A new modular procedure for industrial plant simulations and its reliable implementation

Digital Object Identifier (DOI): http://dx.doi.org/10.1016/j.energy.2015.10.122

A new modular procedure for industrial plant simulations and its reliable implementation

C. Carcasci[a], L. Marini[a], B. Morini[a], M. Porcelli[b]

[a] *Università degli Studi di Firenze, Dipartimento di Ingegneria Industriale, via di S. Marta 3 e Viale Morgagni 40, 50134 Firenze, Italy.*
[b] *Università di Bologna, Dipartimento di Matematica, Piazza di Porta S. Donato 5, 40127 Bologna, Italy.*

Abstract

Modeling of industrial plants, and especially energy systems, has become increasingly important in industrial engineering and the need for accurate information on their behavior has grown along with the complexity of the industrial processes. Consequently, accurate and flexible simulation tools became essential yielding the development of modular codes. The aim of this work is to propose a new modular mathematical modeling for industrial plant simulation and its reliable numerical implementation. Regardless of their layout, a large class of plant's configurations is modeled by a library of elementary parts; then the physical properties, compositions of the working fluid, and plant's performance are estimated. Each plant component is represented by equations modeling fundamental mechanical and thermodynamic laws and giving rise to a system of algebraic nonlinear equations; remarkably, suitable restrictions on the variables of such nonlinear equations are imposed to guarantee solutions of physical meaning. The proposed numerical procedure combines an outer iterative process which refines plants characteristic parameters and an inner one which solves the arising nonlinear systems and consists of a trust-region solver for bound-constrained nonlinear equalities. The new procedure has been validated performing simulations against an existing modular tool on two compression train arrangements with both series and parallel-mounted compressors..

Keywords:
Industrial plants simulation, Modular 0/1-D codes, Fully implicit methods, Constrained nonlinear systems of equations, Trust-region Gauss-Newton

Email addresses: carlo.carcasci@unifi.it (C. Carcasci), leopoldo.marini@unifi.it (L. Marini), benedetta.morini@unifi.it (B. Morini), margherita.porcelli@unibo.it (M. Porcelli)

Accepted Version in ENERGY:
Volume 94, 1 January 2016, Pages 380-390

methods.

1. Introduction

Industrial plants are subject to standard requirements as low equipment costs, high energy conversion/transmission efficiency, low environmental impact and high operational flexibility. In particular, in industrial plant design and in-service behavior prediction, high calculation accuracy and competitive computational time are fundamental to meet the customers' needs. These are the reasons why traditional methods for such simulations involve the use of numerical 0/1-D codes, known to fully satisfy the above requirements. In particular, the dedicated approach leads to procedures for specific plant configurations where either none or a few input data are allowed to vary. Over the last decades, this approach has progressively been replaced by a modular one that can handle general plant's layout and general input data.

Recently, there has been a renewed interest in global plant landscape driven by the increasing demand for low-cost energy and reduced environmental impact; this has led to continuous efforts for enhancing the elementary plants' components and to the theoretical and practical study of alternative thermodynamic cycles. On one side, this process resulted in a general complication of the plants' arrangements and, correspondingly, in a high demand of flexible tools to numerically estimate the plants' performance. On the other hand, it yielded to the description of a large variety of plant solutions as a combination of a finite number of elementary components (pumps, heat exchangers, valves, turbines, compressors) connected with each other. Therefore a numerical code equipped with a database of elements representing their physical behavior and suited to combine these with general criteria, turns out to be much more versatile than a code designed for a specific plant's configuration. These types of codes are known as *modular* codes and are generally characterized by the following properties [1]. They must be able to:

- create a plant configuration without requiring a new program source;
- handle any combination of input data if a sufficient number of parameters for the plant's solution is provided;
- find the characteristic parameters of the elementary components, even with an increased number of input data.

Since the early 1990s many modular codes for plants' simulation have been developed to fulfill the growing needs of flexible and fast numerical tools for the study of complex flow networks; firstly Perz [2] built an elemental code for mass and heat balances, named `IPSEpro`, that has been subsequently developed with commercial

purposes by Simtech Simulation Technology [3]. The program, due to its simplifications, is able to model only the behavior of ideal gas [4]. Carcasci and Facchini [1] proposed a modular code for applied research that, grown over the years, handles thermodynamic [5], design and off-design analysis of industrial plants [6, 7]. Falcetta and Sciubba [8, 9], similarly, developed a modular tool, named `CAMEL`, that has become a commercial code owned by Altran [10]. Also Carapellucci and Cau studied a modular procedure based on fundamental thermodynamic relations, including real gas behavior [11], mainly used for power plants simulations [12]. Many other modular-based codes have been developed for commercial uses since their origin; examples include `THERMOFLEX`, property of Thermoflow Inc. [13], that is a fully flexible program for heat balance modeling and engineering, particularly suited to model both conventional [14] and unconventional power plants, like solar ones [15]; GE's `GateCycle` [16], a professional tool for both the gas and steam sides of power plant design and analysis [17]; `Prosim`, a modular simulation and design environment for power processes [18], developed by Endat Oy (Prosim, `www.endat.fi`). The great advantages of the modular approach have yielded the development of a whole class of object-oriented programming languages, among which the most notably is `Modelica`, introduced in 1996 within the project ESPRIT [19]. `Modelica` deals with component-oriented modeling of complex physical systems consisting of mechanical, electrical, hydraulic, thermal and control equipment [20]. The algorithms and programming approaches adopted in both such commercial and applied research-based codes are rather obscure and often confidential; this makes extremely hard to gain an in-depth understanding of their properties and potentials in plants'simulations.

One of the issues which greatly influences the performance of the codes but has not been thoroughly discussed in the literature consists in the numerical solution of the equations for process simulation. In particular, a common task of all the above-mentioned codes is the solution of a set of equations, including differential and algebraic equations, that represents the physical behavior of the modeled problem. Such set can be either split into subsets of equations, each coming from a particular module [21], that are solved using a proper sequential approach (alike the approach adopted in dedicated codes) or solved simultaneously by a *parallel/full implicit mode*. None of above papers offer details for this algorithmic phase.

Differential equations have a marginal role in industrial plants' simulations, since the modular codes are generally zero dimensional and model steady state flows, whereby neither spatial nor temporal evolution of the phenomena is taken into account within each element. On the contrary, systems of nonlinear equations (i.e., systems where at least one equation is not linear) constitute the mathematical models for a surprisingly large number of problems of real concern, as they

model both the behavior of dynamic and thermodynamic systems, through the discretization of ordinary or partial differential equations, and equilibrium systems, see e.g. [1, 2, 4, 10, 18, 22, 23].

Newton method and its variants are arguably the most popular class of procedures for solving nonlinear systems of algebraic equations [24] and they are used in most of the above codes [4, 10, 18, 22, 23]. It is well-known that Newton method is an iterative procedure and its convergence depends critically on the vicinity of the initial guess to a solution of the nonlinear system. For practical applications in modular codes, this feature may represent a severe limitation as the nonlinearity of the equations and the number of unknowns make the location of the roots highly difficult. In order to enhance convergence, Newton's method is combined with so-called globalization strategies which include linesearch and trust-regions methods, see e.g. [24, 25, 26, 27, 28, 29, 30, 31]. An alternative solution strategy, implemented in codes for industrial plants' simulations [1, 2], consists in a simplification of the mathematical problem to be solved where, by means of first-order Taylor expansion, nonlinear equations are replaced with linear equations. The resulting problem is a linear system which can be solved with standard algorithms such as Gauss-Jordan Elimination or Lower-Upper Decomposition [1, 2]. Clearly, though computational cheaper than Newton method, this approach may provide an inaccurate approximation to a solution of the original problem and does not overcome the need to locate solutions of the nonlinear system.

A further issue that requires modification of standard procedures for solving nonlinear systems derives from the fact the systems of our interest are constrained. Specifically, bounds have to be imposed to find a solution of physical meaning, e.g., a solution where absolute pressures are positive, and possibly to restrict the search space for a desired solution, see e.g. [26, 32].

The aim of this paper is to provide a detailed overview of a new modular procedure for industrial plants' simulation that can handle a broad class of plant's layouts through a wide library of elementary components, and determine the physical properties and composition of the working fluid, as well as plant's performance, in steady state operational conditions. Each module of the developed code has been made as independent as possible from the others, enhancing flexibility and allowing for upgrades; e.g. the solver of the arising nonlinear systems of equations can be readily replaced or improved without altering the other parts of the code. The core of this implementation is given by use of the nonlinear optimization solver `TRESNEI` for bound constrained nonlinear least-squares problems [29]. This solver implements a trust-region Gauss-Newton method and is suitable for the solution of "smooth" problems, that is problems described by continuously differentiable functions. It provides enhancements with respect to standard solvers for nonlinear systems in the following respects:

- being a solver for bound constrained problems, once proper upper and lower bounds for the variables are fixed, it prevents the computation of undesirable solutions lying outside from the feasible solution's domain;
- by implementing a globally convergent method, it avoids the tricky issue of selecting an initial guess close enough to the problem's solution.

Moreover, since our approach does not rely on simplified versions of the nonlinear system, the solutions computed are expect to be more accurate than those obtained with the approach in [1, 2].

In this paper, a thorough description of both our modular procedure and `TRESNEI`'s algorithm is provided. Our goal is to offer a scheme that can serve as a template to users interested in reproducing, and possibly adapting, our code. The performance of the proposed modular procedure is illustrated on two compression train arrangements with both series and parallel-mounted compressors; the results obtained have been compared with `ESMS` [1], a pre-existing in-house modular tool, based on a Gauss-Jordan solver, that has been widely validated over a broad range of industrial plants' problems [5, 6, 7]. The comparison shows a good agreement between the results of the two codes and comparable computational speed.

2. The modular approach

In this section we describe the mathematical model for industrial plants and focus on the formulation of bond's equations and on the solution of the resulting constrained nonlinear system of equations.

2.1. Description of the plant

An industrial plant consists of a certain number of elementary components such as pumps, heat exchangers, valves, turbines, compressors, etc., where thermodynamic, energetic or chemical transformations of the operational flows take place; flows can be of mass, power or heat. In order to model this plant structure, one can define a list/library of elementary components where each transformation (thermodynamic transformation, mass and energy flows' continuity) is mathematically described. Then, in a plant configuration setting phase, components in the library can be used whenever similar ones are present in the considered plant.

In this paper, we use the symbols and notations given in Figure 1: a single elementary component is referred as a unit; depending on the internal flows, each unit has some inlet and outlet ports which handle the corresponding working flows and generate the system networking. Generally, the number of inlet and outlet ports is different in units representing different types of components. At each unit's

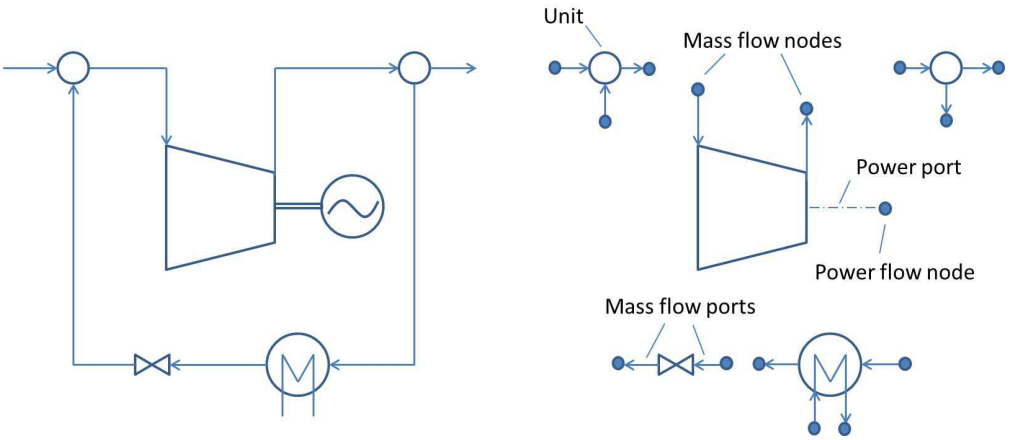

Figure 1: A compressor plant and its modular model.

port corresponds a node that, in accordance with the passing through operational flow, can be classified as mass, power or heat flow node. Therefore, flows connect the units through nodes.

In every single node, the flow state can be fully determined by the following properties:

- *mass flow* properties: mass flow rate, chemical compositions (mass/molar fractions of the chemical species) and thermodynamic parameters (pressure, temperature, enthalpy) [1];
- *power flow* properties: mechanical power and rotational speed (typically the power flows represent shafts);
- *heat flow* properties: thermal power.

Clearly the flow state continuity condition holds between adjacent nodes.

Every unit can be treated as a black-box that represents a particular energetic transformation and links the flow properties of each operational flow between the entry and exit nodes. A unit is also characterized by some typical parameters that affect its performance, as for example efficiency, pressure or heat losses.

Solving the plant consists in finding all the flow properties in each node and all the typical parameters of each plant's element. To pursue this issue, some of

[1]Temperature, pressure and enthalpy together might appear redundant in expressing a flow's thermodynamic state, as only the last two could be sufficient, even when working with multi-phase fluids. The choice to use both the thermodynamic parameters temperature and enthalpy is due to the fact that usually operational conditions are given in terms of temperature and not in terms of enthalpy. The use of these properties guarantees maximum flexibility of the code.

these properties and typical parameters are known and constitute the *boundary conditions* of the problem. Notably, for this modular framework, there is no need to specify which conditions have to be imposed and in which node, provided that there is a sufficient number of independent parameters for the plant's solution. Moreover, it is important to remark that the values of the flow properties should satisfy some *bound constraints* which are generally specified by the plant designer and guarantee that the computed values have physical meaning, e.g. trivially, absolute pressure must be nonnegative.

2.2. Mathematical model overview

Once defined the plant's layout, the physical processes are modeled in mathematical terms.

Let a plant be composed by N units with inlet and outlet connections. Let $N_{M,j}$ be the number of mass flow ports, $N_{W,j}$ be the number of mechanical connections and $N_{Q,j}$ be the number of heat flow ports of each unit, where the subscript j refers to the j-th unit. For each port, depending on the kind of passing through flow there are some unknown parameters that represent the flow properties. These unknowns are $4 + N_S$ associated to mass flow rate, pressure, temperature, enthalpy and compositions of the N_S involved species, two associated to power and rotational speed, one associated to power. Consequently, the number of problem's unknowns is

$$\sum_{j=1}^{N}(4 + N_S)N_{M,j} + 2N_{W,j} + N_{Q,j}. \tag{1}$$

Three kinds of governing equations are necessary to describe the plant: the flow continuity equations, the bond's equations and the boundary conditions. The continuity equations impose the conservation of flow properties between connected nodes; since the number of flow properties depends on the type of node, let n_f be the number of connections between two mass flow's nodes of different elements and n_p and n_h be the number of connections between power and heat flow's nodes. Then the total number of continuity equations is:

$$(4 + N_S)n_f + 2n_p + n_h. \tag{2}$$

The unit bond's equations describe the physical transformations occurring in each component of the plant (mass balances, energy balances, adiabatic relations for expansion or compression, equations of state, heat exchanges and many other), and are generally nonlinear (e.g., the equation $TP^{\epsilon} = cost$, representing the adiabatic expansion or compression of gases, is typically nonlinear). Since each unit determines the number of bond's equation, denoted as $N_{BE,j}$, the total number of

bond's equations is:

$$\sum_{j=1}^{N} N_{BE,j}. \tag{3}$$

Finally, the last kind of equations is represented by boundary conditions that fix the value of known flow properties in some nodes of the plant and characterize the solution of the problem; let N_{BC} be the number of these boundary conditions.

Summarizing, if the overall number of unknowns and equations coincides, i.e.

$$\sum_{j=1}^{N}(4+N_S)N_{M,j} + 2N_{W,j} + N_{Q,j} = N_{BC} + \sum_{j=1}^{N} N_{BE,j} + N_{CE}, \tag{4}$$

then the system of nonlinear equations is square, i.e. the number of equations equals the number of unknowns. Under suitable assumptions on the nonlinear function, the solutions are locally unique and, imposing reasonable physical bounds on the variables, it is expected to have only one solution.

Interestingly, this modular approach allows the definition of a fully implicit mathematical model for a plant. Hence, differently from several existing sequential or semi-parallel approaches adopted in dedicated simulators, the nonlinear equations can be solved simultaneously. Advantages of this feature are threefold: the problem setting and the solution is not affected from the ordering of the plant's elements; how and where imposing the boundary conditions of the problem is not relevant; it is possible to set boundary conditions in order to reduce the number of required operational parameters, e.g. efficiency.

2.2.1. Formulation of the bond's equations

The system of equations described above is nonlinear since some bond's equations, such as those involving gas compression/expansion, pressure losses or linking the thermodynamic properties through thermodynamic libraries, are nonlinear. Besides these nonlinear equations, the modeled transformations also include linear equations thus yielding a large variety of bond's equations from a wide library of elementary components.

Handling and solving such equations requires a computational effort proportional to the number of bond's equations. Therefore, in order to reduce both the code maintenance and the computational overhead, it is necessary to keep the number of different equations' types as low as possible. This issue can be addressed by arranging bond's equations into the four groups given below; as a consequence, by varying the multiplicative and the exponential constants in the equations, a number of equations is obtained, each characterizing a specific thermodynamic transformation. We refer to Table A.5 for details on the nomenclature. In particular, in the following formulas, subscripts equal to 1 and 2 refer to inlet and outlet

ports, respectively.

The equations in the first group state: continuity equations; mass, energy, heat balances; simple equations involving power, rotational speed, heat, mass flow, temperature, pressure and enthalpy. They take the form

$$\sum_{l=1}^{N_W}(\kappa_{l,W}W^{\epsilon_{l,W}} + \kappa_{l,\omega}\omega^{\epsilon_{l,\omega}}) + \sum_{l=1}^{N_Q}\kappa_{l,Q}Q^{\epsilon_{l,Q}} + \tag{5a}$$

$$\sum_{i=1}^{N_M} k_{i,1}M_{1,i}^{e_{i,1,M}}T_{1,i}^{e_{i,1,T}}P_{1,i}^{e_{i,1,P}}H_{1,i}^{e_{i,1,H}} + \tag{5b}$$

$$\sum_{j=1}^{N_M} k_{j,2}M_{2,j}^{e_{j,2,M}}T_{2,j}^{e_{j,2,T}}P_{2,j}^{e_{j,2,P}}H_{2,j}^{e_{j,2,H}} = k_{known}, \tag{5c}$$

where W and ω are power and rotational speed respectively, Q is the heat flow, M, T, P, H are mass flow rate, temperature, pressure and enthalpy respectively. The scalars N_W, N_Q and N_M denote the number of power, heat and mass ports of the unit respectively; κ and ϵ are the multiplicative and exponential constants of the power and heat ports' parameters (the subscript indicating the variable they correspond to); similarly, k and e are the constants of the mass ports' parameters. Finally, k_{known} is the known right-hand side of the equation.

The second set of equations concerns chemical species,e.g. the continuity of species' concentration, and is given by

$$\sum_{i=1}^{N_M} k_{i,1}M_{1,i}^{e_{i,1,M}}\left(\sum_{n=1}^{N_S} K_{i,1,y_n}y_{i,2,n}^{E_{i,1,y_n}}\right) + \tag{6a}$$

$$\sum_{j=1}^{N_M} k_{j,2}M_{2,j}^{e_{j,2,M}}\left(\sum_{n=1}^{N_S} K_{j,2,y_n}y_{j,2,n}^{E_{j,2,y_n}}\right) = k_{known}, \tag{6b}$$

where y is the chemical molar/mass concentrations of the flow (depending on the species n), N_S represents the number of chemical species involved, K and E are the multiplicative and exponential constants of the chemical concentrations.

The third type of equations states, in a general form, pressure losses within a duct as a function of mass flow, temperature or pressure,

$$D_{1,P}P_{1,i}^{\eta_{1,P,1}} + D_{2,P}P_{1,i}^{\eta_{1,P,2}}P_{2,j}^{\eta_{2,P,1}} = \tag{7a}$$

$$D_{1,M}\left(|M_{1,i}|^{\eta_{1,M,1}}M_{1,i}T_{1,i}^{\eta_{1,T,1}}P_{1,i}^{\eta_{1,P,3}}\right) + \tag{7b}$$

$$D_{2,M}\left(|M_{2,j}|^{\eta_{2,M,1}}M_{2,j}T_{2,j}^{\eta_{2,T,1}}P_{2,j}^{\eta_{2,P,2}}\right) + k_{known}, \tag{7c}$$

where D are the multiplicative constants of the various terms (the first subscript refers to the inlet/outlet port, the second one to the related term); η are the power of the flow's parameters (the third subscript indicates each flow's parameter may appear more than once in the above equation, e.g. the inlet pressure can appear a maximum of three times). Subscripts i and j specify respectively the number of inlet and outlet ports to which flow's parameters are associated, since pressure losses can occur within an element with multiple inlet/outlet ports.

The fourth type of equations concerns the thermodynamic properties of the fluids. Since a consistent modeling of real fluids requires an accurate computation of the thermodynamic properties of pure fluids and mixtures, we use thermodynamic libraries for real gas and water-steam behaviour. Functions that bind the state variables (pressure, temperature and enthalpy) at every thermodynamic state of the working flow are considered and the equations take the following exponential form

$$T + BH^b + CP^c = k_{known}. \tag{8}$$

Clearly, the solution of equation (8) in the unknowns T, H and P, requires the knowledge of the constants B, b, C and c and k_{known}. These quantities are computed for each node solving an auxiliary nonlinear system of 5 equations in the 5 unknowns B, b, C and c and k_{known} with low accuracy. This system is built by considering starting values for pressure P and enthalpy H and slightly perturbing these values forwardly and backwardly from the starting known points, obtaining 5 equations of the form (8). Then, the system is solved in the variables B, b, C and c and k_{known}.

In conclusion, equations (5)-(8) are the general formulation of all the bond's equations governing the physical problem considered. The constants involved are determined for each elementary unit. It is important to note that further bond's equations can be easily added to the code if those already developed and included in the model are not sufficient to describe the plant.

2.3. Solving the mathematical model

The equations introduced in the previous section can be stated as a square nonlinear system of equations, say $F(x) = 0$, with $F : \mathbb{R}^n \to \mathbb{R}^n$. Taking into account the physical meaning of the variables and imposing inequality bounds, the mathematical model described in the previous sections takes the form

$$\begin{aligned} &F(x) = 0, \\ &l \leq x \leq u. \end{aligned} \tag{9}$$

where l, $u \in \mathbb{R}^n$ are given. Letting the i-th component of a vector x be denoted by either x_i or $(x)_i$, we suppose that $-\infty \leq l_i < u_i \leq \infty$, $i = 1, \cdots, n$ and let the inequalities $l \leq x \leq u$ be meant componentwise.

The methods for solving problem (9) are iterative and a reliable methodology should have the following characteristics:

- the ability in locating solutions is mathematically guaranteed;
- for any initial iterate the procedure either converges to a solution or fails to so do in a small number of detectable ways;
- the rate of convergence is fast (at least close to a solution) in order to approximate, in a small amount of time, a solution at a prescribed accuracy.

The numerical approach for solving (9) followed in this paper, was proposed in [28] and it is based on the minimization of the function

$$f(x) = \frac{1}{2} \|F(x)\|_2^2, \tag{10}$$

where $\|\cdot\|_2$ indicates the euclidean norm, i.e. $\|F(x)\|_2 = \sqrt{F(x)^T F(x)}$. Since f is a nonnegative function and vanishes at the solutions of (9), it is minimized including the simple bounds in (9), i.e.

$$\min_{l \le x \le u} f(x). \tag{11}$$

The method used in this work, denotes as `TRESNEI` [29](Trust-REgion Solver for Nonlinear Equalities and Inequalities), was developed in [28] and originally implemented in the MATLAB freely accessible solver, `http://TRESNEI.de.unifi.it`. We refer to such a MATLAB implementation as a template for different programming languages.

The necessary conditions for the optimality at a point x can be expressed as ([33])

$$\tilde{D}(x)\nabla f(x) = 0, \tag{12}$$

where $\nabla f(x) = F'(x)^T F(x)$, $F'(x) \in \mathbb{R}^{n\times n}$ is the Jacobian matrix of F, and $\tilde{D}(x) \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonal entries $(\tilde{D}(x))_{i,i}$ given by

$$(\tilde{D}(x))_{i,i} = \begin{cases} x_i - u_i & \text{if } (\nabla f(x))_i < 0,\ u_i < \infty, \\ x_i - l_i & \text{if } (\nabla f(x))_i \ge 0,\ l_i > -\infty, \\ 1 & \text{if } (\nabla f(x))_i \ge 0,\ l_i = -\infty, \text{ or} \\ & \quad (\nabla f(x))_i < 0,\ u_i = \infty. \end{cases}$$

Let a point x be denoted as feasible if $l \le x \le u$. Given a feasible initial guess x_0, `TRESNEI` is an iterative procedure such that

- the iterates x_k generated are feasible;

- irrespective of the initial guess x_0 used, every limit point of the sequence $\{x_k\}$ satisfies (12);
- if x^* is a limit point of $\{x_k\}$ such that $f(x^*) = 0$, then all the limit points of $\{x_k\}$ solve the problem (9).

In order to fulfill the above features, we use a *trust-region Gauss-Newton* scheme sketched below. Theoretically, the stated convergence properties are guaranteed if F' is Lipschitz continuous and bounded in norm in an open, bounded and convex set containing the level set $\{x \in \mathbb{R}^n \text{ s.t. } f(x) \leq f(x_0)\}$ [28, 34].

The basic idea of a trust-region method is to fix the radius Δ_k of a ball about x_k in which the quadratic model

$$m_k(p) = \frac{1}{2}\|F'(x_k)\,p + F(x_k)\|_2^2, \tag{13}$$

for f can be trusted to accurately represent the function. The ball $\{p \in \mathbb{R}^n \text{ s.t. } \|p\|_2 \leq \Delta_k\}$ is called the trust-region and Δ_k is the trust-region radius. Then, by using the so-called trust-region problem

$$\min_{p\in\mathbb{R}^n}\{m_k(p) : \quad \|p\|_2 \leq \Delta_k\}, \tag{14}$$

and an appropriate adjustment of Δ_k, it is possible to enforce a strictly monotonic reduction in the value of f through the iterates. The quadratic model (13) is known in the literature as the Gauss-Newton model.

The progressive decrease of f is guaranteed by imposing specific conditions on the acceptance of the iterates. Suppose that the sequence $\{x_k\}$ has the form $x_{k+1} = x_k + p_k$ for $k \geq 0$. By [28, 33] it is known that the first-order optimality conditions (12) are satisfied at every limit point of $\{x_k\}$ if the step p_k satisfies

$$\rho_c(p_k) = \frac{m_k(0) - m_k(p_k)}{m_k(0) - m_k(p_k^C)} \geq \beta_1, \qquad \beta_1 \in (0,1), \tag{15}$$

where p_k^C is the scaled Cauchy step defined as

$$p_k^C = \underset{p\in span\{-\tilde{D}(x_k)\nabla f(x_k)\}}{\operatorname{argmin}} m_k(p) \quad \text{subject to} \quad \|p\|_2 \leq \Delta_k, \;\; l \leq x_k + p \leq u, \tag{16}$$

Therefore, in order to find a suitable step p_k, first we compute a step p_{tr} by (approximately) solving the trust-region problem (14). Second, we form the projected step $\bar{p}_{tr} = P(x_k + p_{tr}) - x_k$, where $P(x) = \max\{l, \min\{x, u\}\}$ is the projection map onto the set $l \leq x \leq u$, and find a step of the form $p_k = t\,p_k^C + (1-t)\bar{p}_{tr}$, for some $t \in [0,1]$, such that $\rho_c(p_k) = \beta_1$.

Finally, the trust region radius and the trial point $x_k + p_p$ are tested simultaneously. In particular, the predicted reduction of the quadratic model m_k and the actual reduction of the objective function f at the trial point $x_k + p_k$ are compared using the standard rule

$$\rho_f(p_k) = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \geq \beta_2, \qquad \beta_2 \in (0, 1). \tag{17}$$

If (17) is satisfied, then a reduction in the value of f has been obtained, the trial point is accepted, and a new iteration begins with possibly a larger trust-region radius. Otherwise the step is rejected and the trust-region radius is reduced.

We conclude this section giving some algorithmic details. The parameters β_1 and β_2 and the rules for the trust-region update are given in [29]. The Jacobian matrix $F'(x_k)$ is formed by using finite differences. In particular, the sparsity pattern of the Jacobian is detected and the nonzero entries $(F'(x_k))_{i,j}$ are computed by setting

$$(F'(x_k))_{i,j} \approx \frac{1}{h_j}(F_i(x_k + h_j e_j) - F_i(x_k)),$$

where e_j is the j-th vector of the canonical basis of $\mathbb{R}^n$, ϵ_m is the machine precision and

$$h_j = \begin{cases} \sqrt{\epsilon}_m & \text{if } (x_k)_j = 0 \\ \sqrt{\epsilon}_m sign((x_k)_j) \max\{|(x_k)_j|, \|x_k\|_1/n\} & \text{otherwise .} \end{cases}$$

If the point $x_k + h_j e_j$ is not feasible, the backward approximation

$$(F'(x_k))_{i,j} \approx \frac{1}{h_j}(F_i(x_k - h_j e_j) - F_i(x_k),$$

is used.

Successful termination in the solution of (11) is declared when one of the following conditions is met

$$\|F(x_k)\|_\infty \leq \epsilon_1, \tag{18}$$

$$\min\{\|\tilde{D}(x_k)\nabla f(x_k)\|_2, \|P(x_k - \nabla f(x_k)) - x_k\|_2\} \leq \epsilon_2\sqrt{n}, \tag{19}$$

where ϵ_1 and ϵ_2 are prescribed tolerances and $\|\cdot\|_\infty$ indicates the ∞-norm, i.e. $\|F(x)\|_\infty = \max_{1\leq i\leq n} |(F(x_k)_i|$. In the simulation reported in Section 5, bound-constrained nonlinear systems are solved with high accuracy setting $\epsilon_1 = \epsilon_2 = 10^{-12}$.

3. Solution of the model

3.1. The overall iterative procedure

The definition of the bound-constrained nonlinear system depends on the knowledge of several coefficients present in the bond's equation, as e.g. the quantities B, b, C, c and k_{known} in equation (8). Once these values are determined, the solution of the nonlinear system is performed by the trust-region solver and in the remaining of this work, we refer to the iterations in this phase as to *inner* iterations.

The estimate of various constants can be performed in two manners. One possibility is to update these coefficients at each inner iteration. However, this could be time consuming, as many of such scalars depend on the thermodynamic properties of the working flow which are computed by dedicated database libraries. Another possibility, adopted in this procedure, is to approximate the coefficients, solve the nonlinear equations and use the solution computed for updating the coefficients. This procedure is iterative and we call its iterations as *outer* iterations.

More specifically, at the beginning of each outer iteration, the models defining each elementary unit are solved to get the coefficients of the bond's equations. After that, the resulting system of equations is solved iteratively. It is interesting to note that if the bond's equations and thermodynamic properties of real mixtures are not related, the solution computed after the first outer iteration remains unchanged, as no modifications on the coefficients occur between successive iterations. Suitable termination criteria are included to established the convergence of the outer procedure (see Section 4).

3.2. Reducing the system dimension

The mathematical model described in Section 2 has the same number of equations and unknowns. However, the number of independent variables is generally less than the number of unknowns introduced for each node of the plant. In fact, some flow properties are constant between connected elements and within each element. This property is known as *continuity* and can be of two different types: internal and external. The former concerns the element's behaviour and has effect on a bond's equation; e.g. it represents the mass and species continuity in a real compression inside the unit compressor. The latter is implicit as it does not arise from a unit bond's equation and would require imposing a further equation; e.g. in the case of two elements connected via a mass flow port, the thermodynamic parameters (M, P, T, H) and the flow's composition are the same in the two linked nodes. The simple form of continuity equations suggests a reduction in the dimension of the system by eliminating proper unknowns.

For equation (5), for example, continuity between two power, rotational speed or heat occurs when the known term is null and the coefficients κ and ϵ have value equal to 1 for two power or heat ports, and value equal to 0 for all the others,

including mass flow ports. The same happens for continuity of mass, temperature, pressure and enthalpy, but with coefficients k and e and in equation (6).

Finally, it is important to note that external continuity leads to two identical bond's equations (8) for each mass flow node of the plant. Hence, these bond's equations are included in the system of equations only if related to mass flow's outlet ports or to inlet ports connected to the outside.

3.3. The initialization procedure

Initial values of the flow properties in each node of the plant, except for boundary conditions, are unknown at the first iteration along with the coefficients of bond's equations. These initial values have a great impact on the convergence speed and success of the solver's iterative process. Three possible initialization approaches are possible. The first is a *default mode*: for each unit, initial values of the flow properties are fixed accordingly to standard working performance of the corresponding modeled element. A second possibility consists in choosing some of the flow properties' values on the base of the designer experience; one can initialize the calculation completely or partially by combining this approach with the default mode. Finally, in the case of sensitivity analysis or with little plant's layout modifications, it is reasonable to use a previously computed solution as the starting guess. Once the starting values of the flow properties have been fixed, the initial coefficients of the bond's equations can be straightforwardly obtained.

We underline that global convergence of the trust-region solver allows to overcome the difficulties in selecting an initial hint sufficiently close to the solution.

4. Structure of the code

The modular approach described in the previous sections was implemented using the ANSI Fortran 90 standard. The code is outlined in the flowchart in Figure 2 and consists of three parts: the main program, the units' subroutines (or modules) of the various elements, the solver for the systems of nonlinear equations generated.

The main program handles the various phases of the solution process: problem setting, system formulation, outer/inner iterations. The modules give rise to the equations for the flow transformations and get the coefficients of the bond's equations. The nonlinear solver is the core of the code and performs the inner iterations. Each of these parts is separated from the others allowing for maximum flexibility and expandability; therefore an easy reading, maintenance and expansion of the code is ensured.

More specifically, the main program first calls the subroutines to acquire information about plant's layout, general chemical compositions of the working fluids,

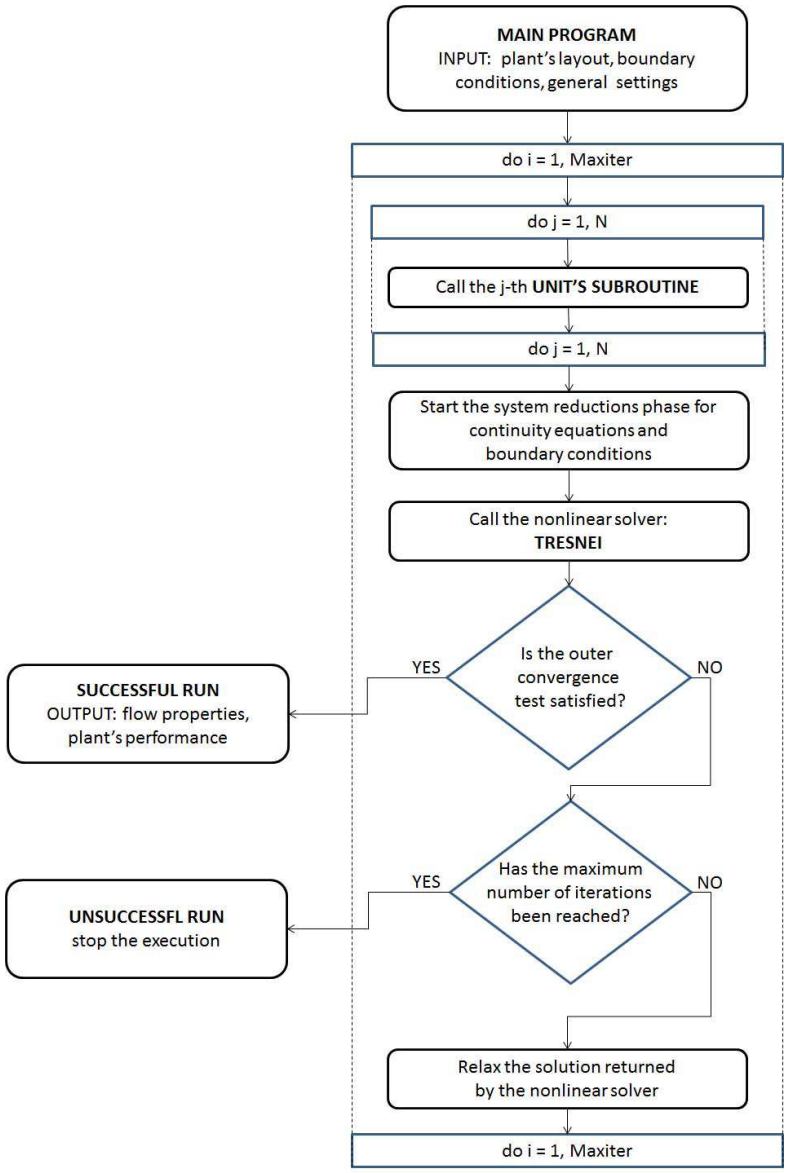

Figure 2: Program flow chart.

boundary conditions and code's general settings. Then it performs the outer iterations; in each of such iteration it calls the units' subroutines to obtain the coefficients of the bond's equations, takes into account the boundary conditions and calls the subroutines for the system reduction described in Section 3.2. Successively, the reduced system is solved by the iterative procedure `TRESNEI` described in Section 2.3 which gives rise to the inner iteration. Once a sufficiently accurate approximation to the solution is found, convergence of outer iterations is tested. If convergence in the outer iteration is not declared, the current values of the flow properties are possibly relaxed and used as initial guesses for the successive outer iteration. The outer termination criteria and the relaxation procedure are implemented as follows. Let x_0^i be the starting guess for `TRESNEI` at the i-th outer iteration and x_K^i be the corresponding computed solution. Moreover, let $\bar{x}_K^i$ be the average value of x_K^i computed (componentswise) over the all network. Then, the outer iteration is stopped if

$$\|(x_K^i - x_0^i)/\bar{x}_K^i\|_\infty \leq \epsilon_3$$

where the ratio is meant componentwise and $\epsilon_3 > 0$ is a prescribed tolerance. Moreover, the "relaxed" starting point is defined by

$$x_0^{i+1} = x_0^i - C_{rel}(x_0^i - x_K^i),$$

with relaxation coefficient $C_{rel} \in [0, 1]$. If $C_{rel} = 1$, no relaxation is imposed on the computed solution. In the experiments described in Section 5 we set $\epsilon_3 = 10^{-6}$ and $C_{rel} = 1$.

4.1. The units' subroutines

Modularity of the code imposes similar formal structure to all units' subroutines, even if the physical transformations represented may differ. Indeed units' subroutines differ in the part where calculation of the physical transformations is carried out.

Figure 3 shows the structure of a generic unit's subroutine. At the first outer iteration, after the initial variables' declaration, the characteristic element's input data are read and stored for use at following iterations. Depending on the current iteration, flow properties' initial values are assigned; an initialization procedure is performed at the first iteration while the current approximation is used at the following ones. Successively, the element's calculations bind flow properties in each inlet/outlet port of the element for the represented physical transformations. Finally, the multiplicative and exponential coefficients of the bond's equations are evaluated.

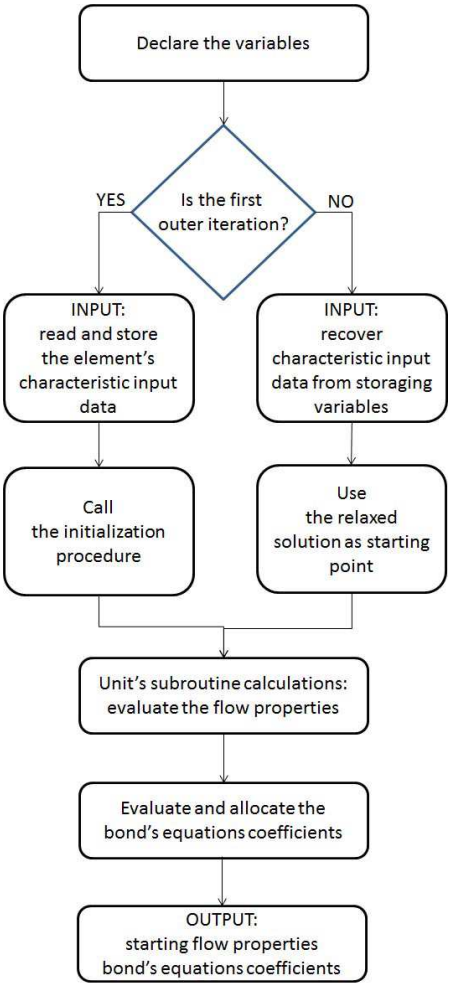

Figure 3: Unit's subroutine flow chart.

5. Code validation

The aim of this section is to make preliminary tests on the performance of the new code and compare it with the non-commercial code ESMS introduced in [1]. Tests were carried out on two compression train arrangements with both series and parallel-mounted compressors and illustrate the behaviour of our code under thermodynamic working condition. The two plants considered are shown in Figure 4 and 5 along with boundary conditions and operating parameters. Table 2 summarizes the system's parameters for the two problems.

The first plant is composed by 9 elements and 18 nodes (16 mass flow nodes and 2 power flow nodes). Starting with an initial number of 128 bond's equations and 220 unknowns, the system is reduced to 54 equations in 54 unknowns; such reduction of the unknown quantities is achieved by using boundary and continuity conditions and eliminating continuity equations. With our new tool, a feasible solution is found in 4 outer iterations; the number of inner iterations performed at each outer iteration is 157, 10, 4 and 2 respectively. On the other hand, a solution is found by `ESMS` in 15 outer iterations. We remark that, at each `ESMS`'s outer iteration, the nonlinear system is replaced with a simplified model represented by a linear system; then, the linear system is solved by the Gauss-Jordan Elimination and the solution obtained is refined inside the characteristic elements at the successive iterative step [1]. The loss of accuracy caused by the linearization of the original nonlinear system motivates the higher number of `ESMS`'s outer iterations with respect to our Trust-Region based approach. On an Intel i7-4770 processor, the performed computational time is below one second for both codes.

Results from the two codes in terms of physical properties in each node of the plant and plant's performance are given in Table 3, where bold numbers indicate boundary conditions (equal for both input data of the two codes). Variables *P, y air, y H2O* and ω are reported only once since they can be retrieved either from continuity relations or from trivial ones and are therefore equal for both codes. The table shows that the computed solutions differ slightly in terms of mechanical power and outlet temperature across the compressors; relative errors between these two quantities are smaller than 10^{-3}. This discrepancy is due to the different thermodynamic libraries used in the two codes for computing the specific heats involved into adiabatic compression equations which lead to outlet temperatures. Moreover, we observe that such a discrepancy affects the temperature values until the coolers; successively, coolers' pinch point temperature differences are fixed equal for both codes so that gas temperatures are forced to take the same values.

We remark that no information on enthalpy is obtainable with `ESMS`, as it is works with the two state variables temperature and pressure. It is therefore interesting to note that pressure drops across the lamination valve and the second splitter could only be modelled as simple variations on pressure with `ESMS`, as it

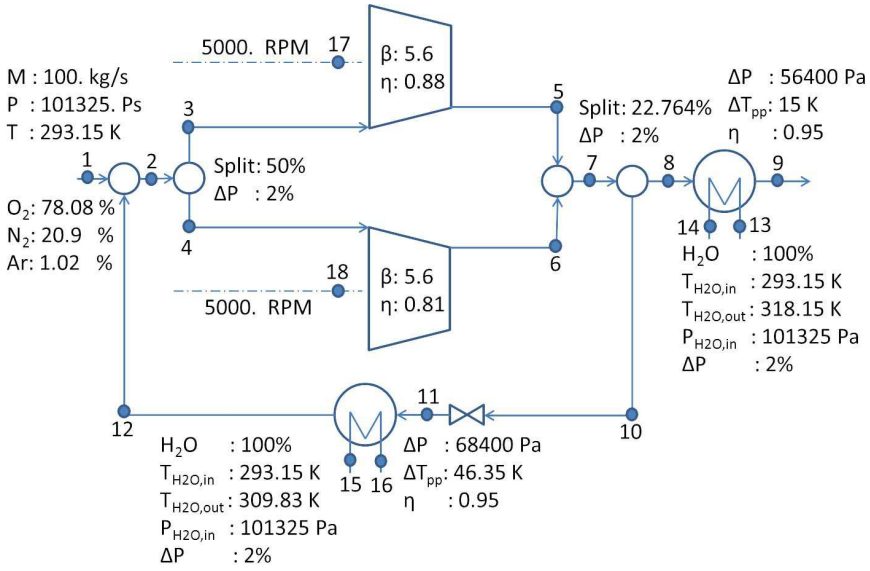

Figure 4: Parallel-mounted compression train plant.

	M [kg/s]	P [Pa]	T [K]	H [J/kg]
lower	0	100	180	$-2 \cdot 10^5$
upper bound	10^3	10^8	3000	$5 \cdot 10^6$

	Y [-]	W [W]	ω [rad/s]	Q [W]
lower bound	0	-10^9	0	-10^9
upper bound	1	10^9	$3 \cdot 10^4$	10^9

Table 1: Lower and upper bounds for the two train plants.

is unable to represent the effect that isenthalpic expansions have on temperature. On the contrary this effect is displayed by the new modular tool.

The second plant, Figure 5, is made up of 8 elements and 17 nodes (16 mass flow nodes and 1 power flow node). Our new modular code finds a feasible solution after 4 outer iterations and `TRESNEI` requires 77, 5, 2 and 2 respectively, inner iterations to reach convergence; `ESMS` takes 13 iterations to achieve the solution at the required accuracy. Analogously to the previous simulation, comparison of plant's results in terms of flow properties between the two codes are given in Table 4. Similarly to the previous test case, the slight differences on the computed temperatures and mechanical power can be ascribed to the different thermodynamic libraries used. Again, both codes takes less than one second to return the solution.

Overall, the agreement between the results obtained with the new tool and the well-assessed ESMS indicates that the code presented in this paper is reliable and efficient on the proposed test cases.

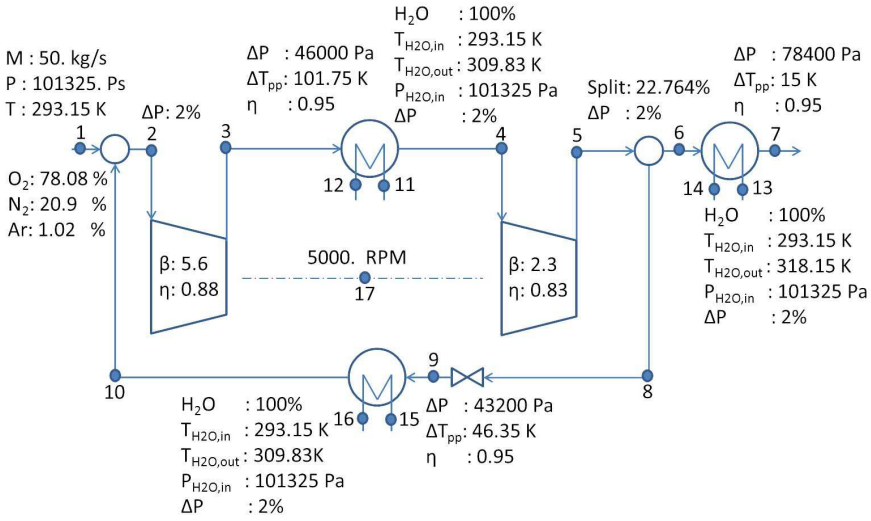

Figure 5: Series-mounted compression train plant.

	parallel-mounted plant	series-mounted plant
# of elements	9	8
# of nodes	18	17
# of bonds equations	128	118
# of unknowns	220	204
# internal continuity equations	64	64
# of external continuity equations of mass flows	80	64
# of external continuity equations of power flows	0	1
# of boundary conditions	22	29
# of redundant P-T-H equations	10	8
# of independent equations	54	46
# of independent variables	54	46

Table 2: System's parameters for the two plants.

Node	*M* *[kg/s]*	*M* *[kg/s]* ESMS	*P* *[Pa]*	*T* *[K]*	*T* *[K]* ESMS	*H* *[kJ/kg]*	*y air*	*y H2O*	*W* *[kW]*	*W* *[kW]* ESMS	*ω* *[rad/s]*
1	**100.00**	**100.00**	**101325**	**293.15**	**293.15**	271.68	**1**	**0**	-	-	-
2	129.47	129.47	101325	303.73	303.73	281.58	1	0	-	-	-
3	64.74	64.74	99298	303.73	303.73	281.58	1	0	-	-	-
4	64.74	64.74	99298	303.73	303.73	281.58	1	0	-	-	-
5	64.74	64.74	556072	516.38	515.94	485.19	1	0	-	-	-
6	64.74	64.74	556072	534.14	533.73	502.79	1	0	-	-	-
7	129.47	129.47	556072	525.27	524.84	493.99	1	0	-	-	-
8	100.00	100.00	544950	525.26	524.84	493.99	1	0	-	-	-
9	100.00	100.00	488550	308.15	308.15	284.84	1	0	-	-	-
10	29.47	29.47	544950	525.26	524.84	493.99	1	0	-	-	-
11	29.47	29.47	169725	525.00	524.84	493.99	1	0	-	-	-
12	29.47	29.47	101325	339.50	339.50	315.17	1	0	-	-	-
13	190.2	189.77	**101325**	**293.15**	**293.15**	84.01	**0**	**1**	-	-	-
14	190.2	189.77	99298	**318.15**	**318.15**	188.52	0	1	-	-	-
15	71.98	71.92	**101325**	**293.15**	**293.15**	84.01	**0**	**1**	-	-	-
16	71.98	71.92	99298	**309.83**	**309.83**	153.75	0	1	-	-	-
17	-	-	-	-		-	-	-	-13181	-13222	**523.6**
18	-	-	-	-		-	-	-	-14320	-14364	**523.6**

Table 3: Parallel-mounted plant's results in each node

Node	*M* *[kg/s]*	*M* *[kg/s]* ESMS	*P* *[Pa]*	*T* *[K]*	*T* *[K]* ESMS	*H* *[kJ/kg]*	*y air*	*y H2O*	*W* *[kW]*	*W* *[kW]* ESMS	*ω* *[rad/s]*
1	**50.00**	**50**	**101325**	**293.15**	**293.15**	271.68	**1**	**0**	-	-	-
2	64.74	64.74	99298	303.73	303.73	281.58	1	0	-	-	-
3	64.74	64.74	556072	516.38	515.94	485.19	1	0	-	-	-
4	64.74	64.74	510072	394.90	394.9	367.13	1	0	-	-	-
5	64.74	64.74	1173165	517.47	517.19	485.83	1	0	-	-	-
6	50.00	50.00	1149701	517.47	517.19	485.83	1	0	-	-	-
7	50.00	50.00	1071301	308.15	308.15	284.84	1	0	-	-	-
8	14.73	14.73	1149701	517.47	517.19	485.83	1	0	-	-	-
9	14.73	14.73	144525	516.72	517.19	485.83	1	0	-	-	-
10	14.73	14.73	101325	339.50	339.50	315.17	1	0	-	-	-
11	104.2	104.15	**101325**	**293.15**	**293.15**	84.01	**0**	**1**	-	-	-
12	104.2	104.15	99298	**309.83**	**309.83**	153.75	0	1	-	-	-
13	91.57	91.44	**101325**	**293.15**	**293.15**	84.01	**0**	**1**	-	-	-
14	91.57	91.44	99298	**318.15**	**318.15**	188.52	0	1	-	-	-
15	34.33	34.43	**101325**	**293.15**	**293.15**	84.01	**0**	**1**	-	-	-
16	34.33	34.43	99298	**309.83**	**309.83**	153.75	0	1	-	-	-
17	-		-	-		-	-	-	-20866	-20927	**523.6**

Table 4: Series-mounted plant's results in each node

6. Conclusions

This paper presents a new modular approach for industrial plant simulation. Special emphasis has been put on the mathematical model which consists of a constrained nonlinear system of equations and on its numerical solution.

The modular code introduced is capable of simulating industrial plant configurations, irrespective from input data (provided they are consistent), and plant's components (as long as they are present within the components' library). The former feature is guaranteed by the solver for the mathematical problem. The nonlinear system of equations to be solved describes thermo-fluid dynamic and mechanical processes that take place within each element of the plant and are known as bond's equations. After a proper simplification, the system is solved iteratively by the solver `TRESNEI` with a parallel/full implicit mode; the equations are solved simultaneously and an accurate hint for the solution is not required. The latter feature is enforced by modeling plant's elements through independent subroutines from a components' library. This allows maximum flexibility and expandability; the operating range of the code can be easily increased by adding new kind of elements.

In order to validate the new modular tool against thermodynamic simulations and to state its reliability, two compression trains have been thermodynamically simulated and the results obtained by our code have been compared with those computed by an extensively tested pre-existing modular code. Further development of the code may include the implementation of tools for design and off-design cycle calculations of the plant's elements.

7. Acknowledgments

The work of the third and fourth author was partially supported by INdAM-GNCS, under the 2015 Projects "Regularization methods for optimization problems and applications" and "High order methods for large-scale composite optimization problems".

Appendix A. Nomenclature

[1] C. Carcasci and B. Facchini. A numerical method for power plant simulation. *ASME Journal of Energy Resources Technology*, 118:36–43, 1996.

[2] E. Perz. A computer method for thermal power cycle calculation. *ASME Journal of Engineering for Gas Turbine and Power*, 113:184–189, 1991.

[3] SimTech Simulation Technology. *User Documentation: Program Modules and Model, IPSEpro Process Simulator*, 2003.

[4] E. Thorbergsson, T. Grnstedt, and C. Robinson. Integration of fluid thermodynamic and transport properties in conceptual turbomachinery design. In *Proceedings of ASME Turbo Expo 2013: Turbine Technical Conference and Exposition*, 2013.

[5] C. Carcasci, B. Facchini, and S. Harvey. Modular approach to analysis of chemically recuperated gas turbine cycles. *Energy Conversion and Management*, 39:1693–1703, 1998.

[6] C. Carcasci, F. Costanzi, B. Facchini, and B. Pacifici. Performance analysis in off-design condition of gas turbine air-bottoming combined system. *ATI 2013, 68th Conference of the Italian Thermal Machines Engineering Association; Elsevier Energy Procedia*, 45:1037–1046, 2014.

[7] C. Carcasci and N. A. Colitto Cormacchione. Part load operating strategies for gas turbines in district heating applications. In *Proceedings of the Institution of Mechanical Engineers, Journal of Power and Energy*, volume 215, pages 529–544, 2001.

[8] M. Falcetta and E. Sciubba. A computational, modular approach to the simulation of power plants. *Heat Recovery Systems CHP*, 15:131–145, 1995.

[9] P. Fiorini and E. Sciubba. Modular simulation and thermoeconomic analysis of a multi-effect distillation desalination plant. *Energy*, 32:459–466, 2007.

[10] Altran Italia. *Technology Review n. 4*, 2010.

[11] R. Carapellucci and G. Cau. Un sistema di simulazione modulare per la valutazione delle prestazioni dei sistemi energetici. In *IV Covegno Nazionale Gruppi Combinati Prospettive Tecniche ed Economiche*, volume 215, pages 237–250, 1992.

[12] K. Volkov. *Efficiency, Performance and Robustness of Gas Turbines*. InTech, 2012.

[13] Thermoflow Inc. *Thermoflow 15*. Sudbury, MA, USA, release 1 edition, 2010.

[14] M. Bagnoli, M. Bianchini, F. Melino, A. Peretto, P. R. Spina, R. Bhargava, and S. Ingistov. A parametric study of interstage injection on ge frame 7ea gas turbine. In *Proceedings of ASME Turbo Expo 2004*, volume GT2004-53042, pages 489–499, 2014.

[15] E. M. A. Mokheimer, Y.N. Dabwan, M. A. Habib, S. A. M. Said, and F. A. Al-Sulaiman. Techno-economic performance analysis of parabolic trough collector in dhahran, saudi arabia. *Energy Conversion and Management*, 86:622–633, 2014.

[16] GE Enter Software, LLC. *GateCycle: Combined-Cycle Design and Thermal Performance Modeling Software*. Sudbury, MA, USA, release 1 edition, 2012.

[17] K. Wan, S. Zhang, J. Wang, and Y. Xiao. Performance of humid air turbine with exhaust gas expanded to below ambient pressure based on microturbine. *Energy Conversion and Management*, 51:2127–2133, 2010.

[18] E. Dahlquist D. Haggstahl. Evaluation of prosim and ipsepro, two heat and mass balance simulation softwares. In *Conference Proceedings of SIMS*, 2003.

[19] S. Mattsson and H. Elmqvist. Modelica an international effort to design the next generation modeling language. In *7th IFAC Symposium on Computer Aided Control Systems Design, CACSD 97*, pages 1–5, 1997.

[20] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. John Wiley and Sons, 2004.

[21] J. W. Bush, X. Liu, H. E. Khalifa, and R. T. Drost. Compressor simulation: a modular approach. In *Purdue University, International Compressor Engineering Conference*, pages 701–706, 1998.

[22] L. Coco-Enriquez, J. Munoz-Anton, and J. M. Martinez-Val. Innovations on direct steam generation in linear fresnel collectors. In *19th SolarPACES Conference*, 2013.

[23] M. Tiller, P. Bowles, H. Elmqvist, D. Bruck, S. E. Mattson, A. Moller, H. Olsson, and M. Otter. Detailed vehicle powertrain modeling in molelica. In *Proceedings of Modelica Workshop 2000*, pages 167–178, 2000.

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[25] S. Bellavia, M. Macconi, and B. Morini. An affine scaling trust-region approach to bound-constrained nonlinear systems. *Applied Numerical Mathematics*, 44:257–280, 2003.

[26] S. Bellavia, M. Macconi, and B. Morini. STRSCNE: A scaled trust-region solver for constrained nonlinear equations. *Computational Optimization and Applications*, 28:31–50, 2004.

[27] S. C. Eisenstat and H. F. Walker. Globally convergent Inexact Newton methods. *SIAM J. Optim.*, 4:393–422, 1994.

[28] M. Macconi, B. Morini, and M. Porcelli. Trust-region quadratic methods for nonlinear systems of mixed equalities and inequalities. *Applied Numerical Mathematics*, 59:859–876, 2009.

[29] B. Morini and M. Porcelli. TRESNEI, a Matlab trust-region solver for systems of nonlinear equalities and inequalities. *Computational Optimization and Applications*, 51:27–49, 2012.

[30] M. Pernice and H. F. Walker. Nitsol: A newton iterative solver for nonlinear systems. *SIAM Journal on Scientific Computing*, 19(1):302–318, 1998.

[31] M. Porcelli. On the convergence of an Inexact Gauss-Newton trust-region method for nonlinear least-squares problems with simple bounds. *Optimization Letters*, 7:321–340, 2014.

[32] C. A. Floudas et al. *Handbook of test problems in local and global optimization.* Kluwer Academic Publishers, Nonconvex Optimization and its Applications, 33, 1999.

[33] T.F. Coleman and Y. Li. An interior trust-region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6:418–445, 1996.

[34] M. Macconi, B. Morini, and M. Porcelli. A Gauss-Newton method for solving bound-constrained underdetermined nonlinear systems. *Optimazion Methods and Software*, 24:219–235, 2009.

Latin letters

a	Exponential constant terms of temperature in bond's equations (4)
A	Multiplying constant terms of temperature in bond's equations (1)
b	Exponential constant terms of enthalpy in bond's equations (4)
B	Multiplying constant terms of enthalpy in bond's equations (1)
C	Exponential constant terms of pressure in bond's equations (4)
C	Multiplying constant terms of pressure in bond's equations (1)
D	Multiplicative constants of the mass flow properties of pressure loss equations (bond's equations (3))
e	Exponential constants of the mass flow properties of bond's equations (1) and (2)
E	Species concentrations' exponential constants in bond's equations (2)
H	Enthalpy [kJ/kg]
k	Multiplicative constants of the mass flow properties of bond's equations (1) and (2)
K	Species concentrations' multiplying constants in bond's equations (2)
k_{known}	Right-hand sides in the bond's equations
M	Mass flow [kg/s]
N	Total number of elements
n_f	Number of mass flow ports of each element
n_h	Number of heat flow ports of each element
n_p	Number of power ports of each element
N_{BC}	Total number of boundary conditions of the problem
N_{BE}	Number of bond's equation of each element
N_{CE}	Total number of continuity equations
N_M	Number of mass flow ports of each element
N_Q	Number of heat flow ports of each element
N_S	Number of considered chemical species
N_W	Number of mechanical connections of each element
P	Pressure [Pa]
Q	Heat flow [W]
T	Temperature [K]
W	Mechanical power [W]
y	Mass/molar fraction

Greek letters

ϵ	Exponential constant terms of power and heat flow properties of bond's equations (1)
η	Exponential constants of the mass flow properties of pressure loss equations (bond's equations (3))
κ	Multiplying constant terms of power and heat flow properties of bond's equations (1)
ω	Rotational speed [rad/s]

Table A.5: Nomenclature