generation, 5–30 °C [3]) temperatures. Hereby the power mostly comes from waste heat of industries or from renewable sources (biomass, geothermal

* Corresponding author.
E-mail address: cornelia.blanke@hefr.ch (C. Blanke).

https://doi.org/10.1016/j.ecmx.2024.100690
Received 23 May 2024; Received in revised form 31 July 2024; Accepted 13 August 2024
Available online 14 August 2024

energy, lake water). If necessary, these low temperatures are augmented by centralized or decentralized heat pumps according to the customers' needs. Thus, the DHNs maximise the share of variable renewable energy, minimise heat loss in the piping system, and create a link between thermal and electrical networks.

With slight modifications the fifth generation DHNs can also be used for cooling purposes what is a growing need because of the increasing amount of IT infrastructure but also due to global warming.

1.1. Simulation of district heating networks

The growing complexity when designing new DHNs is tackled with modern means of computer-aided engineering. While it is nowadays common standard to use a variety of software packages for technical and geographical drawings or for business operations, the use of thermal-fluid simulation software is not yet so widespread in the corporate world.

Among researchers, on the other hand, the modelling and simulation of district heating networks has strongly evolved in the recent years. Several big review articles have been published citing hundreds of references. Allegrini et al. [4] focus on the district level, but include thermal networks. In a tabular overview, they list 24 distinct software packages and evaluate their usability to 17 fields of application. Therefrom they identify 17 software packages to be at least basically suited for thermal networks. Sarbu et al. [5] summarize physical and numerical models commonly used in DHN modelling, mention several software packages, before they come to a survey of optimisation approaches. Brown et al. [6] cover the modelling of the energy source, the end users, the distribution network, and thermal storage. They distinguish between physics-based and data-based models, between bottom-up and top-down approaches, and also provide a tabular overview of software packages. The review of Kuntuarova et al. [7] is based on a systematic evaluation of the Scopus database in order to find long-term trends. They state that traditional node-based models are still widely implemented, but that newer plug flow methods are increasingly used, as those are able to describe dynamical aspects at a lower computational cost than the traditional solvers.

To summarize, we can split the modelling approaches and tools into different groups:

Highly specialised software: Three-dimensional computational fluid dynamics (CFD) solvers like OpenFOAM, CFX, STAR-CCM+ are appropriate for detailed studies of single flow components but are far too complex for the time-dependent analysis of a full DHN. In the latter case, software packages specialised in 0D or 1D pipe network flows that make use of approximative correlations seem to be a better choice. Among them are commercial tools like Fluidit, NetSim, IQGeo Comsof or NEPLAN that provide modules specifically aimed at DHN operators. One of their big advantages with regard to practical applications is their integration of geographic information systems (GIS) that are needed for reading existing data and for depicting the DHN to scale on a city map. Furthermore, they often provide options to run coupled simulations of thermal/electrical grids or to include a thermo-economic analysis. The feedback from our industrial project partners was that those packages have brought real added value to their traditional design processes, that they enjoy working with these tools, but that it is rather tedious and time-consuming to run simplified studies of early concepts because the simulations take time and too many details need to be modelled.

General-purpose tools: A more challenging, but also more flexible way of modelling a DHN can be implemented with TRNSYS, MATLAB-Simulink or with Modelica-based software like Dymola or Simcenter Amesim [5]. Here the considered level of detail can be freely set by the user. Those tools provide ready-to-use flow components like pipes or valves and can be arbitrarily extended using own programme code. The connections in between the components are represented by a flow chart but in contrast to the aforementioned tools they do not show the real geographical scale.

In their study, Schweiger et al. [8] compare four tools of this group and benchmark them on their capacity to capture the transient thermal dynamics of a piping system. They distinguish between the causal modelling paradigm (used in TRNSYS, Simulink) that requires the direction of the flow of information known a priori, and the acausal one (used in Modelica) that describes the whole system by a set of differential-algebraic equations. In accordance with [7] they found out that the acausal approach is of growing interest due to its higher flexibility not only for DHNs but in many fields of application, and that it is superior when it comes to the mathematical formulation of optimisation problems. On the other hand, Casella [9] pointed out in 2015 that the then current version of Modelica suffered from multiple performance and stability issues when very large models need to be calculated. Hirsch and Nicolai [10] provide a concrete example written in C++ that is not only 75 times faster, but also more accurate than its equivalent in Modelica.

Coding languages: Thus said, the third option is always to start from scratch and build a new tailored software using either the MATLAB language or a high-level programming language like C++ or Python. This option provides the highest flexibility and often the highest computational speed but at the cost of a large manual effort. That is why it is favourable and widely used in a research context where new approaches and tools are to be developed, but in a corporate environment it only makes sense if all previous approaches were inadequate and a positive "return-of-invest" is expected.

Within this context, several open-source Python libraries have been developed in the past five years for the simulation of district heating networks. DHNx [11,12] is part of the bigger library oemof (Open Energy Modelling Framework) around energy modelling and is not only able to perform thermo-hydraulic simulations, but also to trace an optimised distribution system and to estimate its cost. DiGriPy [13] is also built upon a package included in oemof, namely TESPy [14], and focuses on heat losses of buried pipes. Just like DHNx, PyDHN [15,16] makes use of the Python graph library Networkx and implements among others the novel Lagrangian and lump-mass based approach of Dénarié et al. [17] covering the transient thermal dynamics of the pipes. GridPenguin [18] uses a plug-flow method and is aimed to be coupled with machine learning approaches.

While lots of mature software applications are written in C++, many DHN researchers seem to avoid working with this language said to be more complicated to learn. A quick keyword search on the titles/abstracts/keywords in the Scopus database underlines this impression and gives an insight of the preferred DHN modelling approaches among academic researchers (Table 1). Note that many more authors were actually using such tools but did not mention them in their abstract, nor can that spontaneous evaluation be considered as a valid proof. Nevertheless the result is interesting in several respects. The keywork "district heating" was found 13,324 times with the oldest article dating from 1928, but 55 % of the database entries were published in the last ten years. When combining "district heating" with each of the tools or languages, we discover that TRNSYS, MATLAB and Modelica are almost equally dominant over the high-level programming languages and that 90 % of these objects were published in the last ten years (whereby the spread of Fortran is declining, as expected). Only one hit was found for C++, and this conference paper from 2019 has only little to do with the

Table 1
Number of Entries in Scopus (accessed 23 July 2024).

keywords	total	2014–2024
district heating	13,324	7325
+ TRNSYS	145	130
+ MATLAB	117	103
+ Modelica	100	96
+ Python	32	32
+ Fortran	7	3
+ C++	1	1

modelling of DHNs.

1.2. Differentiation from other modelling approaches

When talking about DHN modelling here, we restrict ourselves to the primary circuit between one (or more) thermal power plant(s) and the substations where the thermal power is transferred to the secondary circuit. The primary circuit includes the piping system with its various branches as well as centralized or decentralized energy sources (central plant, waste heat, geothermal, heat pumps etc.) and sinks (handover to users, heat losses).

Some researchers in mechanical or process engineering deal with the details of single components. They work for example on the improvement of pumps, turbines, burners, heat exchangers, heat pumps, just to name a few. In their work they rely on predefined boundary conditions that map the typical situation of installation.

Other researchers focus mainly on urban districts, buildings and/or users on the secondary side. They investigate the energy efficiency of standalone buildings, the interaction of multiple buildings in the context of residential areas [4], or they study users' behaviour and expectations [5]. Here we can distinguish between physics-based models and data-based models which may be on a single-user scale or an aggregation of multiple users or buildings [6]. In both cases, for the appropriate scale, the outcome of these models yields a boundary condition for the simulation of the primary network.

Modelling on a regional, national or international scale is a relevant domain as well [6] and involves advanced methods of clustering [19,20], especially when it comes to analysing the interaction between multiple networks (water, thermal, electrical), to predicting future needs and costs, or to assessing the impact on the ecosystem.

1.3. Objectives of this work

In [21], Kane and Rolle introduce the concept of quantum networks and demonstrate the procedure for converting an arbitrary DHN into such a network. A major strength of this definition is the reduction of all components present in a real DHN to a handful of quantum element types obeying a standardised numbering convention. This results in a high flexibility regarding the modelled level of detail, as it is equally possible to represent singular elements as well as clustered aggregates.

The present work discusses the subsequent step, namely an object-oriented C++ implementation for the modelling and computation of those quantum networks. Using the relevant principles of object-oriented programming, a hierarchical library for all individual quantum network elements will be introduced. Next, a C++ class representing the overall quantum network will be established. In addition to data handling, it will primarily contain a recursive solver for mass flow and temperature. Finally the implementation will be applied to two use cases.

Thanks to the hierarchical design of the C++ library with a very basic base class and multiple derived specialisations representing the quantum network elements, a high modularity is reached. Whatever will come, it will always be possible to build a quantum network model out of rather basic elements or to create new specialisations containing more and more details. As the classes corresponding to quantum network elements are always derived from the same base class, they all have a common appearance and interface to the outside world.

In addition, the highly standardised definition of the topology of quantum networks makes it possible to develop a single solver that fits them all. As quantum networks do not contain any loops in their branches by construction, a direct method without any outer iterations will be used. (If needed, there may be iterations inside quantum elements though.) For this reason, even for extended DHNs a very low computational time is reached, making it possible to calculate a large number of quasi-steady time-steps and/or to compare a huge number of variants in an optimisation study.

2. Quantum networks

The idea of modelling DHNs as "quantum networks" has been brought up by Kane [21] in the past years. A preliminary implementation in MATLAB has been realised by Jérémy Rolle, applied by Pierre Barre and extended by Simon Rime [22] as major parts of their master theses, whereas the objective of the present article is to present an object-oriented implementation in C++ that, thanks to its modularity, can always be further extended.

The quantum networks theory assumes that every DHN can be topologically transcribed as a binary tree with the central plant being the head of the tree and the substations being the leaves. All pipes of the DHN always run in an antiparallel manner with one supply and one return path. The customers connected to the substations are not part of the study, but are considered as boundary conditions to the DHN (see section 4.1.). In the frequent case that the studied DHN does not directly fulfil that requirement (i. e. having more than one central plant and/or including loops in the branches), it has to be split into sub-models and/or simplified by clustering. For example, if a branch contains a loop as depicted in the first sketch shown in Fig. 1, all substations on the loop can be glued together into a cluster in order to approximate an overall solution. Only if more details along the loop are needed, an iterative approach must be applied on that sub-model of reduced size. The second sketch of Fig. 1 can be seen as an interconnection of multiple DHNs of branched topology. Each one can be studied individually with an estimated supply temperature. In a second step, the interconnection can be analysed without entering into the details of each branch. However, the third case of Fig. 1 showing a strongly meshed topology causes serious difficulties. If additional knowledge of the DHN operation is available, it may be possible to eliminate some segments from the computational model. Such a method is proposed by Guelpa and Verda [23] who replace the deleted segments by additional mass flow boundary conditions. Otherwise, the inner details of such a DHN cannot be covered with the library presented in this work, but the DHN can still be modelled as a whole.

Based on that prerequisite, the quantum networks theory [21] introduces a consistent naming and numbering convention for the elementary components of the network that was inspired by quantum mechanics. The whole DHN is considered to be an atom with the central plant being the centre and various elements positioned around. The atomic core corresponds to the branches of the main distribution network: as outstanding elements, the bifurcations that divide one branch into two are called "neutrons", and the tees where the substations are connected to the big branches are called "protons". The substations themselves are not part of that atomic core, but are placed around it and are thus called "electrons". Note that thereby the electrons and protons always come in pairs and that only the number of neutrons may change when different versions of a future DHN are studied. By analogy with quantum mechanics, these versions are called isotopes of the same atom. The piping system acts as the glue whereby the section between two protons and/or neutrons is called "segment" and is of constant diameter, and everything that lies between two bifurcations is called "branch".

Then, all neutrons are associated to a quantum number $n = 1, 2, 3\ldots$ from top (central plant) to down, and the two branches emerging from each neutron are numbered as $l = +n$ and $l = -n$. Last, like in quantum mechanics, the segments, protons and electrons on each branch are numbered from top to down as $k = s, p, d\ldots$ (or $k = 1, 2, 3\ldots$). An example showing this numbering convention is depicted in Fig. 2. Now, if every substation has a unique identifier (a name or a number), the full information about the DHN topology can easily be stored in the form of a table: all substation identifiers are written down in the cell corresponding to their respective l and k value, whereas in the last line of the table the quantum numbers of the neutrons are positioned under their feeding branch. With the help of that compact table, a computer code can always reconstruct the connectivities in the DHN without the need

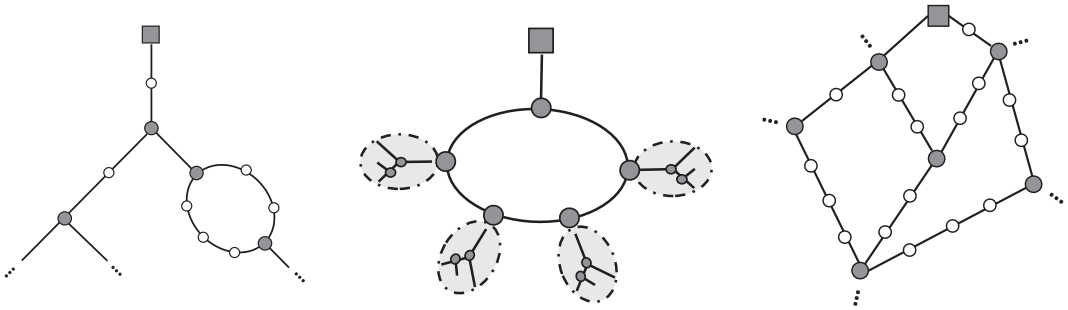
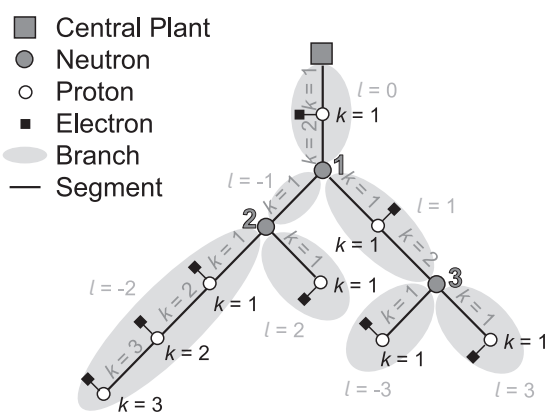


Fig. 1. Examples of DHN topologies that cannot be directly described as binary trees.



l		-3	-2	-1	0	1	2	3
k	1	x	x		x	x	x	x
	2		x					
	3		x					
n				2	1	3		

Fig. 2. Example of a Quantum Network, its Numbering and its Representation as Table (with x standing for the identifiers of the substations).

of additional lookup tables or expensive search algorithms.

Note that this numbering convention mimics the fact that network elements with similar quantum numbers are at similar energetic levels. Of course, these levels are not strictly discretised as in quantum mechanics, but nevertheless it can be stated that the branches with the lowest l values are the most important and carry the most load, that branches $l = +n$ and $l = -n$ are on the same energetic level as neutron n but turn into different directions, and that the k values define the sublayer structure inside each branch whereby segments with higher k values are less charged.

In addition, Kane [24] also introduces several physics-based quantum numbers. For example, the $\overline{\mu}$ value of each element is defined as the scaled local mass flow divided by the number m of substations that are alimented through that element

$$\overline{\mu}(t) = \frac{1}{m}\frac{\dot{m}(t)}{\dot{M}/Z}$$

(with $\dot{M}$ the nominal mass flow of the central plant and Z the atomic number, i. e. the total number of substations in the DHN). In a perfectly balanced DHN, all elements would have the same $\overline{\mu} \leq 1$, but in a real DHN, the $\overline{\mu}$ value can serve as a measure of imperfection. Further physics-based quantum numbers concern temperatures, energies or global characteristics of the DHN. These quantum numbers could be useful for simplified approximations or may serve as an unbiased method when it comes to judging the performance of different DHNs against each other and are thus evaluated as an optional postprocessing step in the here-presented C++ library.

3. Object-oriented class structure

The idea is now to implement and solve such quantum networks on a computer. As programming language C++ was chosen [25,26]. One reason was simply the fact that there was already some reusable code from previous projects at the institute, but the key advantage was its high speed at runtime as Bansal explains in his introductory textbook [27]. Indeed, as C++ is a compiled language, an executable programme is created only once in advance whereas with interpreted languages such as Python, the code is always read and interpreted during runtime. If we talk about a single small case, the runtime of a slower language might still be acceptable but as soon as the solver will be used for optimisation problems there will be huge amounts of variants to be run.

As many modern programming languages, C++ is an object-oriented language. Hence the code can be organized into libraries and classes. A library refers to a collection of classes, and a class is a structure that encompasses its characteristic variables and methods. New classes can be defined on their own or can be derived as specialisations of already existing classes. In the latter case, the class inherits all the member variables and methods from the base class and implements additional specialties. Thereby the new class might also redefine methods inherited from the base class, if needed.

Finally, those classes can be imported as reusable code into other classes or into a main() programme in order to create instances of the respective class type.

3.1. Base class "Quantum"

This concept of a base class and derived classes will now be used for quantum network elements (Fig. 4). The base class will simply be called "Quantum" and will contain all properties and methods needed for all the derived classes. Later on, the derived classes will be defined inheriting these properties and methods, redefining the methods if necessary, and adding further variables and methods specific for the respective element.

As member variables the base class "Quantum" includes physical properties common to all network elements like mass flow, temperature

differential between supply and return flux, pressure loss and heat loss, etc. It also comprises the various quantum numbers defined by Kane [21,24] as well as different types of cost (investment cost, operating cost). Furthermore, there is a pointer to a class "Flow" containing the fluid definition and the thermophysical properties of the flow. Typically, the flow is different for each network element which is why those pointers will usually point to different instances of "Flow". In contrast, the "Tambient" pointer will inform all elements about the current outside temperature.

The last member variable is a vector of pointers to "Quantum" instances called "next". This idea can be seen as a generalization of the principle of linked lists [26,27]. In a linked list (see Fig. 3), each element holds a pointer to the next element in addition to its own variables and methods. Thus, a chain of elements can be built up. Here in the case of quantum networks, the situation is a bit more complicated as the elements of the network may have either zero or one or two successors. That is why the common "next" pointer was replaced by a vector of pointers that could be of arbitrary length ≥ 0 (Fig. 5).

In addition to those member variables, the class "Quantum" encompasses a bunch of compute methods, mainly for the physical properties and the quantum numbers. Many of them are just empty placeholders and will be actually implemented in the derived classes. Some others hold formulas that are equally valid for all network elements. And few of them contain a common base that will later be overridden in part of the inherited classes. Other methods deal with the initialization of the aforementioned pointers, i. e. the "Flow" pointer, the "Tambient" pointer and the vector of pointers "next" that makes use of the rigorous definition of the quantum network.

3.2. Inherited classes (Specialisation)

All elements of a quantum network are derived classes from the base class "Quantum". They are called "TCentralPlant", "TElectron" (with another derived specialisation "TElectronHP"), "TProton", "TNeutron", "TSegment" and "TBranch". The class "TCluster" and its potential specialisations will reside here as well, but is not yet implemented. Only the class "QNetwork" that represents the overall network is fundamentally different and will be described later in section 3.3. below.

All of the first-mentioned classes include as member variables their characteristic quantum numbers and a pointer to an instance of the overall network "QNetwork".

In an instance of the class "TCentralPlant", one or more boilers may be defined. This allows the estimation of the maximum power of the central plant, the investment costs and the costs of combustibles.

The class "TSegment" holds a pointer to a hydraulic pipe class "HPipe" that enables the use of characteristic variables and methods of hydraulic pipes (e. g. length, diameter, Reynolds number, pressure and heat loss etc.). Thereby it is assumed that both supply and return pipes of the segment are identical. If this is not the case, a specialisation of the class can still be derived. Currently the class "TSegment" only deals with straight pipes but other hydraulic elements like elbows or valves can likewise be included.

A "TBranch" is simply the collection of all protons and segments on a branch.

The class "TElectron" represents the substations of the network that are the handover points of the heating power to the customers. The member variable of type "District" informs about the time-dependent power demand of the connected district (see section 4.1.). In the simplest case, the power transfer from the network to the customers' district is a one-to-one transfer without losses. The specialisation "TElectronHP" adds a heat pump to the substation that augments the network's power. This difference between the two is realised in the method compute_power_from_district() that in the first case simply equals the time-dependent powers

$$P_{\text{network}}(t) = P_{\text{district}}(t) \tag{1}$$

and that is overridden for the second case with the relation

$$P_{\text{network}}(t) = (1 - 1/COP(t)) \cdot P_{\text{district}}(t), \tag{2}$$

using the definition of the (time-dependent) coefficient of performance COP of the heat pump:

$$COP(t) = \frac{P_{\text{district}}(t)}{P_{\text{electric}}(t)} = \frac{P_{\text{district}}(t)}{P_{\text{district}}(t) - P_{\text{network}}(t)}$$

3.3. "QNetwork" class (Input/output, overall control, solvers)

As said, the "QNetwork" class is basically different. It takes over the global control of the quantum network's calculation. First, it is in charge of reading and processing the text files containing all input data. Afterwards, it creates and initializes the elements of the network, and organizes their memory addresses in pointer arrays. Therefrom it is straightforward to determine the global quantum numbers N (number of neutrons), Z (number of protons) and the mass number $A = N + Z$, and to implement algorithms that loop over all elements of a certain type (e. g. loop over all segments).

Then the "QNetwork" class also implements the different parts of the solver that will be described in more detail in section 4.3.–4.5., and the output of the results into text files.

4. Boundary conditions and solver

The solver that is described in this work is a quasi-steady solver. That means that it iterates over several time-steps, but for every time-step it assumes a steady-state condition. All transient dynamics in the district heating network are neglected. As a consequence, it is clear that sudden hydraulic shocks and pulses cannot be covered, and consequently the final solution will be smoothed out. However, as long as the question is not about investigating transient dynamics, and as long as the input data does not cover these effects either, this seems to be a valid approximation.

For the thermal solution, the solver postulates a steady-state condition for each time-step, too. As in reality all components have a thermal inertia, and as the total travelling time for the fluid through the network is often bigger than the time-step size of the simulation and may even become extremely large if the DHN is operated at very low load, temporal lags and smearing effects in the solution must be expected. The importance of these effects was studied in more detail by Guelpa [28]. For a DHN that is completely cooling down during the nights, she found out that the morning peak is reduced by up to 20 % if thermal transients are neglected, but that the influence is much lower when the network is operated in a more or less continuous mode.

4.1. Boundary conditions for substations

When introducing in section 3.2. the class "TElectron" representing the substations, we already mentioned the link between customers' need of power and the amount of power supplied from the network.

One output of the "District" class (implemented by Lucile Schulthess)

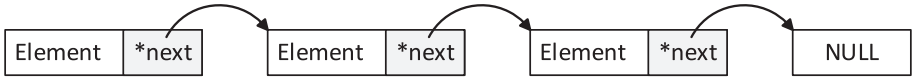


Fig. 3. Principle of a Linked List.

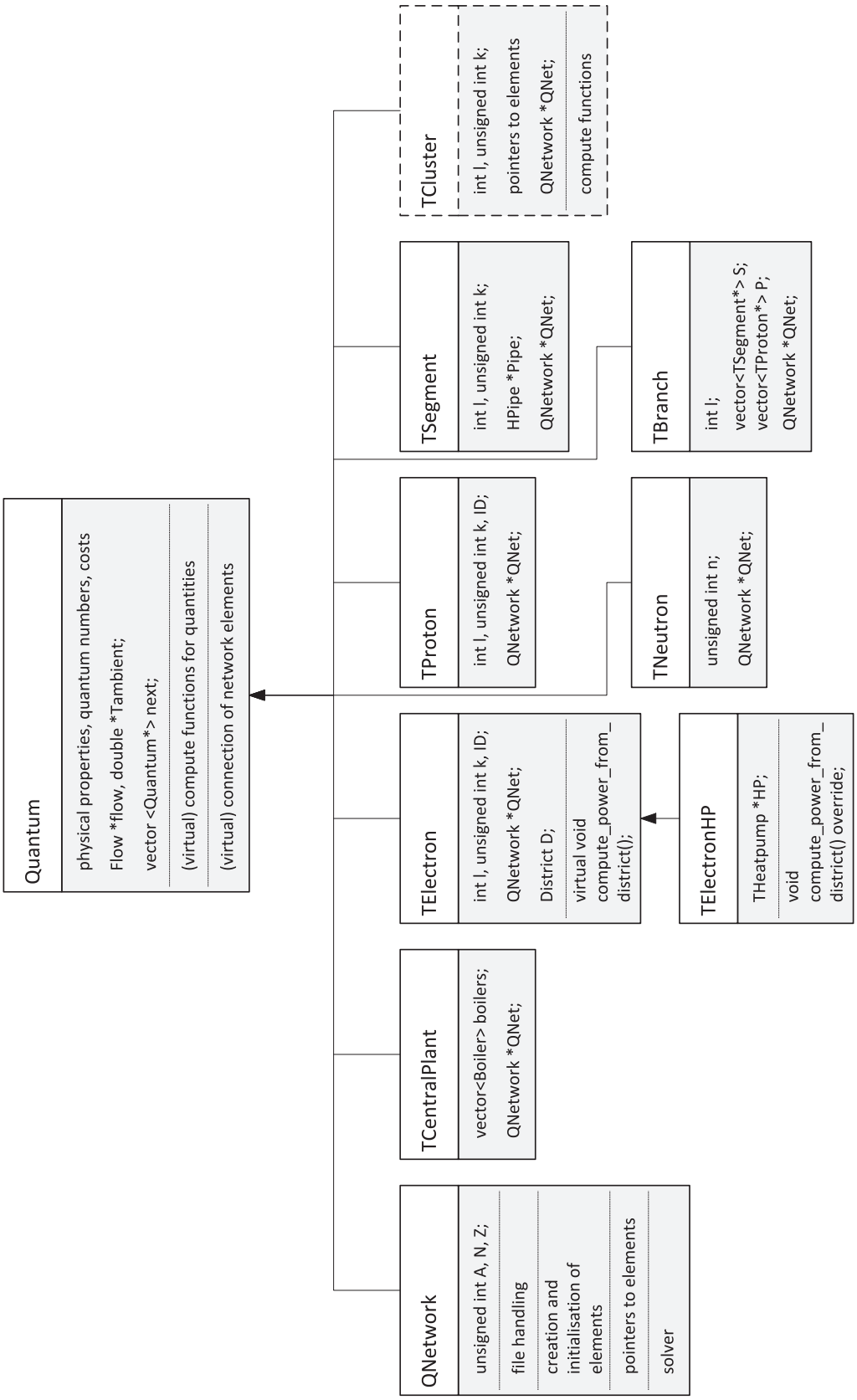


Fig. 4. C++ Class Diagram of Quantum Network Classes.

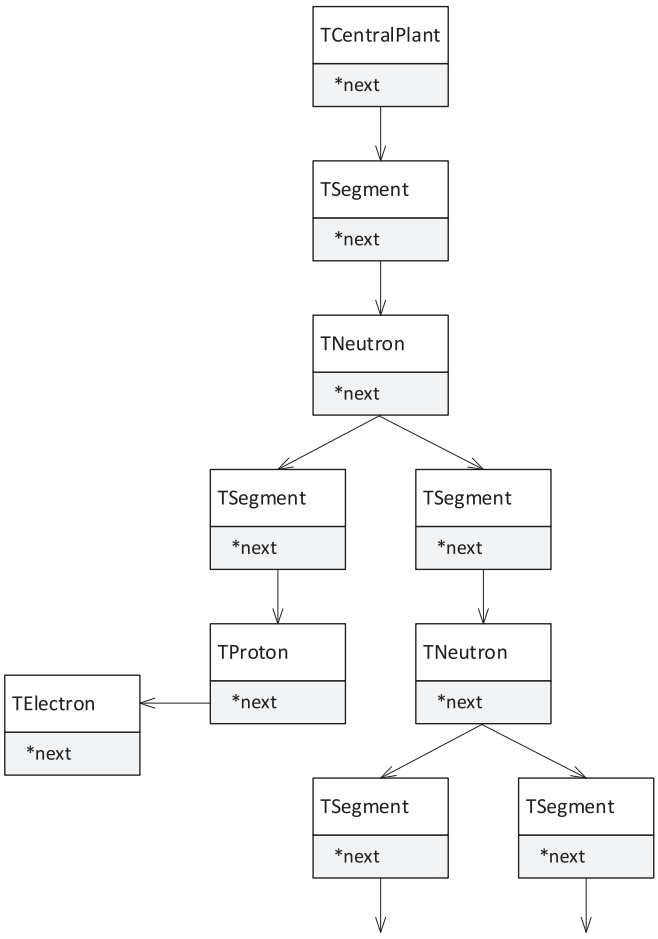


Fig. 5. Principle of Linked Lists applied to Quantum Networks.

is a time-dependent profile of the power demand of that particular district. There are several ways to get such a profile. A good way would be to use measurement data but those are often not available with the required level of detail. In contrast, the energy demand for a full year E_{annual} is often known from previous years or can be estimated by the help of the characteristics of the buildings forming the district (e. g. building type, year of construction, surface area) and the SIA norm [29]. The SIA norm [29] also informs about the relation between the energy needed for heating and the one for sanitary hot water. For apartments blocks this is for example 93:16 whereas for indoor swimming pools it is 102:94. The annual energy demand can thus be split into two parts $E_{\text{heating}} = (1-\alpha)E_{\text{annual}}$ and $E_{\text{water}} = \alpha E_{\text{annual}}$. In the case that a customer only asks for one of these two types of energy, of course this split will be omitted.

Then, in this work and following again the SIA norm [29], the time-dependent power demand for sanitary hot water is assumed to be the same during every day of the year. No difference is made between workdays and weekends, or between different types of end users. However, a distinction is made between different types of buildings (see Fig. 6).

The time-dependent power demand for heating is approximated by

$$P_{\text{heating}}(t) = \begin{cases} C_{\text{district}} \cdot \left(T_{\text{ref}} - T_{\text{ambient}}(t)\right), & T_{\text{ambient}} < T_{\text{ref}} \\ 0, & T_{\text{ambient}} \geq T_{\text{ref}} \end{cases} \tag{3}$$

where the proportionality constant C_{district} is determined so that the total energy needed for heating during the year equals E_{heating}. The reference temperature T_{ref} represents the threshold where heating is switched on and might be different for each building. If the ambient temperature is above this threshold, no heating is needed, if it is below, a linear dependency of the temperature difference is assumed.

There are currently many projects underway investigating the customer needs in more detail [4,6]. So in the near future, it will be possible to refine the described models.

In any case, as soon as we know the time-dependent power demand of the districts, the power supply requested from the network can be computed from equation (1) or (2) respectively. Finally, for a predefined temperature differential dT of the substation and a known specific heat capacity of the circulating fluid (usually water), this yields a boundary condition for the mass flow in the substation:

$$\dot{m}(t) = \frac{P_{\text{network}}(t)}{c_p\, dT} \tag{4}$$

4.2. Other boundary conditions

As we want to analyse not only the hydraulics, but also the thermal and energetic situation of the DHN, some additional boundary conditions need to be specified.

4.2.1. Central plant

If we assume that the central plant of the network is dimensioned as “sufficiently strong”, the only boundary condition that must be indicated is the inflow temperature into the network. Later on, the simulation model might be enhanced by adding restrictions concerning the range of possible mass flow and/or possible power supply which will limit the operation of the network.

Hour of Day

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
apartment block	16	16	16	16	16	33	131	33	16	16	16	16	131	33	16	16	16	33	131	164	33	33	33	16
individual housing	16	16	16	16	16	33	131	33	16	16	16	16	131	33	16	16	16	33	131	164	33	33	33	16
administration	12	12	12	12	12	12	12	24	72	96	120	96	48	72	120	96	72	24	12	12	12	12	12	12
school	11	11	11	11	11	11	11	46	69	92	115	92	23	69	115	92	92	46	11	11	11	11	11	11
commerce	13	13	13	13	13	13	13	66	66	66	66	66	66	66	66	66	66	66	66	66	13	13	13	13
restaurant	12	12	12	12	12	12	12	12	47	47	47	70	116	116	12	12	12	12	47	47	70	116	93	47
gathering place	9	9	9	9	9	9	9	18	54	89	89	18	18	89	89	54	54	54	54	89	89	54	18	9
hospital	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	133	100	67	33	33
industry	20	20	20	20	20	20	20	20	61	102	102	82	41	61	102	82	61	20	20	20	20	20	20	20
stock	20	20	20	20	20	20	20	20	61	102	102	82	41	61	102	82	61	20	20	20	20	20	20	20
sport	8	8	8	8	8	8	8	40	65	81	81	65	0	65	81	81	65	40	65	65	65	40	40	8
indoor swimming	8	8	8	8	8	8	8	40	65	81	81	65	0	65	81	81	65	40	65	65	65	40	40	8

Fig. 6. Time-dependency of Sanitary Hot Water Demand (in per mill of daily need).

4.2.2. Pipe walls

The fluid flowing through the pipes of the network suffers from a significant pressure and heat loss. As long as we consider only straight pipes, the pressure loss can be evaluated by the well-known Goudar-Sonnad approximation for rough pipes [30]. Similar formulas are available for standard flow configurations like bends, contractions etc. [31] For the heat loss, the situation is slightly more complicated as the pipe wall may be built of several layers (e. g. inner pipe, insulation and shell) and the pipe may be laid either above or below ground. Therefore, the overall conductive heat transfer coefficient of the pipe (often this value is listed on the data sheet of the pipe, e. g. [32], or can be approximated by a multilayer model [33]) must be added to the convective heat transfer coefficient computed according to Gnielinski equation [33,34]. Then, assuming a steady-state condition and neglecting the temperature variation in the cross directions of the pipe, for every infinitesimal pipe element of length dx (see Fig. 7), the change of the fluid's enthalpy equals the heat loss through the pipe walls:

$$-\dot{m}c_p\, dT(x) = U(T(x) - T_{\text{ambient}})\cdot D\pi\, dx$$

Thereby T_{ambient} is the ambient temperature either above or below ground, U is the total heat transfer coefficient as discussed before, and D denotes the diameter of the pipe.

Integrating this equation along the pipe length L yields the correlation given by Adihou et al. [35]

$$T_{\text{out}} = T_{\text{ambient}} + (T_{\text{in}} - T_{\text{ambient}})\exp(-\tau) \tag{5}$$

with

$$\tau = \frac{ULD\pi}{\dot{m}c_p}$$

and T_{in} and T_{out} the inflow and outflow temperatures of the pipe. Afterwards, the power loss through the entire pipe wall can be easily computed from

$$dP_{\text{pipe}} = \dot{m}c_p(T_{\text{in}} - T_{\text{out}}) \tag{6}$$

4.3. Mass flow solver

Assuming a quasi-steady state condition, the mass flow boundary condition found in 4.1. for the substations dictates the mass flow distribution in the whole quantum network. Due to mass conservation, for every quantum element except the "TElectrons" the following equation holds:

$$\dot{m}_{\text{element}}(t) = \sum_{\text{next elements}} \dot{m}_{\text{next}}(t) \tag{7}$$

where the "next elements" are exactly the quantum elements the pointers of the "next" vector are pointing to. That is straightforward for all elements having only one successor, i. e. a "next" vector of length one – the outflow of the current element equals the inflow of the next element – but due to mass conservation of the fluid it is also valid for all flow dividers, i. e. all "next" vectors of length ≥ 2.

Equation (7) has been implemented into a recursive function "compute_massflow". Its flow chart is shown in Fig. 8. Thereby it is important to note that the argument of the function is of type Quantum pointer and

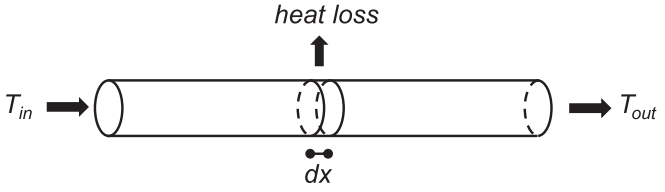


Fig. 7. Heat balance in an infinitesimal pipe element of length dx.

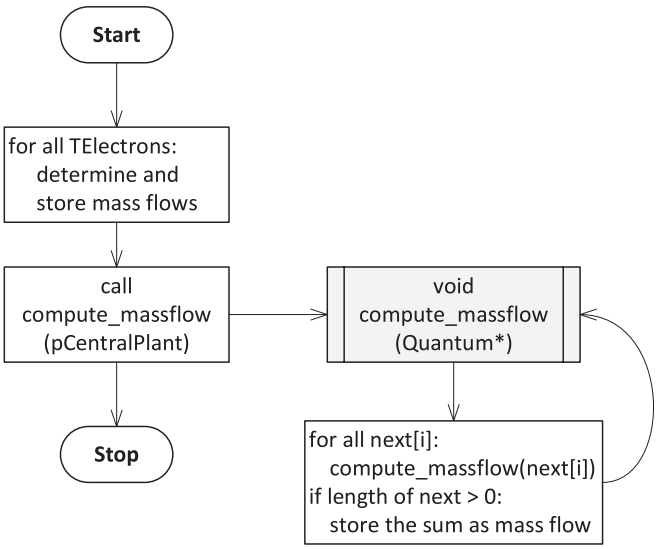


Fig. 8. Flow Chart of Mass Flow Solver.

thus the function is callable for all kinds of quantum elements. The function makes use of the vector of "next" pointers that was defined as a member variable of every quantum element. If the length of the "next" vector is bigger than zero, the function "compute_massflow" is called recursively from within itself when evaluating the right side of equation (7). Only if the "next" vector is empty what is the case only for the "TElectrons", the function does not start a new recursive call and exits without doing anything.

To make full use of this recursive function from within another function or a main() programme it is essential that the individual mass flows of the "TElectrons" are determined first. Next the function "compute_massflow" is called with its argument being a pointer "pCentralPlant" that points to the central plant. Then, thanks to the recursive set-up, the algorithm steps down by itself into the quantum network's hierarchy until it hits on a "TElectron" having a known mass flow, and computes the mass flows for all the other quantum elements.

4.4. Temperature solver

The temperature solver also follows the idea of a recursive function call but this time it is a bit more complicated as can be seen from Fig. 9. The idea is to evaluate the temperatures in the network by following the actual flow direction. That means that the temperatures on the supply path are computed in a top-down manner starting from the central plant and going down to the substations whereas the temperatures along the return path are calculated bottom-up.

Before invoking the recursive temperature solver "solve_temperature" one has to make sure that all heat-relevant properties have been initialized or calculated. Those are for example the temperature differentials dT of the substations and the ambient temperatures, but also the mass flows and the heat loss factors τ in equation (5) that are needed to evaluate the heat loss in pipes or other quantum network elements. Afterwards and in analogy to the mass flow solver, the recursive function "solve_temperature" is called with the argument being a pointer to the central plant.

For every quantum element except the "TElectrons", the function "solve_temperature" always consists of the same three steps. First the heat loss and the outflow temperature for the supply side are computed following equation (5). The inflow temperature of all subsequent elements is set equal to this outflow temperature. Thus it is always ensured that the inflow temperature into a certain quantum element is known when the function "solve_temperature" is invoked for that element. Then the temperature solver is called for all element pointers in the "next"

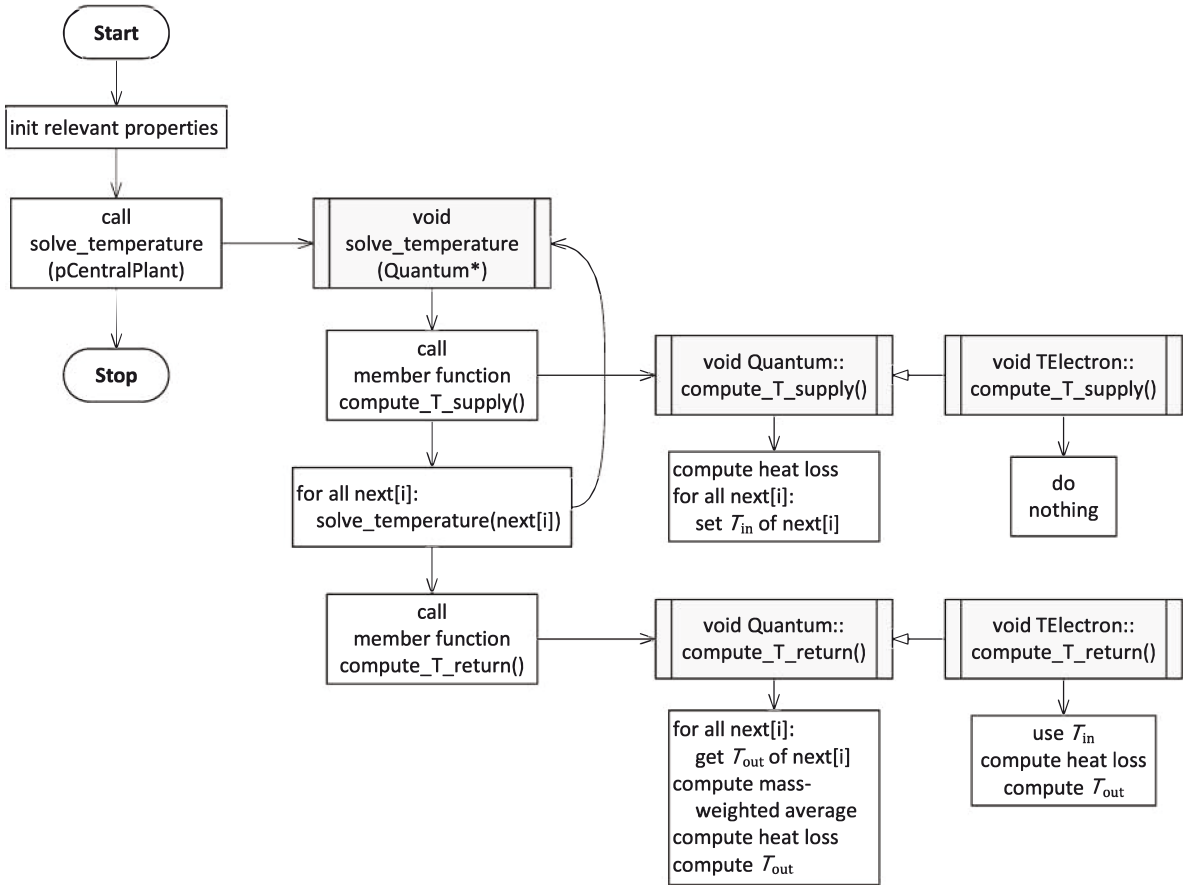


Fig. 9. Flow Chart of Temperature Solver.

vector. Lastly and with the results from the previous step, the heat loss and outflow temperature on the return path can again be evaluated by equation (5). Thereby the inflow temperature on the return side is computed as the mass-weighted average of all outflow temperatures from the quantum elements the "next" pointer is pointing to.

The calculation for the "TElectrons" that are the lowest quantum elements in the quantum network hierarchy works differently. Nevertheless, the overall procedure as depicted by the flow chart can be retained if the member functions "compute_T_supply" and "compute_T_return" of the "Quantum" are overridden for "TElectron". The override of function "compute_T_supply" simply does nothing, whereas the override of "compute_T_return" evaluates the temperatures of the whole section between the supply side's inflow into the substation until the outflow on the return side. Note that the intermediate "solve_temperature" is not called for the case of "TElectrons" as the vector "next" is empty.

4.5. Solving procedure

The overall solving procedure is depicted in Fig. 10. All input data as well as the solver settings are read from text files. After creating and initializing the network's geometry, the first time-step is initialized and solved. First the mass flow solver is called, then the temperature solver. This loop is repeated for all time-steps. In every step, the current results are redirected to output file streams that make use of buffering and that come with the standard library of C++ [25,26].

5. Application to district heating networks

In addition to various test cases, the quantum network solver has been used for the analysis of two district heating networks (DHNs). The operating conditions of the central plant and the substations, the topology of the network including the lengths, diameters and characteristics of the pipes as well as the outside temperatures and the annual energy need of each district were provided as input variables. In the first case the substations were equipped with ideal heat exchangers whereas in the second case a model for heat pumps was used.

5.1. High temperature (HT) network with heat exchangers

The first case treats a typical 3rd generation DHN that was modelled as a quantum network (Estavayer-le-Lac, Switzerland, master thesis of J. Rolle, see Fig. 11). It is fed with water of 84 °C by a central plant consisting of three burners (2 wood, 1 gas) with a maximum capacity of 5.5 MW. The 53 substations of the network are operated with a constant differential $dT = 15$ °C so that the return temperature to the central plant is expected to be close to 69 °C. All substations are equipped with loss-free heat exchangers, i. e. formula (1) is applied for the power transfer between substation and district. Most of the districts are either apartment blocks or individual housings, but there are also three schools and one commerce. All districts need energy for heating and sanitary hot water with a maximum annual demand of 900 MWh/year. Thereby the reference temperatures T_{ref} in (3) vary between 12 to 18 °C.

The simulation presented below was run for the outside temperatures of year 2019 (as provided by Groupe E Ltd. for the master thesis of J. Rolle) with a time-step size of one hour (see Fig. 12 (a)). Even if it must be kept in mind that some physical effects are strongly simplified or not yet implemented in the computational model, we can summarize several important results. For the given annual energy needs of the districts, the total power supply during the year can be seen from Fig. 12 (b). As expected, it is high on cold winter days reaching a maximum of slightly larger than 3 MW and rather low during the summer period. The total

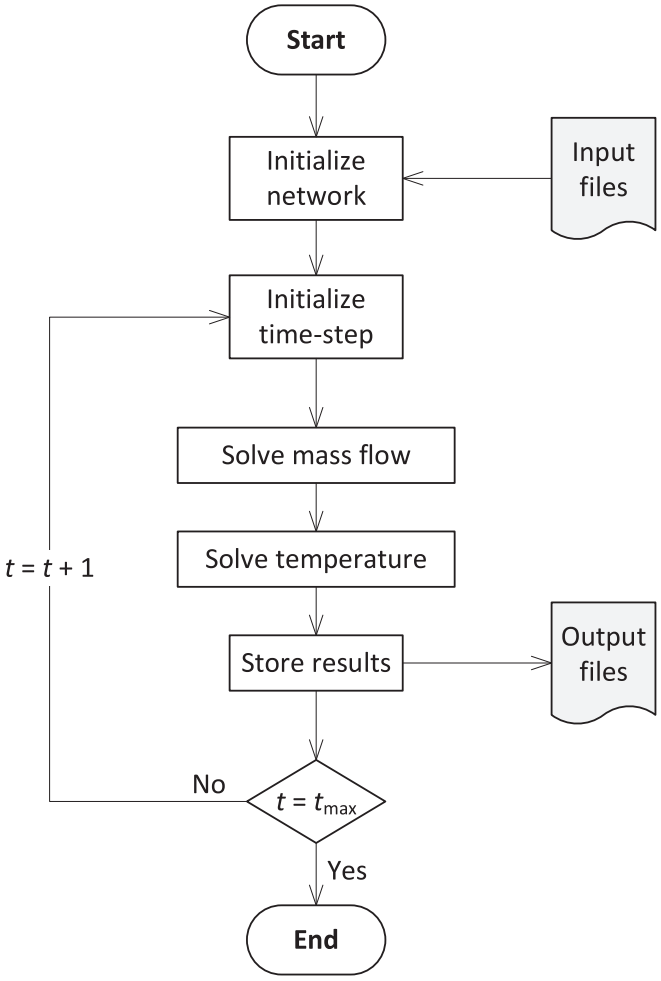


Fig. 10. Quantum Network Solver.

power loss in the pipes of the DHN can be seen from Fig. 12 (d). Again, it is highest on the coldest moments in winter but this time, the dependency of outside temperatures is less strong. It is interesting to see that during summer, the inlet temperatures of many substations (see Fig. 13 (a)) and the return temperature to the central plant (see Fig. 12 (c)) take extremely low values as soon as the need for sanitary hot water is low (e. g. during nights). That is due to the fact that the mass flows circulating in the network are extremely low when little power is needed and thus the water has a lot of time to cool down. Indeed, if substation 23 – an individual housing with a heating reference temperature of 14 °C – is picked out as an example (see Fig. 13 (b-c)), one can see that in the summer months the inflow temperature is generally lower but still takes peak values around 70 °C as soon as the need for sanitary hot water is high whereas it cools down below 30 °C if the need is low. However, if this is considered as a problem, an easy remedy would be to fix a lower bound for the mass flow circulating in the branches.

Integrating the above results for power over the full year yields a total energy of 7.8 GWh/year provided by the central plant and a total energy loss of approximately 900 MWh/year in the pipes which corresponds to 11.5 % (see Table 2).

The computation of these results takes about 1 s of real time on a laptop computer (excluding reading/writing of files). This is extremely fast in comparison to a previous inhouse code implemented in MATLAB (master thesis of J. Rolle) which needed several minutes to solve the same problem. Remember that this speed-up was made possible a) by restricting oneself to a very stringent way of discretizing a DHN, b) by an efficient recursive solver making use of the advantages of object-oriented programming and pointers in C++ and c) by the much higher ground speed of C++ as a compiled language.

Comparison to other DHN solvers: In general, it is difficult to find a proper way of benchmarking different solvers against each other, as all are based on different assumptions and include different physical models. It is clear that direct solvers are always much faster than iterative ones, but on the other hand they are restricted to a limited range of problems. With our approach of rewriting the original DHN into a tree-shaped quantum network, we are enlarging this range. Guelpa and Verda [23] pursue a similar idea when deleting certain edges from their geometries in order to get a tree-shaped topology. They compare a not further described direct solver in conjunction with their method to an iterative solver and find an acceleration of 2 to 3 orders of magnitude.

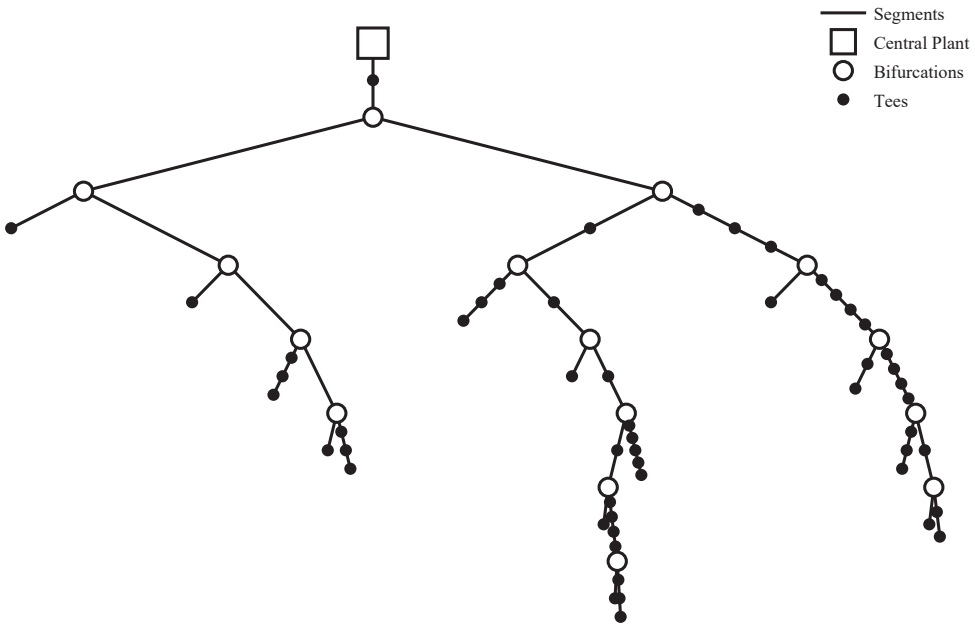


Fig. 11. Quantum Network Model of the HT Network.

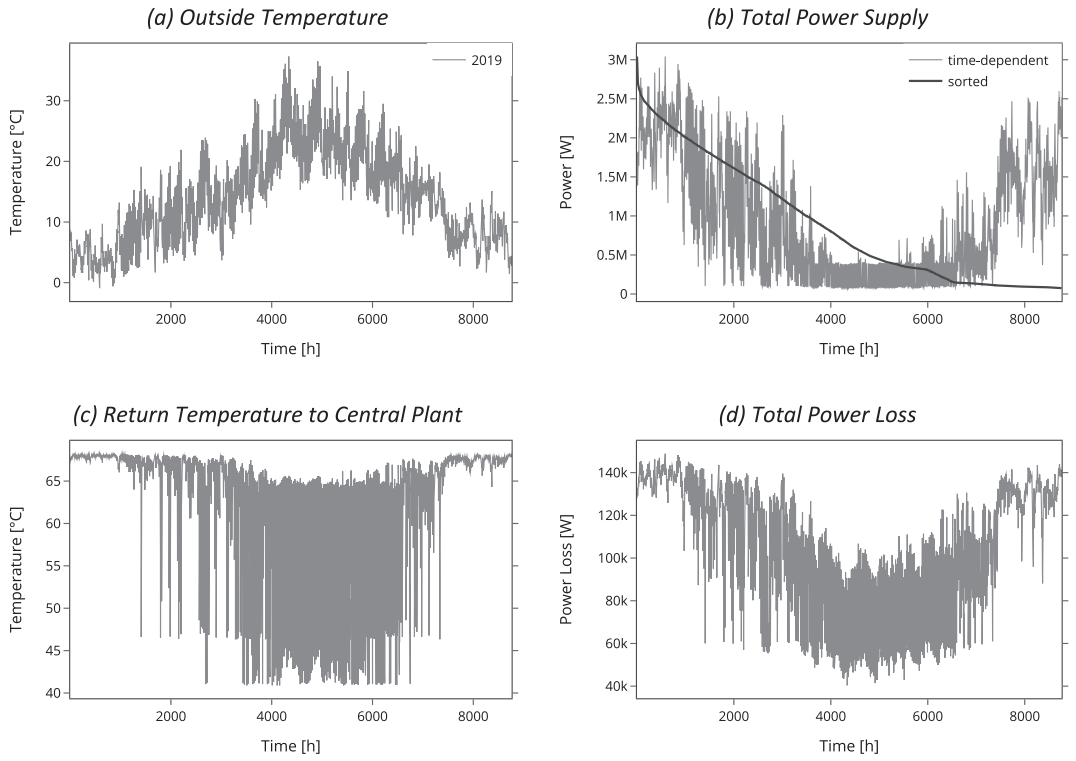


Fig. 12. Time-dependent Outside Temperature (a) and Results (b-d) for the HT Network.

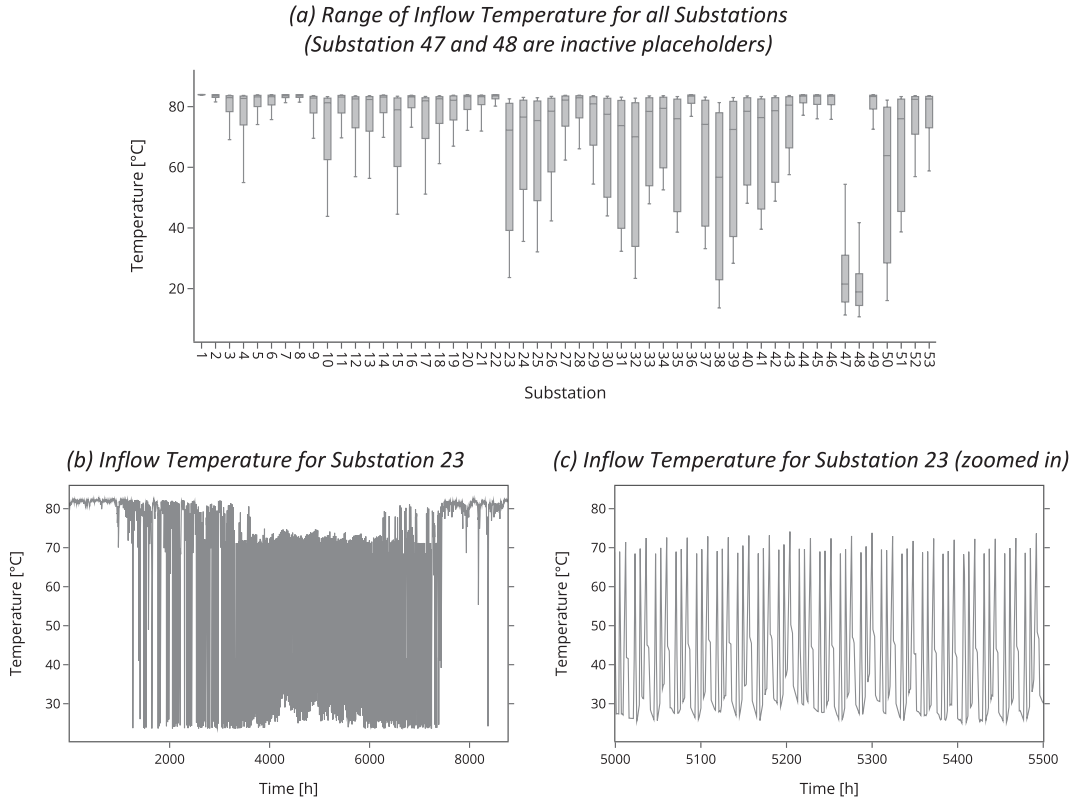


Fig. 13. Inflow Temperatures for Substations (HT Network).

Unfortunately, from the provided information it cannot be explained why both of their solvers show an exponential increase of computational time if the number of consumers in the network is increased. In contrast, Vorspel and Bücker [13] use an iterative Python solver although their setup is restricted to branched networks. They talk about long runtimes, about how to accelerate the simulation by advanced means of high-performance computing (multiprocessing and GPU computing), about incompatibilities in this regard of Python's Numpy library, and last but

Table 2
Boundary conditions and simulation results for the HT and LT network.

	HT network	LT network
Number of substations	53	20
Length [km]	3.7	3.1
Supply temperature [°C]	84	10
Energy demand of Districts [MWh/year]	6930	3653
Energy from Central Plant [MWh/year]	7828	2381
Energy from Heat Pumps [MWh/year]	–	1243
Energy loss in pipes [MWh/year]	898 (11.5 %)	−29 (−1.2 %)
Maximum Power from CP [MW]	3.05	0.89

not least about convergence issues. Without digging deeper into the code, we downloaded their example of a branched network with ten consumers from Github [36] and run it for a full year: it took more than 80 min for one setup. Afterwards, we gave DHNx [12] a try and prolongated their minimal example of a forked network with two customers over 8760 time-steps. Here the computational time was a bit more than one minute. It must be mentioned that in the current version DHNx is also limited to tree-shaped networks, though it is planned to extend its capabilities in the future. The simulation of the full Estavayer-le-Lac network as shown in Fig. 11 took 6–7 min with DHNx (more than our inhouse MATLAB solver). The results show a perfect agreement with our findings for the mass flows and a good agreement for the pressure losses and temperatures. But unfortunately, the subsequent evaluation of the heat losses seems to contain a bug and yields results that are orders of magnitudes too high. For example, for segment $(l, k) = (0, 1)$ the agreement is very good as our solver finds a yearly heat loss of 5.60 MW and DHNx finds 5.57 MW. But for the next segment $(l, k) = (0, 2)$ which is approximately twice as long and of same type, we find 10.96 MW as expected, but DHNx claims 2274 MW. Anyhow, the major difference to emphasize is not the bug but the computational speed for the same quality of results.

The solvers PyDHN [16] and GridPenguin [18] were not tested as they aim to cover the transient dynamics what is naturally of higher computational cost.

5.2. Low temperature (LT) network with heat pumps

The second example considers a case of a typical low-temperature DHN of the newest generation (La Tour-de-Peilz, Switzerland; analysed in the master thesis of S. Rime). This time, there are no burners in the central plant, but the power supply of the network is ensured by heat exchangers located in the depths of a big lake where the water temperature is (almost) constant throughout the year. In the studied case on the shores of Lake Geneva, we assumed an inflow temperature of 10 °C into the DHN.

The 20 substations of the network (15 apartment blocks, 4 individual houses, 1 school; see Fig. 14) now need to be equipped with heat pumps in order to augment the temperature from the supplied 10 °C to the temperatures requested by the districts between 40 and 67 °C. When evaluating formula (2) it must be kept in mind that the *COP* of a heat pump depends on the two adjacent temperatures. If it is not known a priori, a C++ routine modelling a basic heat pump's thermodynamic cycle will be called. Further, due to missing information for this particular DHN, all heating thresholds T_{ref} were set to 18 °C whereas the differentials dT of the substations were taken from measurements and are in the range between 0.15 and 2.36 °C (and constant with respect to time).

The overall simulation results for a full year using an hourly time-step (see outside temperatures of 2021 for MeteoSwiss station in Pully [37] in Fig. 15 (a)) are again listed in Table 2. The power supply over time from the central plant (see Fig. 15 (b)) looks qualitatively as expected with a maximum of 0.9 MW on the coldest day in mid-February. On the other hand, the power loss in the pipes yields a picture that is very different from the high-temperature case with positive values in winter, but negative values in summer (see Fig. 15 (d)). In fact, the warm outside temperatures during summer months in conjunction with the low mass flows are heating up the water circulating in the pipes. This can be seen even more clearly when looking at the return temperature to the central plant (see Fig. 15 (c)): in the winter months, it is rather constant around 8.9 °C whereas in summer it reaches values up to 11.5 °C. This effect is not at all a problem, but even an additional source of sustainable energy.

Equally, the inflow temperature into the substations is higher in summer than in winter, thus supporting the heat pumps and reducing the need of electrical energy. For example, the inflow temperature of substation 11 can be seen in Fig. 16 (a) and the corresponding calculated *COP* of its heat pump working at a target temperature of 55 °C is shown in Fig. 16 (b).

Only if the costs for heat pumps and electrical power are known in comparison to traditional network components and resources, one can decide if this new generation of DHNs is more profitable. In addition, it has to be kept in mind that the availability of electrical power is limited,

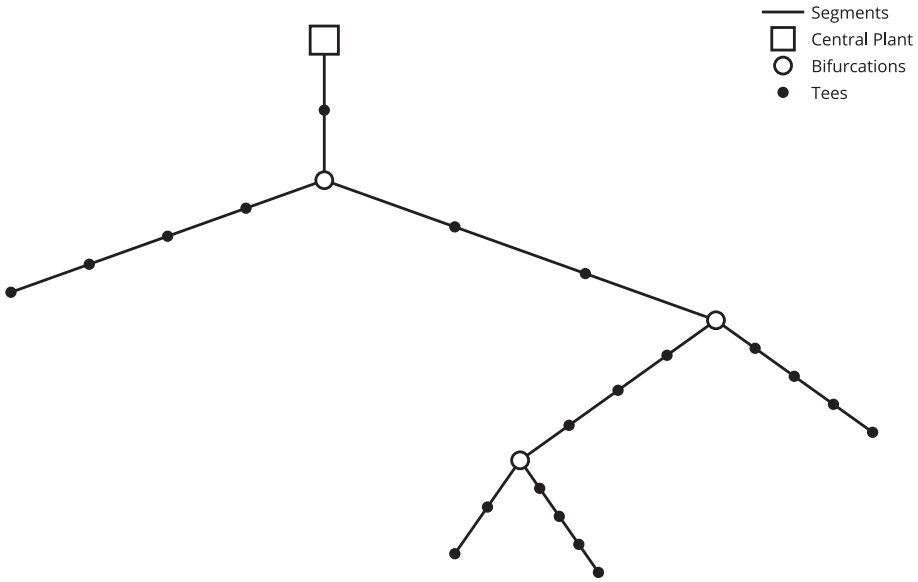


Fig. 14. Quantum Network Model of the LT Network.

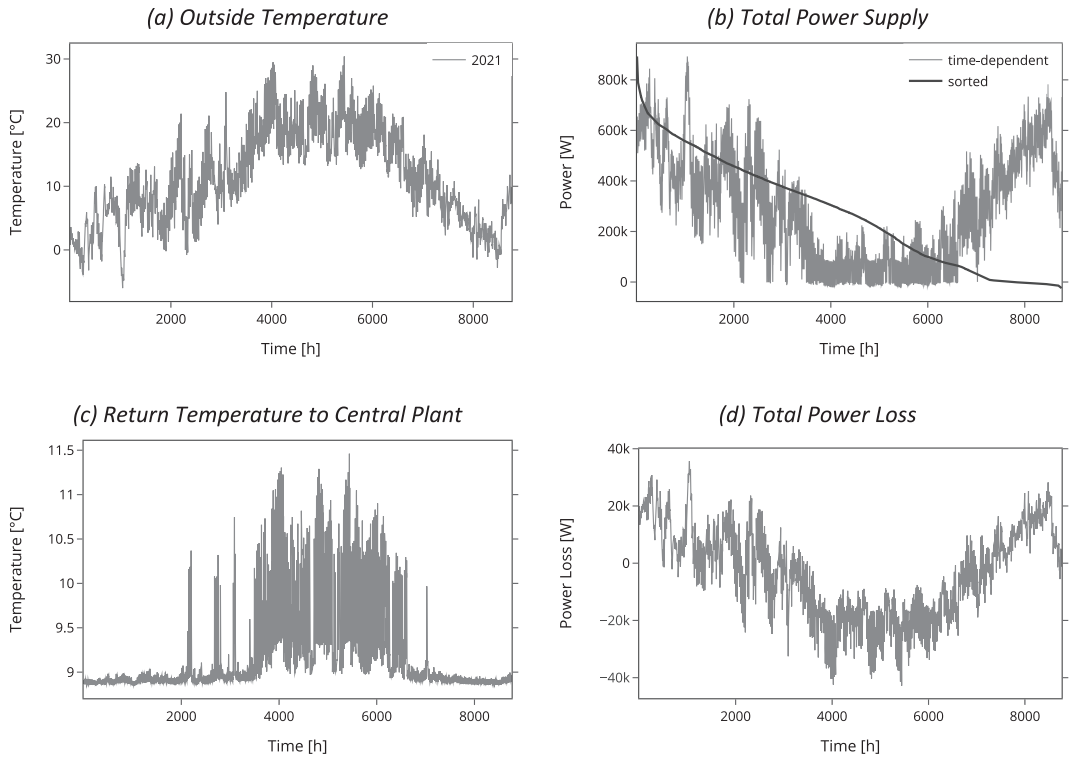


Fig. 15. Time-dependent Outside Temperature (a) and Results (b-d) for the LT Network.

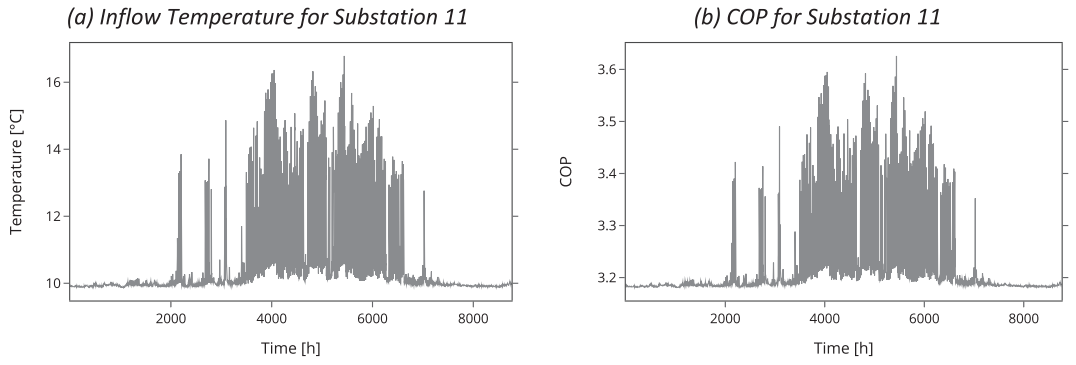


Fig. 16. Inflow Temperature and COP for Substation 11 (LT Network).

especially during the winter months [4,38]. Research is ongoing in order to increase the share of renewable energies and to find better solutions for energy storage and efficiency [39,40].

6. Outlook

From the perspective of potential non-academic users, the biggest pain point of our current library is the lack of a graphical user interface (GUI). We have already developed a tool for the visualisation of our results, but we still miss functionalities for the import of existing GIS geometries and for the setup of the simulation.

6.1. Adding further mathematical models

But also on the methodological side, the current state will be enhanced. Thanks to the object-oriented structure of the quantum network and its solvers, it is always possible to add more advanced models to the library.

Some concepts can be seen as an enhancement of the already implemented quantum elements and are rather straightforward. For example, the singular pressure losses of elbows along the pipes, of flow dividers and unifiers or inside the substations are not yet included. Same is true for residual flows at the end of branches or for altitude differences along the network. As already mentioned in section 4.1., the model for the consumers' energy demand can be improved as well. Other ideas concern the evaluation of local or global energy or exergy balances or the approximation of cost functions.

More challenging, but also much more crucial is the implementation of the "TCluster" class. This class is absolutely mandatory when it comes to simulating DHNs of a more general topology. At the same time, it will enable the application of sub-modelling techniques that can be used to compute the interconnection of several DHNs.

Ultimately it would be interesting to investigate the electricity needs of the network, to add solar panels to the substations and to couple the flow-thermal simulation with the electrical grid.

6.2. Coupling with genetic algorithm

In comparison to the previous inhouse MATLAB code [21,22] and to many other DHN solvers, the method presented here for quantum networks is extremely fast. This makes it very attractive to be applied for optimisation studies where a huge number of configurations needs to be quickly analysed.

Every optimisation problem consists of a set of decision variables and an objective function. In most cases, there are additional constraints that must be respected. The goal is to find a combination of values for the decision variables such that the objective function takes its minimum (or maximum). In the case of DHNs, a goal might be for example to minimize the overall pressure loss, the heat loss and the investment costs by choosing different types of pipes. The challenge hereby is that these goals are often contradictory: a cheaper pipe may have a lower diameter and thus a greater pressure loss or it may have a weaker insulation and thus a greater heat loss. Therefore, the objective function has to be formulated in such a way that it computes a global objective as weighted combination of the different criteria. In addition, only a limited product range of pipes is available on the market – the pipes that fulfil the mathematical optimum cannot be ordered. Consequently, the affected decision variables are restricted to some discrete values whereas others might still be continuous.

An optimiser that can deal with either discrete or continuous decision variables including additional constraints is a genetic algorithm. Its idea is to start with a random initial population and to score its individuals. Then, depending on these scores, some of the individuals ("the elite") are directly propagated to the next generation. Others ("the parents") are coupled in pairs to create a child that inherits a combination of their genes ("crossover"). The last group are random mutations from the previous individuals. Hereby the genes of each individual represent the decision variables, the score is related to the objective function.

The performance of a genetic algorithm strongly depends on how the new generations are formed. Unfortunately, in general there is no guarantee of convergence to a local or global optimum but by construction each population is at least as good as the previous one. Usually, the algorithm is terminated after a predefined number of generations or when it hits a stop criterion. Nevertheless, its strength is its applicability to almost all kind of optimisation problems, regardless of whether their objective function is continuous or discrete, linear or highly nonlinear, constrained or unconstrained etc.

The C++ Genetic Algorithms Library GAlib from Massachusetts Institute of Technology [41] has already been tried out in our organisation and seems to be a good choice in conjunction with the presented quantum network solver. Our intention is to couple it with the model of a full district heating network.

7. Conclusion

In the present article, an object-oriented solver for a so-called "quantum network" has been presented. A quantum network thereby refers to a subclass of district heating networks (DHNs) that can be modelled as a binary tree as was discussed in section 2. After introducing the relevant principles of object-oriented programming and the most important C++ classes implemented in the scope of this project, the recursive procedure of the quasi-steady solver for mass flow and temperature was depicted. This solver was then used for the analysis of two specific examples, namely a high-temperature and a low-temperature DHN.

Our main conclusions from this work can be summarized below the following headlines:

Standardised DHN definition: The standardisation that comes with the definition of quantum networks is twofold. First, it is assumed that the underlying DHN can be transformed into a binary tree. This is directly true for all networks of branched topology and can be achieved for many other DHNs using simplifications, sub-modelling and/or clustering. Second, instead of just giving each element of a DHN model a unique identifier, the stringent numbering convention for quantum networks (see section 2.) helps to identify the relative position of elements in the DHN without the need of additional lookup tables. The approach presented in this article can always be applied for all DHNs fulfilling the prerequisites, no matter of size or exact shape.

Implementation into an object-oriented library: The developed code is organised in a hierarchy of consecutive C++ classes starting from a very basic base class "Quantum" and becoming increasingly detailed. Indeed, it is a fundamental principle of object-orientation that each class can always be extended by a specialisation, and as long as all components of the network are derived from the same base class, the use of the modules and the workflow of the solver will still remain the same. As a direct consequence of this modularity, it is straightforward to use sub-modelling techniques, to implement additional models or to modify the level of detail taken into account without changing the overall functionality.

High computational speed: Using a previous inhouse MATLAB solver, it took several minutes to solve time-dependent cases similar to the ones presented in section 5, but now the computational effort is in the order of one second. This improvement was made possible mainly by a completely new recursive algorithm taking advantage of the object-oriented design, but also by the use of the high-level programming language C++ known for its high execution speed. Thanks to this speed-up, the new solver is suitable not only for the computation of a single DHN but also for optimisation studies where an enormous number of variants need to be calculated.

Scalability and parallelisability: The storage of the DHN model in a structure similar to a linked list and the recursive design of the solver guarantee a good performance for bigger DHNs. Although no systematic evaluation has yet been carried out, a computational complexity that is linearly dependent from the number of substations can be expected. In addition, the recursive design spawning more and more processes when stepping down in the quantum network hierarchy makes it opportune for parallel computing leading to an even higher solver efficiency if very big models are to be studied.

Multi-purpose application: In contrast to many DHN software packages that restrict themselves to a certain class of applications, the presented library can equally be used for rough estimates in the early design phase, detailed analyses of elaborated DHNs, a combination of both when existing DHNs need to be redesigned or extended, as well as in conjunction with other coupled solvers or optimisers.

CRediT authorship contribution statement

Cornelia Blanke: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Malick Kane:** Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgement

The work was supported by the New Regional Policy (NRP) of Canton of Fribourg and by Groupe E Ltd., SINEF Ltd., PSE Energies LLC,

and Town of Fribourg as part of the ADVENS project [PCS-2021-07]. We would like to thank the aforementioned organisations for many fruitful discussions about the real challenges of designing district heating networks. We particularly appreciated the fact that Groupe E Ltd. shared information and data on their district heating networks which formed the basis for several master theses and finally for this work.

References

[1] Werner S. International review of district heating and cooling. Energy 2017;137:617–31. https://doi.org/10.1016/j.energy.2017.04.045.
[2] Lund H, et al. 4th Generation District Heating (4GDH): Integrating smart thermal grids into future sustainable energy systems. Energy 2014;68:1–11. https://doi.org/10.1016/j.energy.2014.02.089.
[3] Buffa S, Cozzini M, D'Antoni M, Baratieri M, Fedrizzi R. 5th generation district heating and cooling systems: A review of existing cases in Europe. Renew Sustain Energy Rev 2019;104:504–22. https://doi.org/10.1016/j.rser.2018.12.059.
[4] Allegrini J, Orehounig K, Mavromatidis G, Ruesch F, Dorer V, Evins R. A review of modelling approaches and tools for the simulation of district-scale energy systems. Renew Sustain Energy Rev 2015;52:1391–404. https://doi.org/10.1016/j.rser.2015.07.123.
[5] Sarbu I, Mirza M, Crasmareanu E. A review of modelling and optimisation techniques for district heating systems. Int J Energy Res 2019;43(13):6572–98. https://doi.org/10.1002/er.4600.
[6] Brown A, Foley A, Laverty D, McLoone S, Keatley P. Heating and cooling networks: A comprehensive review of modelling approaches to map future directions. Energy 2022;261:125060. https://doi.org/10.1016/j.energy.2022.125060.
[7] Kuntuarova S, Lickleder T, Huynh T, Zinsmeister D, Hamacher T, Perić V. Design and simulation of district heating networks: A review of modeling approaches and tools. Energy 2024;305:132189. https://doi.org/10.1016/j.energy.2024.132189.
[8] Schweiger G, et al. District energy systems: Modelling paradigms and general-purpose tools. Energy 2018;164:1326–40. https://doi.org/10.1016/j.energy.2018.08.193.
[9] Casella F. Simulation of large-scale models in modelica: state of the art and future perspectives. In: Presented at the the 11th International Modelica Conference; 2015. p. 459–68. https://doi.org/10.3384/ecp15118459.
[10] Hirsch H, Nicolai A. An efficient numerical solution method for detailed modelling of large 5th generation district heating and cooling networks. Energy 2022;255:124485. https://doi.org/10.1016/j.energy.2022.124485.
[11] Röder J, Meyer B, Krien U, Zimmermann J, Stührmann T, Zondervan E. Optimal design of district heating networks with distributed thermal energy storages – method and case study. Internat J Sustain Energy Plann Manage 2021;31:5–22. https://doi.org/10.5278/ijsepm.6248.
[12] jnnr, Röder J, MaGering O Akca, Zimmermann J. oemof/DHNx: Gorgeous Grids. Zenodo; 2020. doi: 10.5281/zenodo.4147049.
[13] Vorspel L, Bücker J. District-heating-grid simulation in python: DiGriPy. Computation 2021;9(6). https://doi.org/10.3390/computation9060072. Art. no. 6.
[14] Witte F, Tuschy I. TESPy: thermal engineering systems in python. J Open Source Software May 2020;5(49):2178. https://doi.org/10.21105/joss.02178.
[15] Boghetti R, Peronato G, Kämpf J. Verification of PyDHN - a Python library for the thermo-hydraulic simulation of district heating networks - through the DESTEST. Presented at the 2023 Building Simulation Conference. 2023. 10.26868/25222708.2023.1592.
[16] Boghetti R, Kämpf JH. Verification of an open-source Python library for the simulation of district heating networks with complex topologies. Energy 2024;290:130169. https://doi.org/10.1016/j.energy.2023.130169.
[17] Dénarié A, Aprile M, Motta M. Dynamical modelling and experimental validation of a fast and accurate district heating thermo-hydraulic modular simulation tool. Energy 2023;282:128397. https://doi.org/10.1016/j.energy.2023.128397.
[18] *ftbv/grid-penguin*. (Mar. 19, 2024). Python. ftbv. Accessed: Jul. 23, 2024. [Online]. Available: https://github.com/ftbv/grid-penguin.
[19] Girardin L.A GIS-based Methodology for the Evaluation of Integrated Energy Systems in Urban Area, PhD Thesis, Ecole Polytechnique Fédérale de Lausanne, Lausanne, 2012. doi: 10.5075/epfl-thesis-5287.
[20] Loustau J, Lepour D, Terrier C, Maréchal F. Clustering and typification of urban districts for energy system modelling. In: Proceedings of ECOS 2023 - The 36th international conference on efficiency, cost, optimization, simulation and environmental impact of energy systems, Las Palmas De Gran Canaria, Spain; 2023. p. 3206–17. doi: 10.52202/069564-0288.
[21] Kane M, Rolle J. 'Quantum networks': a new approach for representing a network and evaluating hydraulic and thermal losses in district heating/cooling systems. Proceedings of ECOS 2020 - The 33rd international conference on efficiency, cost, optimization, simulation and environmental impact of energy systems, Osaka, Japan. 2020.
[22] Rime S, Kane M, Wyler S. Alternative solutions for the optimal integration of decentralized heat-pumps in district heating/cooling networks. Proceedings of ECOS 2020 - The 33rd international conference on efficiency, cost, optimization, simulation and environmental impact of energy systems, Osaka, Japan. 2020.
[23] Guelpa E, Verda V. Compact physical model for simulation of thermal networks. Energy 2019;175:998–1008. https://doi.org/10.1016/j.energy.2019.03.064.
[24] M. Kane, L. Schulthess, and J. Rolle, "A 'quantum network' theory to model, design and operate a district's energy system." unpublished 2023.
[25] Stroustrup B. The C++ programming language. 4th ed. Westport, Conn: Addison-Wesley Professional; 2013.
[26] M. Müller and S. Schwarzer, "C++, Eine Einführung," 2004. Accessed: Apr. 22, 2024. [Online]. Available: https://fs.hlrs.de/projects/par/events/2006/prog_lang_spring2006/c++.pdf.
[27] Bansal AK. Introduction to programming languages. CRC Press; 2013.
[28] Guelpa E. Impact of network modelling in the analysis of district heating systems. Energy 2020;213:118393. https://doi.org/10.1016/j.energy.2020.118393.
[29] SIA Zürich, "SIA 2024 / 2015 F - Données d'utilisation des locaux pour l'énergie et les installations du bâtiment." Accessed: Apr. 22, 2024. [Online]. Available: https://shop.sia.ch/normenwerk/architekt/sia%202024/f/2015/F/Product.
[30] Goudar CT, Sonnad J. Comparison of the iterative approximations of the Colebrook-White equation. Hydrocarbon Process 2008;87. pp. 79-80+83.
[31] Idelchik IE, Steinberg MO, Martynenko OG. Handbook of hydraulic resistance. 3rd ed. Mumbai: Jaico Publishing House; 2005.
[32] "PREMANT Rohrsystem für Ihr Hochtemperatur-Fernwärmenetz." Accessed: Apr. 22, 2024. [Online]. Available: https://www.bruggpipes.com/fernleitungsrohr-premant/uno/.
[33] "VDI-Wärmeatlas", 11th edition, 2013. Berlin Heidelberg: Springer Vieweg, 2013.
[34] Gnielinski V. Ein neues Berechnungsverfahren für die Wärmeübertragung im Übergangsbereich zwischen laminarer und turbulenter Rohrströmung. Forsch Ing-Wes 1995;61(9):240–8. https://doi.org/10.1007/BF02607964.
[35] Adihou Y, Kane M, Ramousse J, Souyri B, editors. An exergy-based methodology to determine thermal network's optimal temperature level; 2021.
[36] Vorspel L, Bücker J. lvorspel/DiGriPy: First release of DiGriPy. Zenodo; 2021. doi: 10.5281/zenodo.4956223.
[37] "Data portal for teaching and research (IDAweb) - MeteoSwiss." Accessed: Nov. 20, 2023. [Online]. Available: https://www.meteoswiss.admin.ch/services-and-publications/service/weather-and-climate-products/data-portal-for-teaching-and-research.html.
[38] T. Marti, M. Sulzer, and M. Rüdisüli, "Energieversorgung der Schweiz bis 2050. Zusammenfassung von Ergebnissen und Grundlagen (Studienbericht)." Verband Schweizerischer Elektrizitätsunternehmen VSE: «Energiezukunft 2050». Wege in die Energie und Klimazukunft der Schweiz, Dec. 13, 2022. Accessed: Apr. 29, 2024. [Online]. Available: www.energiezukunft2050.ch.
[39] Jodeiri AM, Goldsworthy MJ, Buffa S, Cozzini M. Role of sustainable heat sources in transition towards fourth generation district heating – A review. Renew Sustain Energy Rev 2022;158:112156. https://doi.org/10.1016/j.rser.2022.112156.
[40] Sarbu I, Mirza M, Muntean D. Integration of renewable energy sources into low-temperature district heating systems: a review. Energies 2022;15(18). https://doi.org/10.3390/en15186523.
[41] M. Wall, "GAlib: Matthew's C++ Genetic Algorithms Library." Accessed: Apr. 25, 2024. [Online]. Available: https://web.mit.edu/galib/www/GAlib.html, User Guide: http://lancet.mit.edu/ga/dist/galibdoc.pdf.