Università degli Studi di Firenze

DOTTORATO DI RICERCA IN
"*Energetica a Tecnologie industriali innovative*"

CICLO XXV

COORDINATORE Prof. Maurizio De Lucia

Optimization of an ejector refrigeration cycle

Settore Scientifico Disciplinare ING/IND10

Dottorando
Dott. Dario Paganini

_____
(firma)

Tutore
Prof. Giuseppe Grazzini

_____
(firma)

Anni 2010/2012

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Present work proposes the optimization of an ejector refrigeration plant using some standard and new design techniques.

The project has been developed in the framework of a collaboration between Department of Industrial Engineering of the University of Florence and the industrial partner Frigel Firenze Spa. Starting from experience of the research group in the field of ejector refrigeration, a prototype plant has been designed and then realized at the test facility set up in manufacturer plant of Frigel Firenze Spa.

Optimization is performed considering the refrigeration plant as a whole, including plate heat exchangers, pump and ejector. This represents the main component of the plant and design techniques are proposed to model it. Specifically, ejector geometry definition is based on a literature method (*CRMC* method), which seems to ensure better performance of the component and which is improved with some innovative features.

Starting from optimization program results and proposed design criteria, the prototype plant has been realized and experimental tests performed. Plant nominal cooling power is 40 kW which is considered adequate for industrial temperature control application.

Description of experimental apparatus realization and test are reported and results are analysed, comparing them with typical

ejector refrigeration system behaviour and design choices. Based on experimental results, reliability of proposed models and design techniques is verified.

Chapter 2

Ejector refrigeration

Constantly increasing refrigeration demand for food conservation, air conditioning and other needs is one of the main issues in the energy consumption inventory. In many cases, substitution of electricity with heat, as energy input for refrigeration systems, is advantageous. This may be the case, for example, of air conditioning needs, which are quite closely matched in time with solar energy availability. Another significant case is combined heat and power generation (CHP), which may benefit of increased plant use if waste heat is converted in cooling capacity during summer, when heat loads can be null or drastically reduced. All these cases can benefit from three-thermal systems, i.e. devices consuming heat (instead of work) as "compensation factor", requested by Second Law of Thermodynamics, to allow heat transfer from a cold source to the ambient. Among three-thermal systems, absorption cooling is by far the established and increasingly competitive option. However, the market for three-thermal systems is wide and hopefully due to increase in the near future. Hence further options may be useful and deserve research and development effort. Among various alternatives, ejector refrigeration can play a significant role.

2.1 Historical background

Ejectors, as mechanical devices, have a long history. They were used to create vacuum in train braking systems since the 19th century. In power stations, they are used for removing non-condensable gasses from condenser. Nowadays, ejectors are commercially available in a wide range of configurations, either single or multistage [1]. Use of ejectors in thermal systems have been developed since the early 20th century. In 1910, Maurice Leblanc proposed and built the first steam ejector refrigeration machine driven by heat, although only air conditioning application could be realized. It is interesting that he used water as refrigerant in the steam ejector refrigeration cycle [27]. Around 1930, they were applied to air conditioning on trains and in buildings.

Ejector theory based on one-dimensional ideal gas model was first introduced by Flügel, who applied equations of continuity, momentum, and energy to the design of ejectors ([18],[17]). Same method of analysis of ejectors is used by Keenan *et al.* ([36], [35]), who extended it to different geometries and pressure conditions and compared its results with experimental ones. Their model showed good agreement with experimental data, though neglecting friction losses, and has been used as theoretical basis in ejector design for the past fifty years [7]. Work of Munday and Bagster [40] proposed, about a quarter of a century later, a model which considered primary and secondary flow distinct and postulated secondary flow reach sonic velocity at some cross section of the ejector. This model permitted to explain the constant cooling capacity of ejector refrigerators.

[1] see for example Schutte & Koerting, http://www.s-k.com (accessed 16.12.2012)

2.2 State of the art

Specific advantages of ejection refrigerators are mechanical simplicity, ability to work with "environmental friendly" refrigerants, low investment cost and quite low temperatures of the hot source [15].

However, some relevant drawbacks have to be taken into account when it is compared with competing systems. First, the working fluid in an ejection cycle has to withstand a heat engine and a refrigeration cycle. This brings the fluid along a wide range of temperatures, making fluid selection not easy. The safest and cheapest fluid, water, has been used in the early efforts and extensively studied thereafter ([2],[21]), but it is now less common, a part few works like [45], due to water thermodynamic problems (high specific volume at condenser, very low pressure at evaporator, nucleation process in primary nozzle [23]). Fluid charge is much higher in comparison with vapour compression refrigerators, which makes the use of high cost or dangerous fluids impractical. Moreover, liquid fluid must be pumped from the condenser exit to the vapour generator by a feed pump: this introduces a component which is not familiar for refrigeration experts. The feed pump is prone to cavitation and, for certain working fluids, must be able to deliver fluid at high pressure. Flow rates are small, but absorbed power can be quite high if efficiency is low. It is worth noting that the pump absorbs electric energy instead of low grade heat. Finally efforts to obtain higher COP values have to be done, to make this refrigeration system comparable with the absorption cycle in term of performance.

2.2.1 Modelling techniques

Nowadays, Computational Fluid Dynamics (CFD) is seen as indispensable tool for the verification and eventual optimization of an ejector [3]. It can be used to verify reliability of parameters used in one-dimensional model ([48],[55]) and to validate design strategies [44]. At the same time, turbulence models need to be further

examined to make simulation results more reliable. At high primary pressures, in fact, same ejector performances are predicted with different feature of local flow, using different turbulence models ([29],[30]).

Though entrainment performance is mainly due to secondary nozzle design and care has to be taken to avoid shocks at its exit [3], no specific design criteria are proposed for secondary nozzle and mixing chamber. All cited works, in fact, simulate ejector behaviour considering standard component design. Specifically, a convergent inlet section is provided for the secondary flow, which is connected to a divergent stretch by a cylindrical duct. No simulations are present in literature to evaluate behaviour of ejector designed with different approaches, like *CRMC* method adopted in this work.

Another scarcely addressed feature in CFD simulations is use of real gas models. These result particularly appropriate for modelling refrigerant fluids, while some recent simulation works prefer to use air, with a perfect gas model, to easily obtain experimental data for validation [29].

CFD simulations, however, requires a pre-defined geometry and the main geometric features of the ejector, as well as of other components of the refrigerator, may be calculated, in first instance, only by a one-dimensional procedure and on the basis of designer experience.

As said, ejector is usually realised coupling supersonic nozzle with a constant-area section cylindrical duct, along which primary and secondary flux flows toward a diffuser and then to the condenser of the refrigeration plant. Depending on the position of the nozzle referred to constant-area section, ejector design strategies can be divided into two types. Constant-pressure mixing model is used if primary nozzle exit plane is located within the suction chamber, in front of constant-area section, while constant-area mixing model is considered, if nozzle exit is within the cylindrical section. The first seems to provide better performance, while the latter gives better performance prediction. Almost all current study has been focused

on the constant-pressure mixing ejector.

Starting from historically proposed model (§2.1), several updated models have been proposed, until recently, to take into account for friction losses by isentropic efficiency of primary and secondary nozzles and mixing process efficiency with ideal gas assumption ([32],[59]), while others used real gas model to make simulated model more realistic and improving accuracy [42].

All different mathematical models, proposed to evaluate performance of ejector for several operation and design conditions, are basically expressed as explicit algebraic equations and based on steady one-dimensional assumption. These models do not take into account for detailed local interaction between shock waves and boundary layers and their influence on mixing and compression rates. Isentropic efficiencies are usually added to accounting for friction losses and pressure loss, to improve models reliability [7], but wide range of their values are proposed in literature, depending on ejector geometries and operating conditions ([32], [14], [47], [16]).

Main assumptions made in literature for development of ejector models are:

- Inner wall of the ejector is adiabatic
- Flow inside the ejector is steady and one-dimensional
- Inlet an outlet velocity of primary, secondary and mixed flow are considered low and neglected
- Primary and secondary flow start to mix with uniform pressure at the mixing section
- Friction losses are taken into account with isentropic efficiencies

While, as said, some solutions are proposed to improve one-dimensional model capacity to match real ejector behaviour, only one different design strategy for mixing chamber and diffuser is found in literature, proposed by Eames [12]. He suggested to design

mixing chamber and diffuser using a new criterion, defined *Constant Rate of Momentum Change* (CRMC), which imposes a constant rate of variation of velocity along diffuser profile to obtain its geometric description. This method ensures better performance, in term of ejector entrainment ratio, compared to standard design approaches, as results from experimental analysis performed on small scale prototype [13].

Though a quite large amount of works have been done on modelling and analysing ejector behaviour, further efforts seems to be still necessary [28]

1. to study the influence of isentropic coefficients which are taken as constant in the models and their impact on ejector performance

2. to develop a simulation and design procedure, for the whole refrigeration system, which blends ejector models with the other component of the system

3. to improve, for numerical simulation approach, accuracy of turbulence models

By previous analysis, it can be pointed out that there are still some efforts to make to obtain a funded modelling framework for ejectors applied in refrigeration plants, specially for evaluation of impact of ejector design on plant performance and its interaction with other plant components.

2.3 Working fluid analysis

Starting from the fundamental work [11], which used R11, R22, R114, R123, R133a, R134a, R141b, R142b, R152a, RC318, R141b e RC318, many authors considered pure fluids as well as azeotropic or non-azeotropic mixtures as candidate fluids for ejector refrigerators. Sun [53] compare performances of ejector refrigeration systems

operating with eleven different refrigerants used as working fluids including water, halocarbon compounds (R11, R12, R113, R21, R123, R142b, R134a, R152a), a cyclic organic compound (RC318) and an azeotrope (R500). For given operating conditions, optimum ejector area ratios and COPs are computed for each fluid, concluding that ejectors using environment-friendly HFC refrigerants R134a abd R152a perform very well and pattern of performance variation for different refrigerants is almost independent of chosen operating conditions. Cizungu et al. [9] simulate a vapour jet refrigeration system using one-dimensional ideal gas model based on mass, momentum and energy balances for different environment-friendly refrigerants. Theoretical model validation is carried out by using R11 and then COPs are obtained for four fluids (R123, R134a, R152a, R717), given different operating conditions and area ratios. By comparing results, all fluids show similar performance characteristics, depending on the ejector area ratios and operating conditions and refrigerants R134a and R152a should be used with high area ratios for low grade heat source to achieve better COP. Five refrigerants (R134a, R152a, R290, R600a, R717) are considered by Selvaraju and Mani [49] and performance of the ejector refrigeration system are achieved by a computer simulation, based on one-dimensional ejector theory and mass, momentum and energy balances, while thermodynamic properties of fluids are obtained using REFPROP libraries. Pattern of variation of critical ejector area ratio, critical entrainment ratio and critical COP are similar and better performance are found for R134a. Using empirical correlation from literature [11], comparison of COPs for solar powered ejector refrigeration system, operating with eight different working fluids, is done by Nehdi et al. [41], mainly focusing on system overall efficiency and obtaining best performance for R717 as refrigerant. Two fluids (R142b, R600a) are considered by Boumaraf and Lallemand [5] to analyse refrigeration system operating in design and off-design conditions, using ideal gas and constant-area mixing models and including two-phase heat exchangers. They conclude that ejector refrigerating system operat-

ing with R142b gives better performance due to its higher molecular weight compared to R600a.

Abdulateef et al. [1] use R11, R12, R113, R123, R141b, R134a and R718 (water). Unfortunately many fluids are no longer usable (or will be phased out soon) within the current regulatory framework. Petrenko [43] asserts that refrigerants offering low environmental impact, good performance and moderate generator pressure, are R245fa, R245ca, R600 e R600a. The last feature makes feed pump selection easier.

2.4 Ejector refrigeration cycle

Ejector refrigeration system is based on three heat source refrigeration cycle, which uses high pressure vapour, generated supplying thermal power to the fluid in the generator, and then accelerated through a supersonic nozzle, to extract secondary fluid from evaporator. Primary and secondary flow mix and gain pressure in the diffuser until condenser is reached, where vapour condenses. Liquid is then partly pumped to the generator and partly returned to the evaporator through an expansion valve. Cooling power is obtained by using quite low temperature hot source, usually from 300°C down to 100°C, and only electrical power needed by a feed pump, generally two orders of magnitude lower than thermal power.

A general scheme of ejector refrigeration system is shown in fig.(2.1).

The hot fluid may come from various kinds of thermal source having temperature lower than 300°C, like waste heat from internal combustion engines and turbines or solar energy. Generator, evaporator and condenser may be any kind of heat exchanger. Refrigerant vapour exits generator at maximum cycle temperature and expands in the primary nozzle until supersonic condition. Supersonic expansion establishes low pressure condition at primary nozzle exit, which permits secondary flow to be evaporated and drawn toward mixing chamber. Cold fluid at evaporator pressure enters the secondary

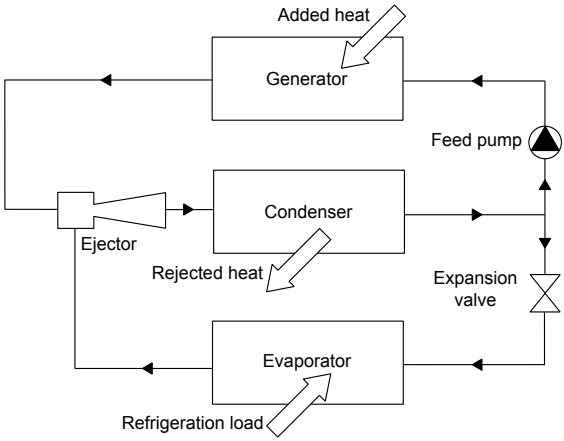

Figure 2.1: Scheme of operating cycle of ejector refrigerator

channel and then expands until sonic conditions. Here primary and secondary streams have the same pressure, even if temperature and velocity are different. Mixing process starts and it is ideally completed at mixing chamber outlet, where two streams are mixed in a single homogeneous flow. After mixing process, flow is still supersonic, so convergent-divergent supersonic diffuser is required to slow down it and to recover pressure, in order to reach the condenser conditions.

Condensed fluid exits the exchanger and reaches the feed pump. Subcooling and adequate head may be necessary to avoid cavitation problems at pump inlet. The pump sends part of the liquid back to the generator, closing the "thermal engine" sub-cycle. Other part of the fluid is sent to an expansion valve, where it expands at constant enthalpy down to evaporator pressure.

Evaporation is the useful phase of the cycle as in any other refrigeration cycle. From the evaporator, the fluid is brought back to the high pressure section of the system by the ejector. Path through ejector and condenser is common between "thermal engine" and "refrigerator" sub-cycles.

Once temperatures of hot, cold and ambient fluids are known, allowing some temperature difference in heat exchangers, the enthalpy differences in the evaporator, generator and condenser are substantially fixed. Therefore, the system performance is strictly related to the entrainment ratio, i.e. the ratio between secondary and primary mass flow rate, which depends on ejector design and configuration.

Typical thermodynamic cycle for ejector refrigerator is shown in fig.(2.2) (Ts diagram).

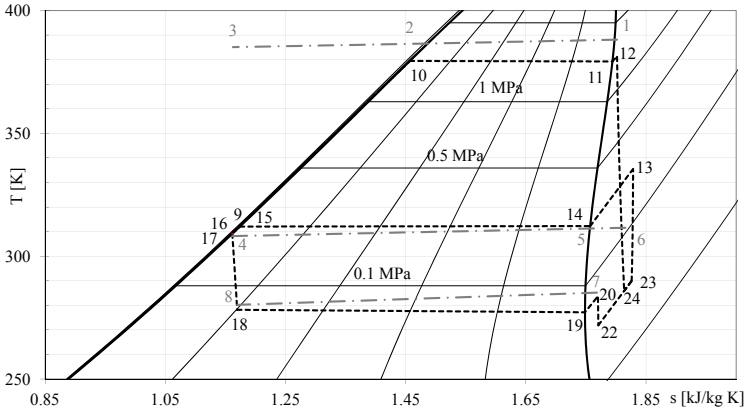

Figure 2.2: Typical ejector refrigeration cycle on Ts diagram

2.5 Working fluid selection

A seen in §2.3, many refrigerants are considered in literature. Unfortunately, fluid selection is made difficult by continuously updating of regulatory framework, which impose limitation to use of halocarbon compounds.

Main favourable features for ejector refrigerator working fluid are:

- Critical temperature well below generator temperature

- High latent heat at evaporator temperature, to increase specific cooling capacity
- Normal boiling pressure close to ambient pressure, to avoid vacuum condenser
- Moderate generator pressure, to make feed pump selection easier and to reduce safety costs

A further interesting feature is the slope of the saturation curve on Ts diagram. Some fluids have a positive slope of the curve (fig.(2.3)). Therefore, any expansion starting from saturated vapour (unless too close to critic point) will stay on the right of saturation curve, even if entropy increase is very low, avoiding liquid condensation within the ejector. Other fluids, in order to guarantee a dry expansion, require vapour superheating.

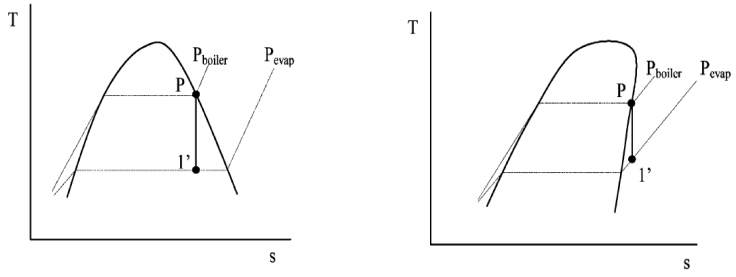

Figure 2.3: Typical Ts diagram for wet (left) and dry (right) fluid [7]

When environmental impact is considered, all ozone depleting fluids should be avoided. Flammable or toxic fluids are problematic as well. Furthermore, the *Global Warming Potential* (GWP) should be accounted for. A few useful data in order to address all these issues are summarized in tab.(2.1).

R365mfc (1,1,1,3,3-pentafluorobutane) has the highest critical temperature and a positive slope saturation curve, though its GWP value is quite high. Immediately below R365mfc in terms of critical

temperature are R245ca and R245fa. Both fluids have a positive slope saturation curve. R245ca has a lower GWP then R245fa and R365mfc. R41 (fluoromethane) has the lowest GWP, but has a low critical temperature, claiming for a supercritical vapour generator. It is possible in principle, but seems scarcely practical at present. R152a (1,1-difluoroethane) has also a low GWP and a more adequate critical temperature. Very high GWP values suggest excluding R143a, R218, R227ea, R236fa e RC318. R134a (1,1,1,2-tetrafluoroethane) and R236ea (1,1,1,2,3,3-hexafluoropropane) are both eligible. The second has a positive slope saturation curve. R134a has a reasonable cost (7 euros/kg) and is well known and accepted in the refrigeration industry. This means for example that problems of material compatibility have been solved. However, it has a relatively high GWP and a high saturation pressure at generator temperature. Another fairly known fluid is R245fa which has a growing diffusion in Organic Rankine Cycles (ORC). Its cost is high (27 euros/kg), but generator pressure may be reduced to one fourth with respect to R134a and superheating is unnecessary. Everything considered, the best compromise seems to be R245fa, henceforth used in calculations. However the proposed calculation tools can be easily modified in order to evaluate R365mfc, R152a, R236ea or any other fluid, as far as its thermodynamic properties are reliably characterized.

Fluid	$T_{crit}[K]$	$P_{crit}[MPa]$	N.B.P.	Expansion	GWP
R134a	374.21	4.0593	247.08	wet	1300
R143a	345.86	3.761	225.91	wet	4300
R152a	386.41	4.5168	249.13	wet	120
R218	345.02	2.64	236.36	dry	8600
R227ea	374.9	2.925	256.81	dry	3500
R236ea	412.44	3.502	279.34	dry	1200
R236fa	398.07	3.2	271.71	dry	9400
R245ca	447.57	3.925	298.28	dry	640
R245fa	427.16	3.651	288.29	dry	950
R32	351.26	5.782	221.5	wet	550
R365mfc	460.0	3.266	313.3	dry	890
R41	317.28	5.897	194.84	wet	97
RC318	388.38	2.7775	267.18	dry	10000

Table 2.1: Relevant features of possible refrigerants

Chapter 3

Cycle optimization

Ejector refrigeration plants, as other three-heat sources refrigeration systems, are subjected to several constraints represented by working temperatures and fluids, heat exchangers type and size, thermal source typology and ejector geometry. This last feature is dealt in detail with the next chapter having ejector geometrical characteristics main impact on plant performance and being a specific design method proposed. In present chapter, procedure to select optimal configuration of the key parameters of ejector plant is described and its results summarized. When several parameters can affect plant performance, it is often not possible to identify a single key parameter, whose influence on the system behaviour prevails over all other ones. In this case, it is then necessary to define strategies to take into account for mutually interactions between parameters and their influence on performance. Optimization procedures have to be used to the aim of finding better performance, subject to boundary constraints.

An optimization program has been realized, based on a previously developed program, used for optimization of an ejector refrigeration system working with water as refrigerant and conceived to produce ice slurry. Many adjustments have been done to take into account for different characteristics of the proposed plant, which

is designed to supply refrigerated water for industrial application, working with halocarbon compounds. Main differences concern with

- use of three plate heat exchangers to take into account for the necessity of distinct circuits for the refrigerant fluid and water used to heat and condense fluid itself
- use of specific functions to compute thermodynamic properties of refrigerant along the cycle

First aspect has to be taken into account modelling exchangers behaviour or characterizing them, and evaluating their heat transfer coefficients and pressure drops for specified working temperatures and mass flow rates of fluid and water. Refrigerant properties have been computed using REFPROP libraries [38], developed by NIST, which ensure high accuracy and wide fluids database.

3.1 Optimization concept

Many complex decision or allocation problems can be approached using optimization. In this way, it is in fact possible to define criterion to guide complex decision problem, involving choose of values for several correlated parameters, by identifying a single objective function, which quantify performance [39]. This objective function can then be maximized or minimized depending on problem formulation, using specific mathematical algorithms and eventually subject to some constraints. Of course, like all other field where modelling is required, choose appropriate and accurate description of constraints and objective in terms of physical and mathematical model is necessary to obtain reliable results. On the other hand, it is however useful to reduce model complexity, avoiding time-consuming computer applications, as possible as reliability is preserved. As a consequence of models and parameters use, optimization results have to be considered as approximation of real behaviour, being the goodness of the results themselves dependent of skill in modelling essential features of a problem.

Non linear relations between parameters imply that changing one variable at time (*OVAT* method) to find optimal solution involves loose of information about variables interactions and correlations [19]. Multivariate optimization method, described in following section, is used in present analysis to better identify optimal solution over a wide range of variables values. To apply optimization approach to ejector refrigeration plant design, this last feature of the method is particularly suitable, permitting it to explore a wide range of possible design solutions in a field where standardization is not yet reached.

Finally, it must be considered that objective function choice strongly influences optimization results [26], representing its value the base on which all other parameters are modified. Any choice is admissible for objective function, as long as its computational complexity remains reasonable, but it has to remember that results meaning is strictly connected to its definition.

3.1.1 Optimization algorithm

Main element of the program is the numerical optimization subroutine based on *Complex (Constrained simplex) method* proposed by Box [6] starting from *Simplex method* proposed by Spendley *et al.* [52]. Its aim is to search constrained maxima and minima of an objective function which, in our case, is represented by COP of the ejector refrigeration plant. The method ensures good performance and quite rapid convergence, when studied function presents several local maxima and minima in the region of interest. In particular it is conceived to find global maximum of functions when local extrema are present. Unlike other methods, *Complex method* searches for the maximum not only in the neighbour of starting point, but analysing a wide area of the region of interest, then guaranteeing the location of the global maximum is as independent as possible of the set of initial values.

Defining an objective function

$$f_{obj} = f(x_1, x_2, ..., x_n) \tag{3.1}$$

as a function of n independent variables x_i and subject to m constraints of the form

$$g_k < x_k < h_k \ , \ k = 1, ..., m \tag{3.2}$$

where $x_{n+1}, ..., x_m$ and the limiting values of the constraints g_k and h_k are function of $x_1, ..., x_n$, the method permits to find the maximum of the function f_{obj}. *Complex method* uses $k \geq n + 1$ points, each represented by a set of values for $x_1, .., x_n$ consistent with the constraints, only first of which is given, while all other $(k-1)$ points used during optimization process are defined by assigning random values, chosen fulfilling (3.2), to independent variables. Explicit constraints hence are satisfied, while each time implicit constraints have to be verified, eventually modifying assigned values of the independent variables to accomplish all conditions of (3.2). This is done moving inconsistent point closer to centroid iteratively until all constraints are satisfied.

Function is evaluated at each of the k points and the point corresponding to least value of the function itself is changed. It is replaced by a point which is $\alpha > 1$ times as far from the centroid defined by the other points as the reflection of the worst point respect to the centroid itself. The new point is then aligned with the removed one and the centroid of the retained points and it can be moved halfway towards that centroid if it still identifies the worst function value. Described procedure is repeated until some constraint is violated. In this case, if the point violates explicit constraint on independent variable $x_i \, (i = 1, ..., n)$ it is brought back just inside the overcome limit, while if some implicit constraint $x_j \, (j = n + 1, ..., m)$ is not satisfied the point is again moved halfway to the centroid. Points updating rules permit to always find an allowed point and to continue procedure until points are collapsed to the centroid.

Stopping criterion for the method is that the computed values of f_{obj} are identical (within defined tolerance) for a fixed number of procedure iteration, so permitting the program to proceed until there is any possibility to improve function value. Global maximum, instead a local one, is found if multiple repeats of the program with different initial points converge to the same solution.

Advantage of the method is its reliability to search maxima of functions, which present multiple local maximum in the region of interest thanks to randomly assigned initial points and repeated over-reflection.

3.2 Plant modelling

Program models include the whole system which is considered an open thermodynamic system exchanging thermal power with three heat sources trough corrugated plate heat exchangers (fig.(3.1)).

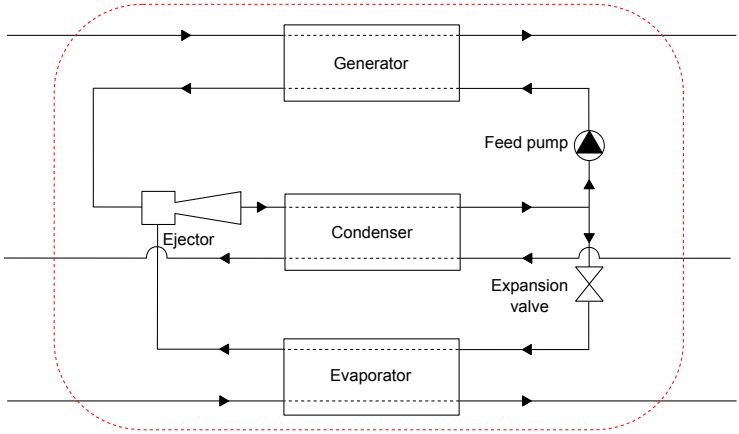

Figure 3.1: Thermodynamic system considered for the optimization

3.2.1 Working fluids properties

Use of refrigerants as working fluids for ejector refrigeration plant requires to find libraries able to determine their properties, which can be integrated in optimization program. NIST REFPROP libraries [38] are chosen to this aim. They supply subroutine to compute thermodynamic and transport properties of industrially significant fluids and their mixtures, with particular attention on refrigerants and hydrocarbons, based on most accurate and mixture models available in literature.

NIST REFPROP libraries are conceived to guarantee flexible use, being possible to explicitly call *Fortran* subroutines in developed program or use external calls to libraries. During this work, they are used first in the optimization program directly, including *Fortran* code in the program, and then though their interface with *Visual Basic for Application* in *Excel*.

Original optimization program consider water as working fluid and as thermal vector in the exchangers, without necessity to distinguish external circuits from refrigerant ones from the point of view of modelling techniques. This is instead necessary in the new version of optimization program, so water and refrigerant properties computing is modified to access NIST REFPROP libraries for the latter, while maintaining previous formulation for water.

All fluids considered in selection (§2.5) are present in NIST REFPROP libraries database, so their use permits great flexibility in refrigeration plant analysis.

Finally NIST REFPROP is constantly updated to include new fluids and models and this guarantees future improvement of the analysis.

3.2.2 Heat exchangers

Original program models steam generator as tube and shell heat exchanger, evaporator as flash evaporator and condenser as corrugated plate heat exchanger, according to steam ejector plant considered

[25]. New program is conceived to project optimized ejector refrigeration system working with halocarbon compounds for refrigeration cycle and water in external channels. It is then necessary to differentiate refrigerant circuit from water one and it is done by using three plate heat exchanger.

Two different approaches can be used to treat heat exchangers depending on what data are at disposal about their building and geometric characteristics and on which type of models are present in literature. If geometry and size of plate are known with an adequate level of accuracy, correlations can be found in literature which permits to compute heat transfer coefficient and pressure drops of the exchangers for some fluids. This method is very sensitive to geometry variation and fluid type and its main drawback is scarce flexibility to refrigerant changes, though it ensures finest setting capacity for mass flow rates, number of plates and temperature levels, whose values can be varied virtually continuously.

Unfortunately geometry of corrugated plate heat exchanger and plate characteristics is often patent pending and it is difficult to obtain all necessary information, with enough level of accuracy to model exchanger behaviour from the point of view of its heat transfer performance. In this case it is however possible to extract heat transfer coefficients and pressure drops estimated with software, that manufacturers make usually available, varying boundary conditions in the range of interest for the values of mass flow rates and temperatures. Moreover same type of heat exchanger is suitable for different fluids so allowing to simulate use of different refrigerants. Results can be finally summarized in tables read by the optimization program to vary working point at discrete steps.

Both described methods are implemented in the optimization program. Correlations are initially used to determine heat exchangers behaviour, while program has then been modified to read tables, obtained by manufacturer software, with the aim to overcome mismatching between computed results and tabulated data and the lack of literature correlations for less common fluids.

Heat exchangers correlations

Models to describe plate heat exchangers are yet quite limited, particularly for what concern phase change zones, and they are further strongly dependent of exchanger geometry (Chevron angle and plate corrugation) and fluid type. Even if models with general validity are not present, some detailed formulation for specific fluids and exchanger geometries can be found in literature. In particular, for R134a, initially selected as suitable for ejector plant application, correlations are presented in literature related to its behaviour during evaporation and condensation phases in corrugated plate heat exchangers, so it is possible to include them in the optimization program.

When these correlations are used to compute heat transfer coefficients and pressure drops, input data are represented by water mass flow rate, water inlet temperature and nominal pressure and number of plates as summarized in tab.(3.1). Superheating at generator outlet and evaporation temperature are further defined for generator and evaporator respectively.

Geometric characteristics of plates and exchangers are moreover provided to the program to complete information (tab.(3.2)).

Input data for plate heat exchangers	
m_w	Water side mass flow rate [kg/s]
$P_{w,nom}$	Water side nominal pressure [Pa]
$T_{w,in}$	Water side inlet temperature [K]
h_p	Plates height [m]

Table 3.1: Thermodynamic input parameters for plate heat exchanger when correlations are used

Global heat transfer coefficient U can be obtained by

$$\frac{1}{U} = \frac{1}{h_r} + \frac{1}{h_w} + \frac{s}{k_p} \tag{3.3}$$

where h_r and h_w represent refrigerant and water side heat transfer coefficients respectively, while k_p and s thermal conductivity ans

Geometric input data for plate heat exchanger	
l_{phe}	Plate height [m]
h_c	Plate width [m]
s_c	Channel thickness [m]
A_t	Total area [m^2]
A_c	Channel section [m^2]
s	Plate thickness [m]
Θ	Chevron angle [°]
D_{in}	Inlet diameter [m]
D_{out}	Outlet diameter [m]
U_m	Metal heat transfer coefficient [W/m/K]
ff	Fouling factor

Table 3.2: Geometric input parameters for plate heat exchanger when correlations are used

thickness of the metal, whose plates are made.

Water side of the exchangers presents the same behaviour, with different pressure and temperature levels, being possible to consider water as a single phase fluid, with no phase changes. Correlations used for plate heat exchangers on the water side are then the same for generator, evaporator and condenser and are reported once in the following, before refrigerant formulation.

Following formulation proposed by Wang *et al.* [56], convective heat transfer coefficient is obtained

$$h = \frac{\lambda\, Nu}{D_{eq}} \tag{3.4}$$

where λ represents thermal conductivity of water and for *Nusselt number* one of the following expression is used depending of the flow regime

$$Nu = 0.3190\, Re^{0.6425}\, Pr^{0.4} \quad \text{for } Re \leq 1000 \tag{3.5}$$

$$Nu = 0.3489\, Re^{0.6418}\, Pr^{0.4} \quad \text{for } Re > 1000 \tag{3.6}$$

Water properties and *Prandtl number* are obtained by [37] for temperature and density values related to the considered exchanger sec-

tion and *Reynolds number* by the formula

$$Re = \frac{\rho\, c\, D_{eq}}{\mu} \tag{3.7}$$

where dynamic viscosity μ again is computed by [37].

Pressure losses on water side of the exchanger are computed starting from relation

$$\Delta P = \frac{f G^2 L}{2 D_{eq} \rho} \tag{3.8}$$

where G represents specific mass flow rate, L the length of the considered section of the exchanger, D_{eq} hydraulic diameter which is assumed to be twice the channel spacing and friction factor f is obtained by

$$f = 94.41\, Re^{-0.9125} \quad \text{for } Re < 200 \tag{3.9}$$

$$f = 2.31\, Re^{-0.2122} \quad \text{for } Re \geq 200 \tag{3.10}$$

Different correlations have to be used for refrigerant fluid, specially for stretch of the exchangers where phase change happens. Correlations used are presented in the following, dividing them depending on exchanger function (to heat or to cool the refrigerant) and fluid state in its section (single phase and phase change).

Generator and evaporator model

From the point of view of their model, generator and evaporator present the same behaviour, entering the refrigerant as subcooled liquid, being it heated until saturated conditions are reached, then completely evaporated and finally exiting as superheated vapour.

For refrigerant side of the plate heat exchanger, different correlations have to be used for each part of the channel, distinguishing between preheating, evaporation and superheating of the fluid. A complete scheme of correlations used for different channels and sections of the exchangers are reported in tab.(3.3). It has to be noted

Correlations used for generator and evaporator		
Exchanger section	Heat transfer coeff.	Pressure losses
Preheating	Kakac *et al.* [33]	Kakac *et al.* [33]
Evaporation	Lin *et al.* [57]	Lin *et al.* [57]
Superheating	Kakac *et al.* [33]	Kakac *et al.* [33]

Table 3.3: Summary of references for correlations used for generator and evaporator plate heat exchangers

that some correlations, particularly those related to phase change, are gas specific and they are valid for R134a only. They eventually must be modified if different refrigerant is treated.

Preheating

In preheating part of the exchanger, where subcooled refrigerant is brought to saturated condition, heat transfer coefficient is computed, following Kakac *et al.* [33], by 3.4 where *Nusselt number* is obtained according to

$$Nu = C_h \, Re^n \, Pr^{1/3} \left(\frac{\mu_b}{\mu_w} \right)^{0.17} \tag{3.11}$$

with coefficients summarized in tab.(3.4) and being μ_b and μ_w dynamic viscosities estimated for bulk and wall temperature respectively.

Pressure drop are computed by

$$\Delta P = 4f \frac{L}{D_e} \frac{G^2}{2\rho} \left(\frac{\mu_b}{\mu_w} \right)^{-0.17} \tag{3.12}$$

where friction factor is obtained by

$$f = \frac{K_p}{Re^m} \tag{3.13}$$

using coefficients reported in tab.(3.4).

Evaporation

For evaporating section, correlations proposed by Lin *et al.* [57] for R134a in plate heat exchanger and reported in the following

	Heat transfer			Pressure loss		
Chevron angle (degree)	Reynolds Number	C_h	n	Reynolds Number	K_p	m
≤ 30	≤ 10	0.718	0.349	< 10	50	1
	> 10	0.348	0.663	$10 - 100$	19.40	0.589
				> 100	2.990	0.183
45	< 10	0.718	0.349	< 15	47	1
	$10 - 100$	0.400	0.598	$15 - 300$	18.29	0.652
	> 100	0.300	0.663	> 300	1.441	0.206
50	< 20	0.630	0.333	< 20	34	1
	$20 - 300$	0.291	0.591	$20 - 300$	11.25	0.631
	> 300	0.130	0.732	> 300	0.772	0.161
60	< 20	0.562	0.326	< 40	24	1
	$20 - 400$	0.306	0.529	$40 - 400$	3.24	0.457
	> 400	0.108	0.703	> 400	0.760	0.215
≥ 65	< 20	0.562	0.326	< 50	24	1
	$20 - 500$	0.331	0.503	$50 - 500$	2.80	0.451
	> 500	0.087	0.718	> 500	0.639	0.213

Table 3.4: Coefficients used for *Nusselt* and friction factor in plate heat exchanger proposed by Kakac *et al.* [33]

are used. Using (3.4), convective heat transfer coefficient for evaporating R134a can be obtained by the *Nusselt* which is computed by

$$Nu = 1.926\, Re_{eq}\, Pr_l^{1/3}\, Bo_{eq}^{0.3}\, Re^{-0.5} \quad \text{for } 2000 < Re < 10000 \tag{3.14}$$

where Pr_l represents *Prandtl* of the liquid phase and Re_{eq} and Bo_{eq} the equivalent *Reynolds* and *Boiling* number respectively, which are expressed as

$$Re_{eq} = \frac{G_{eq} D_{eq}}{\mu_l} \tag{3.15}$$

$$Bo_{eq} = \frac{q''_w}{G_{eq} i_{fg}} \tag{3.16}$$

where q''_w is the heat flux per unit area, i_{fg} the enthalpy of vaporization and G_{eq} the equivalent specific mass flow rate expressed by

$$G_{eq} = G\left((1 - X_m) + X_m \left(\frac{\rho_l}{\rho_g}\right)^{1/2}\right) \tag{3.17}$$

with X_m mean vapour quality between inlet and outlet (0.5 in our case) and ρ_l and ρ_g density of liquid and gas phase respectively.

For what concerns pressure losses, following expression is used, still according to Lin *et al.* [57]

$$\Delta P = \frac{2 f G^2 v_m L}{D_{eq}} \tag{3.18}$$

where for L is used the length of the evaporating stretch, v_m is specific volume of vapour-liquid mixture expressed as

$$v_m = \frac{2}{\rho_l + \rho_g} \tag{3.19}$$

and friction factor can be obtained by

$$f = 6.947 \cdot 10^5\, Re_{eq}^{-1.109}\, Re^{-0.5} \quad \text{for } Re_{eq} < 6000 \tag{3.20}$$

$$f = 31.21\, Re_{eq}^{0.04557}\, Re^{-0.5} \quad \text{for } Re_{eq} \geq 6000 \tag{3.21}$$

Superheating

After evaporation is completed, vapour is superheated in the last stretch of the exchanger and heat transfer coefficient again is obtained by computing *Nusselt* with (3.11) and pressure loss by (3.12).

Condenser model

In condenser, superheated vapour enters, it is brought to saturated conditions and then condensed. Finally condensed refrigerant is slightly subcooled before exiting the exchanger. As in the case of generator and evaporator, heat exchanger can then be ideally divided into three sections which are treated as distinct zones, from the point of view of heat transfer coefficients and pressure losses computing. As done for generator, again water correlations are the same along all these three sections. A list of correlations used for modelling condenser behaviour are summarized in tab.(3.5).

Correlations used for condenser		
Exchanger section	Heat transfer coeff.	Pressure losses
Desuperheating	Kakac *et al.* [33]	Kakac *et al.* [33]
Condensation	Lin *et al.* [58]	Lin *et al.* [58]
Subcooling	Kakac *et al.* [33]	Kakac *et al.* [33]

Table 3.5: Summary of references for correlations used for condenser plate heat exchangers

Desuperheating

Superheated vapour comes from supersonic diffuser of the ejector so it is necessary to desuperheat it before condensation. Heat transfer coefficient and pressure drop along this stretch of the exchanger are computed following formulation proposed by Kakac *et al.* [33], already described in previous sections of this paragraph.

Condensation

For condensing section, again correlations specific for R134a are used, found in Lin *et al.* [58]. Using (3.4), convective heat transfer coefficient for condensing R134a can be obtained by the *Nusselt*

which is computed by

$$Nu = 4.118\, Re_{eq}^{0.4}\, Pr_l^{1/3} \tag{3.22}$$

where Pr_l represents *Prandtl* of the liquid phase and Re_{eq} is defined according to (3.15).

For what concerns pressure losses, expression (3.18) is used for which friction factor is obtained by

$$f = 94.75\, Re_{eq}^{-0.0467}\, Bo^{-0.5} \left(\frac{P_m}{P_c}\right)^{0.8} \tag{3.23}$$

where P_m represents nominal condensation pressure and P_c critical pressure for R134a (corresponding to 4.064 Mpa).

Subcooling

After condensation is completed, liquid is subcooled in the last stretch of the exchanger and heat transfer coefficient and pressure drop are computed by correlations used in desuperheating stretch previously described.

Comparing heat transfer coefficients obtained using correlations with those provided by manufacturer, it appears that they underestimate the values suggested by the manufacturer. Reason of this behaviour is probably due to mismatch between geometrical parameters used in correlations and real ones. Except for external dimensions and inlet and outlet diameter, geometrical information regarding channels characteristics, spacing and inclination are patented and only possible values can be assumed. On the other hand, correlations with general validity for analysis of heat transfer coefficient and pressure drop in plate heat exchangers are not present in literature [24], being proposed models usually derived for specific geometrical configuration and working conditions. Finally, formulations found in literature are specific for considered fluid and rarely they can be extended to different refrigerant type. If it is necessary to test various fluids during analysis of the ejector refrigeration

plant, literature correlations have to be found and difficulties arise to gather them, particularly for recently developed refrigerants.

Based on these considerations, a further method to treat plate heat exchangers is implemented in the program which overcomes almost all limitations of correlations use, though it however implies some drawbacks. This approach is described in the following section.

Plate heat exchangers tables

During plant project different feature can affect fluid choice as previously described (§2.5). It is then often necessary to change refrigerant fluid type and verify working conditions of the plant for the new one. Using correlations to characterize plate heat exchangers shows some limitations connected to literature availability of formulation to describe fluid behaviour, specially during phase changes. In this case, different strategy can be implemented using data provided by heat exchangers manufacturers. In fact, it is usually available software which gets all characteristics of the exchanger starting from some known input parameter (generally user defined). It is possible to summarize these data in formatted tables which in turn can be read from the optimization program. Differently from correlations use, input parameters assigned to define exchanger represent now discrete values instead continuous ones, but on the other hand they better match specific exchanger behaviour. Moreover, manufacturers usually update quite rapidly their software to take into account for new refrigerant fluids, generally supplying them in advance of literature correlations. Example of optimization program input data for heat exchangers is reported in tab.(3.6). Different values for nominal temperatures, powers, water mass flow rates and inlet temperatures and superheating for generator and evaporator are chosen, over which optimization is performed.

Plate heat exchangers input table			
	Generator	Evaporator	Condenser
T_{nom} [°C]	85, 95, 105	2, 4, 6	30, 35, 40
Inlet quality	-	0.2, 0.22, 0.25	-
Power [kW]	150	40, 50, 60	190, 200, 210
$\dot{m}_wat$ [kg/s]	2.6, 3.6, 5	-	5, 10, 15
$T_{in,wat}$ [°C]	110, 115, 120	-	25, 30
Overheat [°C]	6, 8, 10	1, 3, 5	-
Number	81	81	34

Table 3.6: Input data table for plate heat exchangers

3.3 Optimization program description

Optimization is accomplished by determining conditions for which objective function presents its maximum value when working temperatures, fluid mass flow rates and heat exchangers characteristics are varied within intervals depending on system characteristics, chosen fluid and selected components.

Objective function chosen for the analysis is the COP of the plant which is expressed by

$$f_{obj} = \frac{Q_{ref}}{Q_{gen} + W_{gen} + W_{eva} + W_{con} + W_{pmp}} \tag{3.24}$$

where Q_{ref} is refrigeration power of the plant, Q_{gen} thermal power supplied at generator, W_{gen}, W_{eva} and W_{con} heat exchangers friction losses at generator, evaporator and condenser respectively and W_{pmp} the power of the feed pump.

Pressure losses on water side are computed by

$$W_i = 2\dot{m}_w \frac{|P_{w,in} - P_{w,out}|}{(\rho_{w,in} + \rho_{w,out})} \quad ; \quad i = gen, eva, con \tag{3.25}$$

while pump power is expressed as

$$W_{pmp} = 2\dot{m}_{ref} \frac{P_{gen} - P_{con}}{\rho_{gen} + \rho_{con}} \tag{3.26}$$

It has to be noted that optimum for the COP of the plant it is

not necessarily obtained by optimization of single components, being system optimization more relevant than that of single components to obtain better performance [20]. Multivariate optimization method is then used, as said, to consider the several possibilities of parameters combinations for all key components of the plant as a whole.

Optimization program is conceived to present a modular structure by using specific functions and subroutines for each feature so ensuring flexibility, but increasing fragmentation and nesting complexity. However it can be conceptually divided into three main functional blocks as shown in fig.(3.2). First block of subroutines constitutes proper optimization code, second one defines components characteristics and sizing and the last one deals with the output generation. In the remaining section main functions and subroutines defined in the program are described, following proposed conceptual partitioning and focusing on the most significant for program operation, while complete program code is reported in Appendix A.

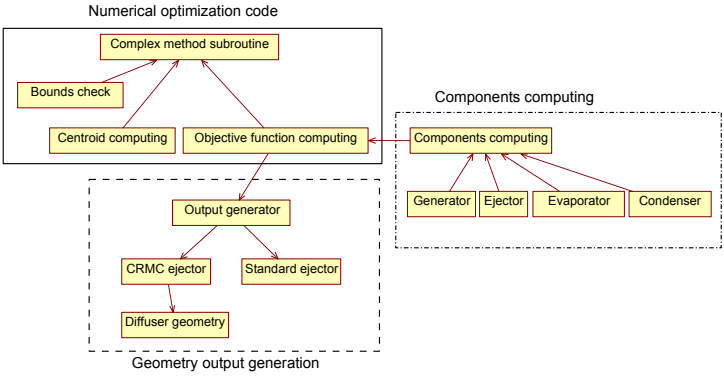

Figure 3.2: Block diagram of the main procedures of the program

Starting from a main subroutine (*PlantDesign*), plant working conditions, summarized in tab.(3.7), are read (*readdat*) from file. Heat exchangers input data are different based on method they are treated (§3.2.2): when their heat transfer coefficients and pressure

drops are determined by correlations, program needs water temperature and pressure conditions and geometrical characteristics of exchanger and plates, otherwise a table containing formatted data (see for example tab.(3.6)), obtained by manufacturer software, is read from the program.

Plant thermodynamic input parameters	
Q_{ref}	Refrigeration power [W]
$T_{g,h,in}$	Generator inlet temperature (water side) [K]
$P_{g,h}$	Generator nominal pressure (water side) [Pa]
$T_{e,h,in}$	Evaporator inlet temperature (water side) [K]
$P_{e,h}$	Evaporator nominal pressure (water side) [Pa]
$T_{c,w,in}$	Condenser inlet temperature (water side) [K]
$P_{c,w,out}$	Condenser nominal pressure (water side) [Pa]

Table 3.7: Optimization program input data related to the plant thermodynamic parameters (input control flags not shown)

Moreover, parameters over which optimization is performed are also read (*updt_bounds*) from file. A list of the parameters varied during optimization and their bounds are reported, as example, in tab.(3.8).

	Lower bound	Upper bound
Generator water mass flow rate [kg/s]	2	10
Generator nominal pressure [MPa]	2.5	3.5
Superheat at generator exit [K]	2	10
Primary mass flow rate [kg/s]	0.8	1
Evaporator water mass flow rate [kg/s]	1	5
Evaporator nominal temperature [K]	277.15	279.15
Condenser water mass flow rate [kg/s]	5	30
Condenser nominal temperature [K]	308.15	318.15
Condenser number of plates	100	200
Evaporator number of plates	20	60
Generator number of plates	80	100

Table 3.8: Range of variable parameters over which optimization is accomplished for R134a

After input parameters are read, optimization starts, following procedure described in §3.1.1 and implemented in subroutines *sub_complex*, *centroid* and *checkbounds* reported in Appendix A. Each time input variables are changed during optimum searching

procedure, objective function has to be updated according to exchangers characteristics assigned by tables or iteratively computed starting from input data and using correlations. Ejector design is done, with subroutine *Eje_design_Eames*, based on *Constant Rate of Momentum Change* proposed by Eames [12], which is deeply described in §4.8.1. Differently from new *CRMC* design procedure proposed in following chapter, no friction losses are explicitly computed in optimization program and they are considered applying an overall isentropic efficiency. Moreover, according to *calorically* perfect gas model ([51],[31]), constant specific heat ratio is considered along mixing chamber and diffuser.

Many iterations are necessary to the optimization program to converge for each single run, mainly because starting from randomly selected input values, which guarantees to widely explore input ranges, causes that even incoherent configurations are considered and then discarded if convergence is not reached for any components. All these trials are saved in specific output file for following check.

3.4 Optimization results

Optimization program outputs are plate heat exchangers selected or their characteristics for the optimum configuration and main geometrical information about ejector. In particular, throat sections of the nozzles and the diffuser is reported which are obtained assuming isentropic efficiencies. Moreover, a profile of the section of the supersonic diffuser is computed at optimum conditions applying *Constant Rate of Momentum Change* method.

Results of optimization program are summarized, as example, in tab.(3.9) and profile of the diffuser is shown in fig.(3.3).

Optimization results - ejector geometric characteristics			
	Area [mm^2]	Radius [mm]	Velocity [m/s]
Primary nozzle			
Throat	50.8	4.02	-
Outlet	140.4	6.69	319
Secondary nozzle			
Outlet	188	-	176
Mixing chamber			
Inlet	328.8	10.23	-
Outlet	315.4	10.02	264
Supersonic diffuser			
Inlet	315.4	10.02	264
Throat	243.3	8.80	188
Outlet	9110.1	53.85	2.8

Table 3.9: Main ejector characteristics obtained as result of the optimization program for fluid R134a

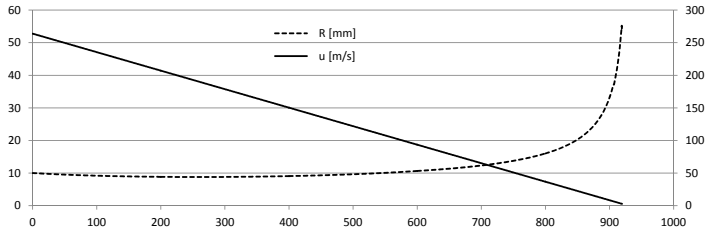

Figure 3.3: Radius and velocity along diffuser designed applying *CRMC* method in the optimization program

Chapter 4

Plant design

Starting from optimization program results, a design program is developed whose purpose is to define ejector geometry based on a one-dimensional fluid flow model [22]. Design program permits to evaluate plant COP varying heat exchangers characteristics and temperature levels and to find geometric design of ejector. Program description follows code structure, analysing, component by component, used model and assumptions made. Finally program results and sensitivity analysis are reported.

4.1 Model description

Fixed working conditions of plate heat exchangers used as input, program defines sections of mixing chamber and supersonic diffuser along flow direction. Inlet conditions for mixing chamber are determined by characteristics of primary and secondary nozzle, whose geometry is defined outside the design program and whose efficiency is found by applying ideal gas model implemented in specific simulation code. Starting from some hypothesises about flow behaviour, program finds area of the duct with a step by step procedure moving forward along x coordinate.

Real gas model is adopted, over the whole ejector, to compute

thermodynamic properties of fluid by using NIST REFPROP libraries [38].

Generator, condenser and evaporator are represented by three counterflow corrugated plate heat exchangers, through which water gives or receive thermal power by refrigerant depending of the function of the exchanger. Design program considers heat exchangers by their temperature levels and pressure drops and by the mass flow rates of water and refrigerant. Their characteristics are found outside the design code using a selection program provided by the manufacturer. Some hypothesis are made on pressure drops distribution internal to exchangers itself, which are described and analysed in the following.

The cycle points, shown in fig.(4.1) and fig.(4.2) on a Ts and on a Ph diagram respectively and in fig.(4.3) on the plant scheme, are defined in tab.(4.1).

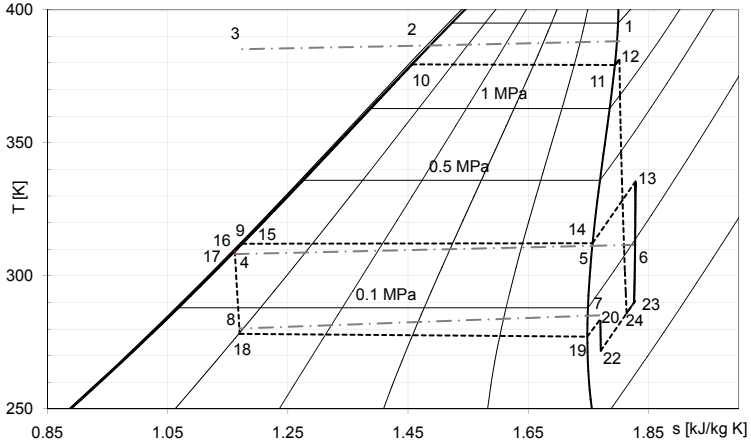

Figure 4.1: Ts diagram for the ejector cycle with R245fa

The following hypotheses are made:

- Flow in ejector is adiabatic and steady and it is modelled as one-dimensional.
- Sonic conditions are established at secondary nozzle exit.

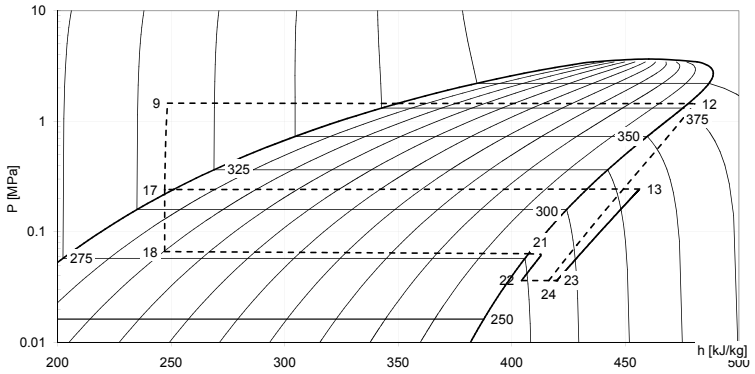

Figure 4.2: Ph diagram for the ejector cycle with R245fa

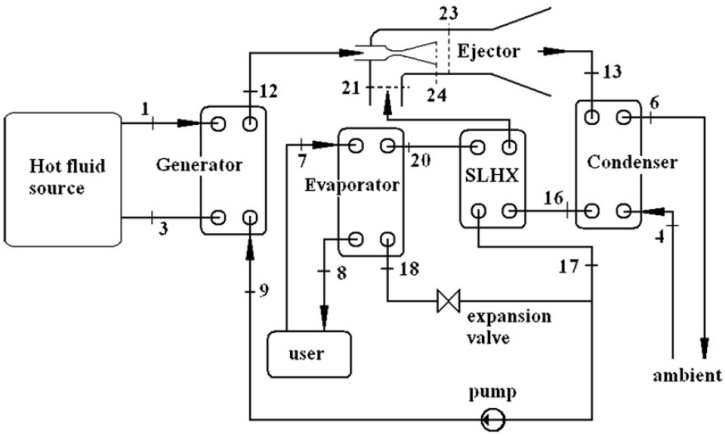

Figure 4.3: Scheme of ejector refrigeration system

1	Water at generator inlet	13	Fluid at condenser inlet
2	Water at generator pinch point	14	Start of condensation
3	Water at generator outlet	15	End of condensation
4	Water at condenser inlet	16	Fluid at condenser exit
5	Water at condenser pinch point	17	Liquid at SLHX exit
6	Water at condenser outlet	18	Evaporator inlet
7	Water at evaporator inlet	19	Evaporation
8	Water at evaporator outlet	20	Fluid at evaporator exit
9	Fluid at generator inlet	21	Vapour at secondary inlet
10	Fluid evaporation	22	Choked secondary flow
11	End of evaporation	23	Mixed fluid/diffuser inlet
12	Fluid at generator exit	24	End of primary expansion
		25	Primary nozzle throat

Table 4.1: List of cycle points

- Constant pressure mixing process is assumed and flows are considered completely mixed at mixing chamber exit.
- Heat losses in transfer lines are neglected.

Description of the program flow is reported in the next sections, giving details of procedures used to model and design each component of the plant.

4.2 Design program description

Tab.(4.2) shows the input data, comprising minimum temperature differences and pressure losses within heat exchangers, water flow rates at generator and condenser and flow sections at generator exit (12), secondary inlet (21) and condenser inlet (13). A complete flow chart of the program is shown in fig.(4.4).

The calculation starts assigning a hypothetical entrainment ratio and then iteratively modifying it until pressure value at condenser inlet is matched.

Fixed parameters

Thermodynamic properties at fixed points are calculated before the iterative loop starts. This is the case for condenser exit temperature $T(16)$, related to water temperature at condenser inlet $T(4)$

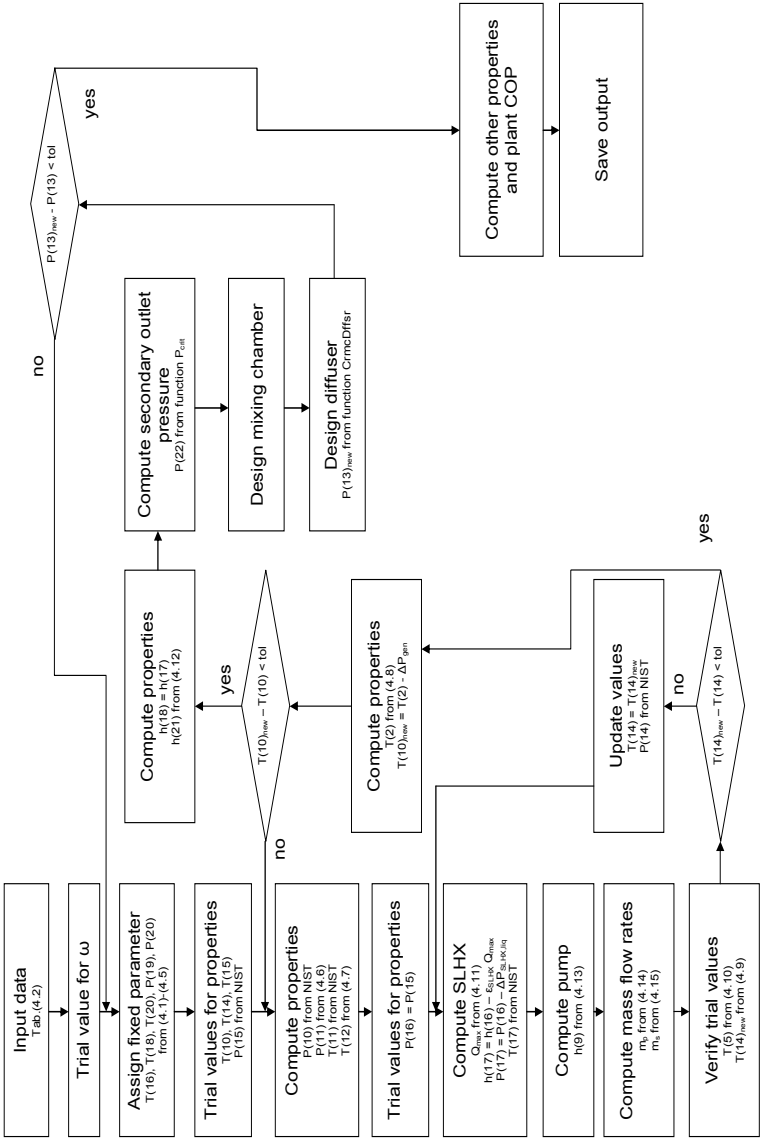

Figure 4.4: Complete flow chart of the design program

ΔT_{gen}	Generator minimum ΔT [°C]	$\dot{m}(4)$	Water flow rate at condenser [kg/s]
ΔT_{con}	Condenser minimum ΔT [°C]	ΔP_{suph}	Superheating at evaporator exit [°C]
ΔT_{eva}	Evaporator minimum ΔT [°C]	ΔP_{subc}	Subcooling at condenser exit [°C]
ΔT_{sh}	Superheating at generator [°C]	ΔP_{con}	Pressure loss at condenser [MPa]
c_0	Minimum fluid velocity [m/s]	ΔP_{eva}	Pressure loss at evaporator [MPa]
η_p	Primary expansion efficiency	ΔP_{gen}	Pressure loss at generator [MPa]
η_s	Secondary expansion efficiency	$\Delta P_{rig,l}$	Pressure loss at SLHX (liquid) [MPa]
η_{mix}	Mixing section efficiency	$\Delta P_{rig,v}$	Pressure loss at SLHX (vapour) [MPa]
η_p	Feed pump efficiency	ϵ_{rig}	SLHX efficiency
W_{gen}	Generator thermal power [kW]	D_{gen}	Diameter of generator pipe [mm]
$T(1)$	Water at generator inlet [°C]	$D_{s,int}$	Diameter of secondary annulus (int) [mm]
$T(4)$	Water at condenser inlet [°C]	$D_{s,ext}$	Diameter of secondary annulus (ext) [mm]
$T(7)$	Cold water return [°C]	D_{con}	Diameter at condenser inlet [mm]
$T(8)$	Cold water at deliver [°C]	l_{CRMC}	Length of the CRMC profile [mm]
$\dot{m}(1)$	Generator water flow rate [kg/s]		

Table 4.2: Design program input data

by

$$T(16) = T(4) + \Delta T_{con} \tag{4.1}$$

and for evaporation inlet temperature $T(18)$, related to water temperature at evaporator outlet $T(8)$ by

$$T(18) = T(8) - \Delta T_{eva} \tag{4.2}$$

A superheating ΔT_{sh} is set at evaporator exit, but the result is corrected if resulting refrigerant temperature $T(20)$ is higher than cold water temperature $T(7)$ minus ΔT_{eva}, choosing the minimum between the two temperatures:

$$T(20) = \min\left[(19) + \Delta T_{suph}\,;\, T(7) - \Delta T_{eva}\right] \tag{4.3}$$

Pressures and enthalpies are calculated from temperatures accounting for evaporator pressure loss ΔP_{eva}, which is assigned for 90% to evaporating section and 10% to superheating section:

$$P(19) = P(18) - 0.9\ \Delta P_{eva} \tag{4.4}$$

$$P(20) = P(19) - 0.1\ \Delta P_{eva} \tag{4.5}$$

Generator and condenser temperatures

In addition, some trial values have to be initially assigned for vaporization and condensing temperatures at the generator and condenser respectively. Fluid side temperature are in fact determined by water temperatures and pinch point temperature differences in heat exchanger, hence trial values are iteratively updated until *pinch point* temperature differences are matched.

Starting vaporization temperature within the generator, given hot water temperature $T(1)$, is set at $T(10) = T(1) - 2\ \Delta T_{gen}$. From $T(10)$, pressure and enthalpy are obtained as saturated liquid values. Pressure $P(11)$ is calculated attributing 1/3 of generator

pressure drop to the evaporation section (§4.10):

$$P(11) = P(10) + \frac{1}{3}\,\Delta P_{gen} \tag{4.6}$$

while temperature $T(11)$ is saturation temperature at $P(11)$.

Temperature $T(12)$ is higher than $T(11)$ by ΔT_{suph}, which represents superheating at generator. However, $T(12)$ must be lower than maximum hot water temperature $T(1)$ by minimum temperature difference ΔT_{gen} so it is assigned by

$$T(12) = min[T(11) - \Delta T_{suph}\ ;\ T(1) - \Delta T_{gen}] \tag{4.7}$$

Another third of the generator pressure drop is assigned to superheating between $P(11)$ and $P(12)$ and enthalpy can be computed knowing pressure and temperature. Considering suction line heat exchanger (§4.3) and pump behaviour (§4.4) primary mass flow rate is achieved. Given primary mass flow rate m_p, water temperature $T(2)$ at the beginning of evaporation can be evaluated by energy balance across the evaporation and superheating part of the generator

$$T(2) = T(1) + \frac{m_p\ (h(12) - h(10))}{c_{p,w}\ m(1)} \tag{4.8}$$

A new value $T(10) = T(2) - \Delta T_{gen}$ is hence found. If the difference between new and previous value is less than a given tolerance, the iteration stops. Otherwise, the new value is assumed for $T(10)$ and iteration is repeated until convergence.

During each iteration, a nested loop is executed to define condensing temperature $T(14)$ which can be assigned by

$$T(14) = T(5) + \Delta T_{con} \tag{4.9}$$

from which pressure and enthalpy are obtained as saturated vapour values.

From $P(14)$, pressure $P(15)$ at condensation end is achieved, assuming 3/4 pressure drop during condensation phase, by $P(15) =$

$P(14) - 3/4\,\Delta P_{con}$. Neglecting subcooling pressure losses ($P(16) = P(15)$) and being temperature value at condenser exit defined by (4.1), enthalpy $h(16)$ can be obtained by temperature and pressure.

Finally water temperature $T(5)$ at start of condensation is found from the energy balance between start of condensation (14) and condenser exit (16) on refrigerant side:

$$T(5) = T(4) + \frac{(m_s + m_p)\,(h(14) - h(16))}{c_{p,w}\,m(4)} \tag{4.10}$$

As said, temperature $T(14)$ must be higher than water temperature $T(5)$ by an amount ΔT_{con}, so program iteratively modify $T(14)$ until condition is satisfied.

Once convergence is reached, the constant enthalpy expansion (17–18) can be modelled.

4.3 Suction line heat exchanger

An internal loop is needed for the regenerator SLHX, which has effectiveness ϵ_{SLHX} (according to Kays and London [34] definition) as input. Pressure drop in the subcooling condenser section is neglected ($P(16) = P(15)$). The minimum heat capacity C_{min} between the two sides of the SLHX should be C_{vap} on the vapour side. The maximum heat transfer is given by minimum heat capacity times maximum temperature difference, i.e. between the liquid exiting from the condenser and the vapour exiting from the evaporator:

$$Q_{max} = C_{min}(T(16) - T(20)) \tag{4.11}$$

Liquid enthalpy at SLHX exit is found from ϵ_{SLHX}. A pressure drop $\Delta P_{SLHX,liq}$ is set on the liquid side of SLHX. Other properties can be calculated from $h(17)$ and $P(17)$.

Vapour heating (20–21) across the SLHX is calculated from the

previously found maximum heat transfer and from effectiveness

$$h(21) = h(20) + \frac{\epsilon_{SLHX}\, Q_{SLHX,max}}{\omega} \tag{4.12}$$

Pressure at SLHX exit is found taking into account for pressure drop $\Delta P_{SHLX,vap}$ on the vapour side and then all properties at secondary inlet (21) are evaluated. If the SLHX is absent, point 16 is equivalent to 17 and also points 20 and 21 coincide.

4.4 Pump

Pressure at generator inlet $P(9)$ is determined assuming another 1/3 of pressure drop ΔP_{gen} on the liquid heating part of the generator. If the pump had no losses, the enthalpy at point 9 would be h_{rev}, i.e. the enthalpy at $P(9)$ and $s(17)$. Applying pump efficiency η_p enthalpy $h(9)$ can be obtained by

$$h(9) = h(7) + \frac{h_{rev} - h(17)}{\eta_p} \tag{4.13}$$

$T(9)$ and $s(9)$ are found from pressure and enthalpy. The primary mass flow rate is calculated from generator power W_{gen} and enthalpy difference

$$m_p = \frac{W_{gen}}{h(12) - h(9)} \tag{4.14}$$

The value of the entrainment ratio ω gives the secondary mass flow rate m_s

$$m_s = \omega\, m_p \tag{4.15}$$

4.5 Secondary expansion

Starting from properties at SLHX exit, the assigned flow area $\Omega(21)$ yields the velocity.

$$c(21) = \frac{m_s}{\rho(21)\, \Omega(21)} \tag{4.16}$$

From this point (21) the secondary fluid expands to sonic conditions. A specific function finds the maximum of the product ρc along an isentropic expansion from the given inlet pressure, enthalpy and velocity. This function is needed because ideal gas assumption has been dropped and yields the critical pressure $P(22)$. An ideal enthalpy value h_{rev} corresponds to pressure $P(22)$ and entropy $s(21)$. To allowing for non isentropic expansion, enthalpy at sonic point (22) is obtained by introducing the secondary expansion efficiency η_s:

$$h(22) = h(21) - \eta_s \left(h(21) - h_{rev}\right) \tag{4.17}$$

Temperature, entropy and density are found from pressure and enthalpy. Being heat exchange and potential energy neglected, the energy balance along the secondary nozzle gives the fluid velocity:

$$c(22) = c(21) - 2 \left(h(21) - h(22)\right)^{0.5} \tag{4.18}$$

Throat section can be finally achieved by flow rate, density and velocity and it is assumed to coincide with the exit section for the secondary expansion.

4.5.1 Secondary expansion efficiency

Secondary expansion efficiency η_s is obtained by modelling secondary duct according to generalized steady one-dimensional flow model ([31],[60]) with a standalone program external to design code. Fluid is considered to flow in a variable-area duct with friction, without mass addition and without significant heat exchange. Area change and friction are the *driving potentials* of the flow dynamics. Even under this hypothesis no closed-form solution for the motion equations is available for this flow therefore a numerical procedure is needed. Solution of the following differential equation in terms of Mach number is needed and the other properties along the flow can

be found by knowing Mach number.

$$\frac{dM}{dx} = \frac{M}{2}\frac{\Psi\, G(x,\gamma,M)}{1-M^2} \tag{4.19}$$

where

$$\Psi = 1 + \frac{\gamma-1}{2}M^2 \tag{4.20}$$

$$G(x,\gamma,M) = -\frac{2}{A}\frac{dA}{dx} + M^2\frac{4f_F}{D_h} \tag{4.21}$$

with M the Mach number, A the duct section and $D_h(=4A/P)$ its hydraulic diameter. Fanning friction factor f_F is given by

$$f_F = \frac{\tau}{1/2\ \rho\ c^2} \tag{4.22}$$

where τ is the viscous shear stress, $\gamma(=c_p/c_v)$ the specific heat ratio, ρ the fluid density and c its velocity.

A fourth order Runge-Kutta method [46] is used to integrate (4.19). To this aim the analytic expression of duct section A as a function of abscissa x must be achieved. Secondary flow moves along an annular duct between conical external surface of the primary nozzle and internal surface of the convergent secondary inlet duct which is designed connecting the Tee shaped pipe upstream the flow with the mixing chamber downstream. Inlet and outlet section is then fixed and connected with one fillet radius defined according to fig.4.5.

Fixing internal $r_{int,in}$ and external $r_{ext,in}$ inlet radii, external outlet radius $r_{ext,out}$, downstream fillet radius $r_{fill,2}$ and angle α_1 and angle α_3 of the conical external surface of the primary, fillet radius $r_{fill,1}$ and nozzle length of the secondary convergent duct along the x-axis can be obtained by

$$r_{fill,1} = \frac{r_{ext,in} - (r_{ext,out} + r_{fill,2}\,(1-\cos\alpha_1))}{1-\cos\alpha_1} \tag{4.23}$$

$$l_x = (r_{fill,1} + r_{fill,2})\sin\alpha_1 - r_{fill,2}\sin\alpha_3 \tag{4.24}$$

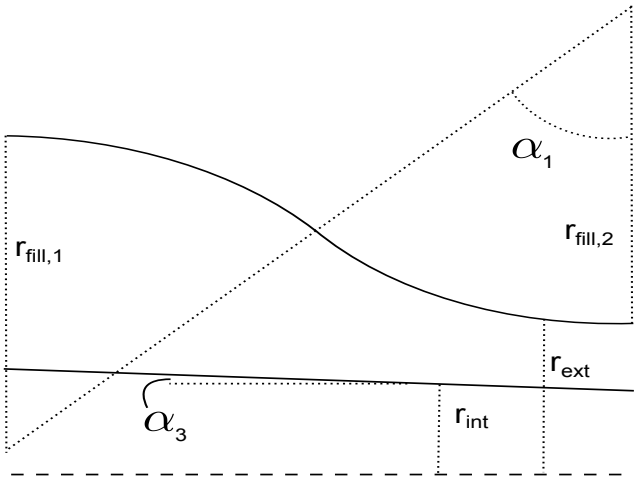

Figure 4.5: Secondary nozzle profile

Duct section is given by

$$A(x) = \pi \left(r_{ext}^2 - r_{int}^2\right) \tag{4.25}$$

so the *driving potential* due to area change of the (4.21) can be expressed by

$$\frac{1}{A}\frac{dA}{dx} = \frac{2}{(r_{ext}^2 - r_{int}^2)}\left(r_{ext}\frac{dr_{ext}}{dx} - r_{int}\frac{dr_{int}}{dx}\right) \tag{4.26}$$

where expression for r_{ext} and its derivative can be obtained dividing duct in two stretches related to concavity of external shape.

For $\mathbf{x < l_{fill,1}}$

$$r_{ext}(x) = r_{ext,in} - r_{fill,1}\left(1 - \cos\left(\frac{x}{r_{fill,1}}\right)\right) \tag{4.27}$$

$$\frac{dr_{ext}}{dx}(x) = -\frac{x}{\sqrt{r_{fill,1}^2 - x^2}} \tag{4.28}$$

For $\mathbf{x > l_{fill,1}}$

$$r_{ext}(x) = r_{ext,out} + r_{fill,2}\left(1 - \cos\left(\frac{l_x - x}{r_{fill,2}}\right)\right) \tag{4.29}$$

$$\frac{dr_{ext}}{dx}(x) = -\frac{l_x - x}{\sqrt{r_{fill,2}^2 - (l_x - x)^2}} \tag{4.30}$$

A single expression is used for the internal surface radius r_{int} and its derivative along the whole secondary duct length

$$r_{int}(x) = r_{int,out} + \tan\alpha_3\,(l_x - x) \tag{4.31}$$

$$\frac{dr_{int}}{dx} = -\tan\alpha_3 \tag{4.32}$$

Solving (4.19) for M expression for thermodynamic properties along the duct can be obtained starting from inlet conditions according to ideal gas hypothesis. Being expansion adiabatic, static temperature can be expressed by

$$\frac{T}{T_{in}} = \frac{1 + \frac{\gamma-1}{2}M_{in}^2}{1 + \frac{\gamma-1}{2}M^2} \tag{4.33}$$

and starting from mass flow rate definition following expression can be derived for static pressure

$$\frac{P}{P_{in}} = \frac{\dot{m}A_{in}M_{in}}{\dot{m_{in}}AM}\sqrt{\frac{T}{T_{in}}} \tag{4.34}$$

Under hypothesis of adiabatic expansion stagnation temperature stays unchanged and stagnation pressure is obtained by

$$\frac{P_0}{P_{0,in}} = \frac{P}{P_{in}}\left(\frac{1 + \frac{\gamma-1}{2}M^2}{1 + \frac{\gamma-1}{2}M_{in}^2}\right)^{\frac{\gamma}{\gamma-1}} \tag{4.35}$$

Input parameters	
$f_F = 0.003$	Fanning friction factor
$\gamma = 1.08$	Specific heat ratio
$T_0 = 283K$	Inlet stagnation temperature
$P_0 = 0.061MPa$	Inlet stagnation pressure
$P_b = 0.036MPa$	Exit pressure
Results	
$\dot{m} = 0.24kg/s$	Mass flow rate
$\eta_s = 0.99$	Isentropic efficiency

Table 4.3: Secondary nozzle efficiency determination results

Finally isentropic efficiency η_s is achieved [10]

$$\eta_s = \frac{T - T_0}{T_s - T_0} = \frac{1 - \frac{T}{T_0}}{1 - \left(\frac{P}{P_0}\right)^{\frac{\gamma-1}{\gamma}}} \tag{4.36}$$

Results for computed efficiency is summarized in tab.(4.3), while pressure profile along duct are shown in fig.(4.6).

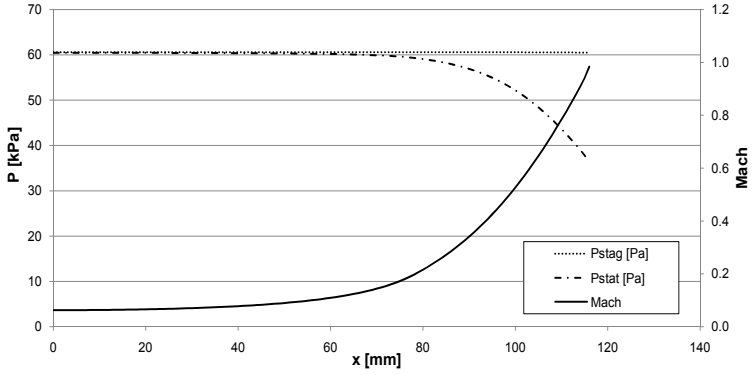

Figure 4.6: Static and stagnation pressure profile along secondary nozzle

Fanning friction factor value is obtained as one fourth of the Darcy friction factor extracted by Moody diagram for the relative roughness value supplied by the duct manufacturer and *Reynolds*

number of the flow. Model assumes constant value for γ while it actually changes slightly along the duct according to gas properties changes. Influence of this behaviour on the value of efficiency is however negligible.

4.6 Primary expansion

Once the pressure at the end of secondary expansion is known, given the hypothesis of constant pressure mixing, the same pressure is assumed for the primary flow at nozzle exit. Hence, from generator exit (12), the fluid expands to pressure $P(24) = P(22)$. Entropy and density can be found from pressure and temperature calculated above. Initial velocity $c(12)$ is found as done for $c(21)$ in (4.16) applying mass flow rate definition.

Ideal enthalpy h_{rev} is found at $P(24)$ and $s(12)$. The real value $h(24)$ comes from the primary nozzle efficiency η_p:

$$h(24) = h(12) - \eta_p \left(h(12) - h_{rev}\right) \tag{4.37}$$

Exit velocity $c(24)$ is found as done for $c(22)$ in (4.18).

4.6.1 Primary expansion efficiency

Like for secondary expansion, primary expansion efficiency is obtained modelling primary nozzle by 1D generalized flow equations and solving them by fourth-order Runge-Kutta method. Sonic point is a singular point of the (4.19) because denominator becomes unbounded while M approaches 1. Following procedure proposed by Beans [4] it is possible to transit sonic point location and overcome discontinuity. Rearranging (4.19) as

$$\frac{1 - M^2}{M^2} \frac{dM^2}{dx} = \Psi \, G(x, \gamma, M) \tag{4.38}$$

and evaluating it at sonic conditions yields

$$\left(\Psi\ G(x,\gamma,M)\right)_{M=1} = 0 \tag{4.39}$$

Since Ψ is not zero at the sonic point, follows that

$$G(x,\gamma,M)|_{M=1} = 0 \tag{4.40}$$

and then the limit

$$\lim_{M\to 1} \frac{G(x,\gamma,M)}{1-M^2} = \frac{0}{0} \tag{4.41}$$

results indeterminate vanishing both numerator and denominator. Applying l'*Hospital's rule* the value of dM/dx can be evaluated at the sonic point by

$$\lim_{M\to 1} \frac{dM)}{dx} = \lim_{M\to 1} \frac{M\Psi G(x,\gamma,M)}{2\left(1-M^2\right)} = \frac{\gamma+1}{4}\left(\frac{dG/dx}{-2MdM/dx}\right)_{M=1} \tag{4.42}$$

Evaluating dG/dx and substituting it in previous equation, following quadratic expression in terms of dM/dx at the sonic point is obtained

$$\left(\frac{dM)}{dx}\right)^2_{M=1} = -\frac{\gamma-1}{8}\left(-2\frac{d}{dx}\left(\frac{1}{A}\frac{dA}{dx}\right)_{M=1} + \gamma\frac{d}{dx}\left(\frac{4f_F}{D_h}\right)_{M=1}\right) \tag{4.43}$$

Negative and positive value of the solution of (4.43) corresponds to subsonic and supersonic flow downstream the sonic point respectively. Using limiting value of dM/dx in the neighbourhood of sonic point permits to overcome singularity and proceed the integration along the duct in the supersonic stretch.

About its geometry, primary nozzle profile is designed fixing inlet r_{in} and outlet r_{ext} section radii, throat radius r_{th} and a fillet radius r_{fill} and exit angle α respectively for downstream stretch and upstream conical stretch (fig.4.7).

To avoid discontinuity in profile derivative circular profile is connected to linear one imposing tangency between circumference and linear segment, so also assuring continuity in the neighbourhood of

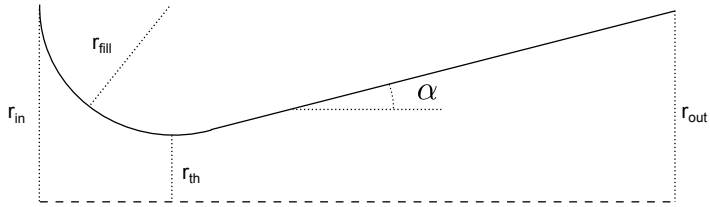

Figure 4.7: Primary nozzle profile

the throat. Nozzle throat position can be obtained by above parameters

$$x_{th} = \sqrt{r_{fill}^2 - (r_{th} + r_{fill} - r_{in})^2} \tag{4.44}$$

while coordinates of the connection of circular and linear profile is done by

$$l_{fill} = x_{th} + r_{fill} \sin \alpha \tag{4.45}$$

$$y_{fill} = r_{th} + r_{fill}(1 - \cos \alpha) \tag{4.46}$$

Area section is simply the area of the circular section associated with duct radius so area change *driving potential* can be defined by

$$\frac{1}{A}\frac{dA}{dx} = \frac{2}{r}\frac{dr}{dx} \tag{4.47}$$

Again, expression of r and its derivative is derived for two different stretches for circular and linear profile.

For $\mathbf{x < l_{fill}}$

$$r(x) = r_{th} + r_{fill}\left(1 - \cos\left(\frac{x - x_{th}}{r_{fill}}\right)\right) \tag{4.48}$$

$$\frac{dr}{dx} = \frac{x - x_{th}}{\sqrt{r_{fill}^2 - (x - x_{th})^2}} \tag{4.49}$$

For $\mathbf{x > l_{fill}}$

$$r(x) = y_{fill} + \tan \alpha \, (x - l_{fill}) \tag{4.50}$$

Input parameters	
$f_F = 0.003$	Fanning friction factor
$\gamma = 1.13$	Specific heat ratio
$T_0 = 381$	Inlet stagnation temperature [K]
$P_0 = 1.44$	Inlet stagnation pressure [MPa]
$P_b = 0.037$	Exit pressure [MPa]
Results	
$x_{sonic} = 28.87$	Sonic point abscissa [mm]
$\dot{m} = 0.36$	Mass flow rate [kg/s]
$\eta_s = 0.98$	Isentropic efficiency

Table 4.4: Primary nozzle efficiency determination results

$$\frac{dr}{dx} = \tan\alpha \tag{4.51}$$

Gas properties and isentropic efficiency for primary nozzle is then achieved, using formulation described for secondary nozzle, with (4.33), (4.34) and (4.36) respectively. Again the single specific heat ratio is assigned so that its value is representative of the values computed by real gas model over temperature and pressure range between inlet and outlet conditions.

Above described formulation is implemented in Excel VBA code (reported in Annex C) and input parameters and results of the program are shown in tab.4.4.

Fanning friction factor value is obtained as described in §4.6.1. Mean value of γ is assigned for efficiency determination. Changes in specific heat ratio, differently to secondary expansion, are quite large for primary nozzle, however running program with maximum and minimum value of γ shows that values of efficiency agrees within 1% which is considered acceptable for our purpose.

4.7 Mixing

After exiting primary and secondary nozzle, two flows are supposed to mixing at constant pressure ($P(22) = P(24) = P(23)$) in a region

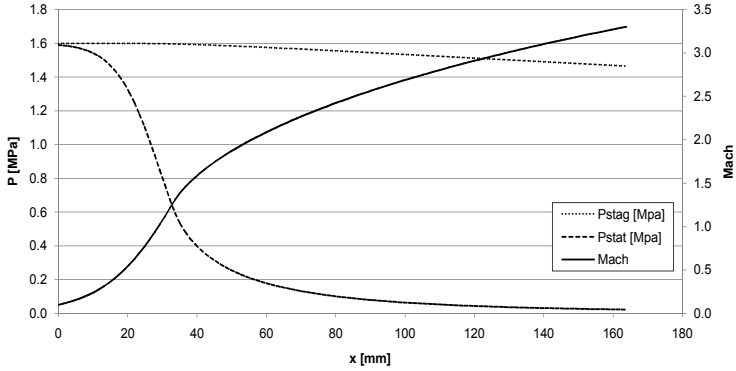

Figure 4.8: Pressure profile along the primary nozzle

called mixing chamber shown in fig.(4.9).

Mixing process is assumed to occur from the exit plane of primary nozzle onwards through the surface of an expansion cone whose apex angle is θ_p.

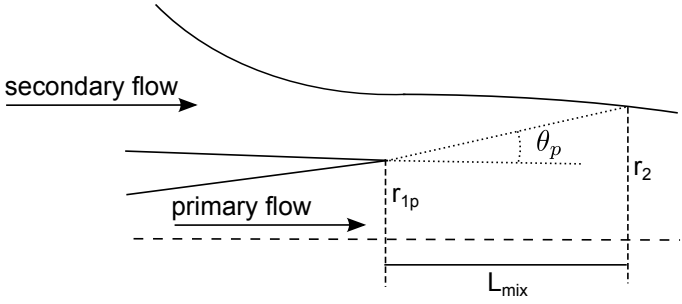

Figure 4.9: Mixing chamber section

Conservation of momentum yields:

$$\int AdP = \dot{Q}_2 - \dot{Q}_1 = (m_p + m_s)c_2 - m_p c_{1,p} - m_s c_{1,s} \qquad (4.52)$$

where subscripts p and s are referred to primary and secondary flow, while subscripts 1 and 2 are referred to inlet and outlet of

the mixing zone. Assuming constant pressure mixing, first term of (4.52) vanishes and introducing entrainment ratio ω

$$c_2 = \frac{c_{1,p} - \omega c_{1,s}}{1 + \omega} \tag{4.53}$$

Mixing efficiency can then be considered obtaining

$$c_2 = \sqrt{\eta_{mix}} \frac{c_{1,p} - \omega c_{1,s}}{1 + \omega} \tag{4.54}$$

In the mixing chamber is

$$\frac{d\dot{Q}}{dx} = 0 \tag{4.55}$$

In a generic section at distance x from the nozzle exit plane, being m_x the entrained mass upstream of that section, the flow rates within the cone m_{jx} and outside of the cone m_{sx} are:

$$m_{jx} = m_p + m_x \tag{4.56}$$

$$m_{sx} = m_s - m_x \tag{4.57}$$

Hence, the momentum at x position is

$$\dot{Q} = m_{jx} c_{jx} + m_{sx} c_{1,s} = m_p c_{jx} + m_x c_{jx} + m_s c_{1,s} + m_x c_{1,s} \tag{4.58}$$

Using (4.55), it follows

$$m_p \frac{dc_{jx}}{dx} + c_{jx} \frac{dm_x}{dx} + m_x \frac{dc_{jx}}{dx} - c_{1,s} \frac{dm_x}{dx} = 0 \tag{4.59}$$

from which

$$(m_p + m_x) \frac{dc_{jx}}{dx} = (c_{1,s} - c_{jx}) \frac{dm_x}{dx} \tag{4.60}$$

and finally

$$\frac{dc_{jx}}{c_{1,s} - c_{jx}} = \frac{dm_x}{m_p + m_x} \tag{4.61}$$

Integrating both sides

$$\ln\left(m_p + mx\right) + \ln\left(c_{1,s} - c_{jx}\right) + C = 0 \tag{4.62}$$

where the arbitrary constant C can be determined by the boundary condition

$$c_{jx} = c_{1,p} \;\; at \;\; m_x = 0 \Rightarrow C = -\ln\left(c_{1,s} - c_{1,p}\right) - \ln\left(m_p\right) \tag{4.63}$$

Substituting in (4.62)

$$\ln\left(\frac{m_p + mx}{m_p}\right) = \ln\left(\frac{c_{1,s} - c_{1,p}}{c_{1,s} - c_{jx}}\right) \tag{4.64}$$

and finally c_{jx} can be found

$$c_{jx} = c_{1,s} + \frac{\left(c_{1,p} - c_{1,s}\right) m_p}{m_p + m_x} \tag{4.65}$$

To determine c_{jx}, explicit expression for m_x must be obtained. Starting from secondary mass flow rate

$$m_s = \rho_{1,s} c_{1,s} A_{1,s} = \rho_{1,s} c_{is} A_{is,mix} \tag{4.66}$$

where subscripts *is* stands for quantity relative to interface between primary and secondary flow represented by conical section. Being $r_{1,p}$ and r_2 radius of mixing chamber inlet and outlet respectively, surface of the interface can be expressed by

$$A_{is,mix} = \frac{r_2 - r_{1,p}}{\sin\theta_p} 2\pi \left(r_{1,p} + \frac{r_2 - r_{1,p}}{2}\right) = \frac{\pi\left(r_{1,p} + r_2\right)\left(r_2 - r_{1,p}\right)}{\sin\theta_p} \tag{4.67}$$

and hence the velocity across this area is

$$c_{is} = \frac{m_x}{\rho_{1,s} A_{is,mix}} \tag{4.68}$$

Being L_{mix} the length of the mixing chamber, for any $x < L_{mix}$

$$m_x = \rho_{1,s} c_{i,s} A_{ix,mix} = \rho_{1,s} c_{i,s} \frac{2\pi x}{\cos\theta_p} \left(r_{1,p} + \frac{x}{2} \tan\theta_p \right) \tag{4.69}$$

Substituting (4.68) in (4.69)

$$m_x = \frac{m_s}{A_{is,mix}} A_{ix,mix} = 2 m_s \frac{x \tan\theta_p \left(r_{1,p} + \frac{x}{2} \tan\theta_p \right)}{(r_{1,p} + r_2)(r_2 - r_{1,p})} \tag{4.70}$$

Once m_x is obtained, c_{jx} can be computed by (4.65).

About design program, to model supersonic diffuser its inlet conditions coinciding with mixing chamber outlet ones have to be known. Applying (4.52), conservation of momentum gives the final velocity simply as a weighted average of primary and secondary velocities reduced by friction losses through efficiency η_{mix}:

$$c(23) = \eta_{mix}^{0.5} \frac{c(24) + \omega\, c(22)}{1 + \omega} \tag{4.71}$$

Being mixing process adiabatic, energy balance gives the enthalpy:

$$h(23) = \frac{h(24) + \frac{c(24)^2}{2} + \omega \left(h(22) + \frac{c(22)^2}{2} \right)}{1 + \omega} - \frac{c(23)^2}{2} \tag{4.72}$$

From $P(23)$ and $h(23)$ temperature, density, and flow area at end of the mixing zone are found.

4.8 Diffuser design: the CRMC method

The diffuser is designed according to *Constant Rate of Momemtum Change* (CRMC) method proposed by Eames [12]. Even if a shock free process may be difficult to achieve, the CRMC design, at least when the ejector operates at design conditions, should reduce the irreversibility due to the normal shock within a constant section mixing chamber, which nowadays represents the standard design geometry of ejector for refrigeration plant. It has to be pointed out that the method represents a design tool permitting to obtain

geometry description of supersonic diffuser imposing rate of momentum change. It is hence the central element of the design program whose aim it to obtain geometric definition of the ejector starting from its operating conditions. *CRMC* method, as proposed in literature, does not provide a specific design for the mixing chamber, it is based on isentropic assumption and does not limit the angle of the divergent stretch of the diffuser. In this work, a specific criterion to design mixing chamber is adopted (§4.7) and friction loss term is included in flow equations. Moreover, diffuser exit angle has been limited to avoid flow separation.

Complete mixing is assumed at diffuser inlet (23). The mixed flow, still supersonic, claims for a converging–diverging profile to complete pressure recovery. Specific function is developed to shape supersonic diffuser, whose purpose is to give pressure and enthalpy at diffuser exit as a function of initial conditions, relative roughness, maximum angle of the divergent duct and condenser inlet diameter.

The diffuser is divided in two stretches: the first is modelled fixing the rate of momentum change and deriving the section profile, while the second is a conical divergent section duct. Distinguishing two stretches, permits to contain apex angle of conical duct limiting flow detachment phenomena. In fact conical section for the second part of the diffuser is introduced to avoid too much rapidly diverging profile at the end of the diffuser itself, and it starts when the diverging angle exceeds a threshold passed to the function as input parameter.

The function gives the diffuser outlet pressure P_{new} and enthalpy $h(13)$ for the initial conditions $P(23)$, $c(23)$, $h(23)$ and radius r_{in} at the inlet section. Relative roughness is used to find the friction factor value. Maximum angle of the divergent duct and final diameter (i.e. condenser inlet diameter) determine the conical stretch geometry. Both the stretches are solved as variable section ducts with friction, but no heat transfer, work or mass addiction. The First and Second Law of Thermodynamics for an infinitesimal open

system, per unit mass flow rate, yield:

$$c\ dc + g\ dz + dh = \delta q - \delta l \tag{4.73}$$

$$ds = \frac{dq}{T} + ds_{irr} \tag{4.74}$$

Eliminating dq and evaluating enthalpy variation dh along any reversible transformation yield

$$c\ dc + g\ dz + \frac{dp}{\rho} + T\ ds_{irr} + \delta l = 0 \tag{4.75}$$

In the case at hand, the term $dR = T\ ds_{irr}$ represents the energy loss due to friction within the viscous flow. Dimensional analysis of the friction loss along a straight pipe yields:

$$R = R(L; D; c; \mu; \rho) = f_D \frac{L}{D} \frac{c^2}{2} \tag{4.76}$$

where Darcy friction factor f is a function of relative roughness ϵ/D and Reynolds number $Re = c\ D/\nu$. A closed form evaluation of f_D can be obtained by the Serghides formula [50]:

$$A = -2 \log \left(\frac{\epsilon}{3.7\ D} \frac{12}{Re} \right) \tag{4.77}$$

$$B = -2 \log \left(\frac{\epsilon}{3.7\ D} \frac{2.51\ A}{Re} \right) \tag{4.78}$$

$$C = -2 \log \left(\frac{\epsilon}{3.7\ D} \frac{2.51\ B}{Re} \right) \tag{4.79}$$

$$f_D = \left(A - \frac{(B-A)^2}{C - 2\ B + C} \right)^{-2} \tag{4.80}$$

valid within 0.0023% for $0.00004 < \epsilon/D < 0.05$ and $2500 < Re < 10^8$. If the variation of duct transverse area is not too abrupt, the relation may be applied also to variable area ducts. The term gdz is negligible for low density fluids. Work term δl is null for a fixed duct. The term dp/ρ claims for a relation between pressure and

density. To this aim, sound speed $a^2 = dp/dq$ can be used, yielding:

$$c\,dc + a^2\,\frac{d\rho}{\rho} + dR = 0 \tag{4.81}$$

Differentiation of mass flow rate conservation $m = \rho c \Omega$ and substitution in (4.75) gives:

$$\frac{d\Omega}{\Omega} = \left(M^2 - 1\right)\frac{dc}{c} + \frac{dR}{a^2} \tag{4.82}$$

Differentiating 4.77 (though this may be questionable, the equation being intended as an integral evaluation along a straight pipe), one has:

$$dR = \frac{f_D}{D}\frac{c^2}{2}dx \tag{4.83}$$

and substituting it in 4.82 finally we obtain

$$\frac{d\Omega}{\Omega} = \left(M^2 - 1\right)\frac{dc}{c} + \frac{M^2}{2}\frac{f_D}{D}dx \tag{4.84}$$

Integrating this equation geometric or velocity profile along the diffuser can be obtained knowing the other one.

Pressure at condenser inlet $P(13)$ is equal to $P(14)$ plus the pressure loss across the de-superheating part $\Delta P_{con}/4$. Pressure calculated by diffuser design function is different from the condenser pressure by an amount ΔP, so iteration is performed until convergence. The entrainment ratio is increased or decreased by a quantity $\Delta\omega$ according to the sign of ΔP. $\Delta\omega$ is reduced if the sign of ΔP has changed from previous iteration, meaning that the solution has been overcome. Once the iterative loop has converged, water temperatures and flow rates across the generator ($\dot{m}_{gen}$), condenser ($\dot{m}_{con}$) and evaporator ($\dot{m}_{eva}$) can be evaluated. Flow areas at diffuser inlet $\Omega(23)$, secondary flow throat $\Omega(22)$, nozzle exit $\Omega(24)$ and nozzle throat $\Omega(25)$ are found. Final results are cooling capacity P_{cw}, pump power P_p, power rejected to condenser P_{con}, COP, Carnot COP and Second Law efficiency η_{II}.

4.8.1 CRMC stretch

Assuming Constant Rate of Momentum Change as in [12], velocity derivative is a constant obtained fixing the length of the stretch and outlet velocity. From velocity change the area variation can be derived integrating the equation 4.84, which rearranged becomes

$$\frac{d\Omega}{dx} = \left(M^2 - 1\right) \frac{\Omega}{c} \frac{dc}{dx} + \frac{M^2}{2} \frac{\Omega}{D} f_D \tag{4.85}$$

which can be numerically solved using fourth order Runge–Kutta method [46]. In order to avoid discontinuity along duct profile the diffuser must be connected to mixing chamber assuring derivative continuity. Being area and velocity derivative linked together according to (4.84), fixing inlet section and its derivatives to smoothly connect with the mixing chamber imposes an initial velocity change rate. Starting from this value, velocity derivative is varied until a constant value. This behaviour is shown in fig.(4.10).

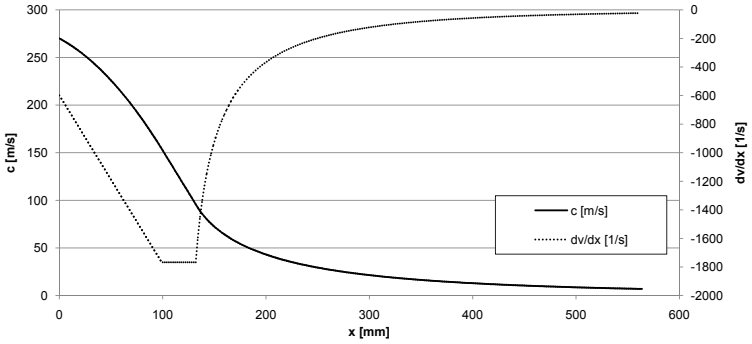

Figure 4.10: Velocity derivative profile along supersonic diffuser

Solving 4.85 simultaneously with continuity equation

$$\frac{1}{\rho} \frac{d\rho}{dx} + \frac{1}{c} \frac{dc}{dx} + \frac{1}{\Omega} \frac{d\Omega}{dx} = 0 \tag{4.86}$$

and energy conservation

$$\frac{dh}{dx} + c\frac{dc}{dx} = 0 \tag{4.87}$$

density and enthalpy along the supersonic diffuser can be obtained. From them, NIST Refprop functions give the other thermodynamic properties (pressure, temperature, entropy, gas specific heat and fluid viscosity).

4.8.2 Conical stretch

If a threshold is set on the aperture angle of the diverging part of the diffuser, the above described procedure must be modified. As soon as the threshold angle is exceeded, the geometry will be imposed, for example by setting a constant value of the diverging angle. Rearranging (4.84), fluid velocity derivative can be expressed as function of area change

$$\frac{dc}{dx} = \frac{1}{M^2 - 1}\frac{c}{\Omega}\frac{d\Omega}{dx} + \frac{M^2}{2\left(M^2 - 1\right)}\frac{c}{D}f_D \tag{4.88}$$

Fluid density and enthalpy are hence obtained simultaneously solving (4.88) together wiht (4.86) and (4.87). Other thermodynamic properties (pressure, temperature, entropy, gas specific heat and fluid viscosity) are computed by NIST Refprop functions. Finally, isentropic efficiency of the diffuser can be obtained dividing the static enthalpy difference along isentropic flow by the same difference computed with friction. No arbitrary assumption, like those done for the efficiency of primary/secondary expansion and mixing, is needed.

4.8.3 Diffuser design results

Main results of previously described procedure are the definition of the internal shape of supersonic diffuser and the computation of gas properties along the duct. Graphical presentation of the results can

be found in fig.(4.11) and fig.(4.12).

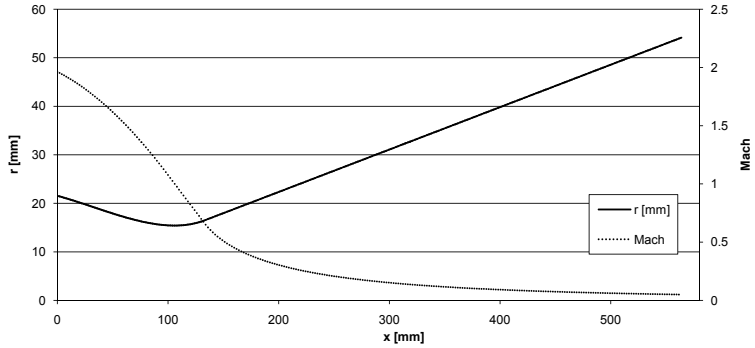

Figure 4.11: Radius (not in scale) and Mach number of supersonic diffuser along x

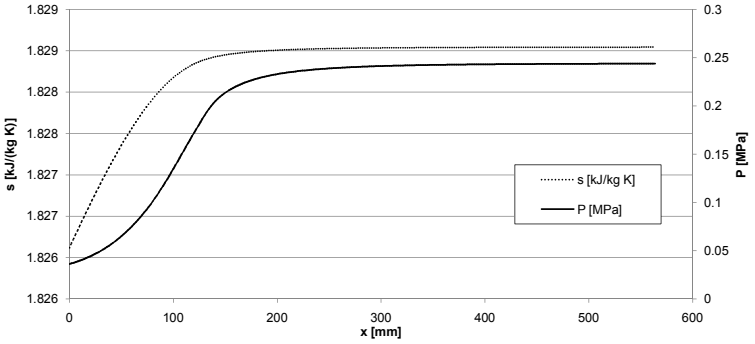

Figure 4.12: Behaviour of static pressure and entropy of supersonic diffuser along x

Analysing fig.(4.12), it emerges that CRMC section is very short when compared with the conical section, due to the combined effect of the assumed CRMC length (150 mm, see tab.(4.5)) and of the maximum allowable cone angle (5°). Accordingly, the Mach number, as well as static temperature and pressure, experience rapid change in the first part of the diffuser and a very slow variation afterwards. Fig.(4.12) shows clearly how entropy increase, due to friction losses, is concentrated in the first part of the duct. Even

if the cone is very long, being velocity fairly small in this zone, its contribution to the total loss is minimal.

4.9 Design program results

Design program can be executed with different values for input data and refrigerants and results are presented which are obtained by the reference data set reported in tab.(4.5) and used to guide prototype plant realization. No SLHX is provided, as explained in (§4.10), and resulting COP is 0.32. As said heat exchangers characteristics are evaluated outside the simulation code using a selection program provided by the manufacturer.

Fluid R245fa		
$W_{gen} = 120$ kW	$\Delta T_{gen} = 7$ °C	$\Delta P_{con} = 2.3$ kPa
$T(1) = 115$ °C	$\Delta T_{con} = 1$ °C	$\Delta P_{eva} = 3.6$ kPa
$T(4) = 35$ °C	$\Delta T_{eva} = 2$ °C	$\Delta P_{gen} = 19$ kPa
$T(7) = 12$ °C	$\Delta T_{suph,gen} = 2$ °C	$D_{gen} = 25$ mm
$T(8) = 7$ °C	$\Delta T_{suph,eva} = 8$ °C	$D_{s,int} = 35$ mm
$\eta_p = 0.95$	$\Delta T_{subc} = 3$ °C	$D_{s,ext} = 108.3$ mm
$\eta_s = 0.98$	$m(4) = 11.5$ kg/s	$D_{con} = 108.3$ mm
$\eta_{mix} = 0.85$	$m(1) = 9.5$ kg/s	$l_{CRMC} = 150$ mm
$\eta_p = 0.75$	$c_0 = 10$ m/s	$\alpha_{max} = 5°$

Table 4.5: Design program input data

About mixing efficiency η_{mix} its value is assigned base on literature analysis [14], while efficiency of primary and secondary nozzle and diffuser are computed by ideal gas and real gas model respectively. Temperature level at evaporator are fixed following standards for industrial and air conditioning applications, while generator and condenser temperature are determined according to generator size and climatic conditions foreseen for application.

A complete section profile of the ejector including primary and secondary nozzle, mixing chamber and supersonic diffuser is shown in fig.(4.13).

Tab.(4.6) summarizes main program outputs obtained starting from input data set of tab.(4.5)

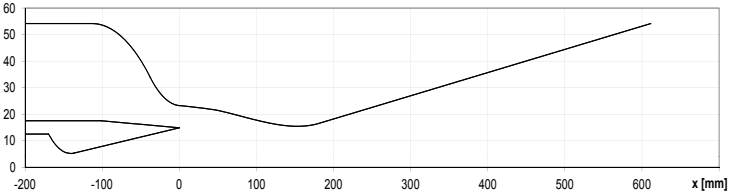

Figure 4.13: Profile of ejector obtained from design program (vertical axis not in scale)

ω - entrainment ratio	0.458	Primary throat Ø [mm]	10.3
COP	0.325	Primary exit Ø [mm]	29.4
2^{nd} Law efficiency	0.127	Diffuser throat Ø [mm]	30
Cooling capacity [kW]	39.2	Density at pump [kg/m^3]	1308
Cold water flow rate [kg/s]	1.87	Pump flow rate [m^3/h]	1.42
Primary flow [kg/s]	0.517	Pump ΔP [Mpa]	1.22
Secondary flow [kg/s]	0.237	Pump power [kW]	0.93
Power to condenser[kW]	160	Pump head [m]	94.8

Table 4.6: Design program output data

Being efficiency of mixing and of primary and secondary nozzle fixed, better performances are obtained by reducing friction losses in diffuser. The model considers uniform flow with local friction coefficient given by (4.77) and then friction losses is proportional to duct length and square of velocity. Reducing length of CRMC stretch, along which flow presents higher velocity and then higher friction losses, results in better ejector performance.

4.10 Sensitivity analysis

Sensitivity analysis is reported to verify impact of some hypothesis on the design and to point out which parameters mainly affect plant performance. Starting from reference value of tab.(4.5), single parameter are changed one at a time and its effect on results evaluated.

Pressure drops in heat exchangers

Pressure drop distributions within the heat exchangers are di-

vided between the different section of the exchangers themselves, trying to take into account for realistic pressure loss distribution. Anyway, the effect of these assumptions on the final result is rather small. For example, if we double the pressure drop during superheating (19-20 in fig.(4.1)) at fixed ΔP_{eva} no change is visible on the model results. Again, about condenser, changing to 1/2 (condensing) and 1/2 (de-superheating) total pressure drops modifies the COP by about 0.3%.

Temperature differences in heat exchangers

Quite more relevant is temperature difference within the heat exchangers. Let the generator minimum temperature difference reduce to 2°C. The pressure loss at generator increases: let 0.03 MPa be the new value. The results are shown in tab.(4.7). The COP increases by a 5%. The design of the diffuser, as well as the other geometric parameters, doesn't change significantly.

ω - entrainment ratio	0.485	Cold water flow rate [kg/s]	1.97
COP	0.340	Power rejected to condenser[kW]	162
2^{nd} Law efficiency	0.094	Pump flow rate [m^3/h]	1.34
Cooling capacity [kW]	41.2	Pump ΔP [Mpa]	1.60
Primary flow [kg/s]	0.492	Pump power [kW]	1.18
Secondary flow [kg/s]	0.239	Pump head [m]	123

Table 4.7: Design program output for different ΔT_{gen}

Mixing efficiency

Let now the mixing efficiency increase to 0.9 with all other parameters fixed at their reference values of tab.(4.5). As clearly comes out from tab.(4.8) results, this variation produces a remarkable effect on the system performance. COP increases by a 23%. This shows that a reliable forecast of the system performance is impaired by the uncertainty on η_{mix}.

CRMC lenght

Another relevant parameter is the CRMC length. If the length increases to 300 mm, fixing all other parameters, the simulation results change as shown in tab.(4.9). The performance is severely decreased. The design of the ejector however is less unconventional.

ω - entrainment ratio	0.557	Cold water flow rate [kg/s]	2.28
COP	0.394	Power rejected to condenser[kW]	169
2^{nd} Law efficiency	0.109	Pump flow rate [m^3/h]	1.35
Cooling capacity [kW]	47.7	Pump ΔP [Mpa]	1.42
Primary flow [kg/s]	0.497	Pump power [kW]	1.11
Secondary flow [kg/s]	0.277	Pump head [m]	110

Table 4.8: Design program output for different η_{mix}

The throat is longer and less steep.

ω - entrainment ratio	0.340	Cold water flow rate [kg/s]	1.39
COP	0.241	Power rejected to condenser[kW]	150
2^{nd} Law efficiency	0.066	Pump flow rate [m^3/h]	1.35
Cooling capacity [kW]	29.2	Pump ΔP [Mpa]	1.42
Primary flow [kg/s]	0.497	Pump power [kW]	0.96
Secondary flow [kg/s]	0.169	Pump head [m]	110

Table 4.9: Design program output for longer CRMC stretch

SLHX

A last issue is the SLHX. In all previous simulations SLHX is absent. In order to simulate its presence, one has to assume an efficiency and two pressure losses (on the liquid and on the vapour side). Let 0.5 be the SLHX efficiency and 2 kPa be the pressure loss on both sides. Results are summarized in tab.(4.10). It may be seen that the COP decreases with respect to the case without SLHX. Therefore, as far as we rely on the simulations results, an SLHX could be used only in order to cool the liquid at condenser exit if pump cavitation problems were experienced. The Ts diagram in fig.(4.14) shows how points 16 and 17 are now separated by the liquid cooling within the SLHX. Accordingly, point 21 is now shifted upward and rightward with respect to point 20. Being point 23 properties a weighted average of the properties in points 24 and 22, a temperature increase in point 24 will produce a corresponding movement of point 23 and hence of the whole compression within the ejector. Moving towards the right increases the distance between to isobaric curves and hence the work needed to compress the fluid. The increased work is not fully compensated by the slightly

increased cooling capacity, i.e. the slight movement of points 17 and 18 to the left. Therefore, the net effect on the COP is negative.

ω - entrainment ratio	0.434	Cold water flow rate [kg/s]	1.78
COP	0.309	Power rejected to condenser[kW]	158
2^{nd} Law efficiency	0.085	Pump flow rate [m^3/h]	1.34
Cooling capacity [kW]	37.4	Pump ΔP [Mpa]	1.42
Primary flow [kg/s]	0.492	Pump power [kW]	1.01
Secondary flow [kg/s]	0.213	Pump head [m]	110

Table 4.10: Design program output for longer CRMC stretch

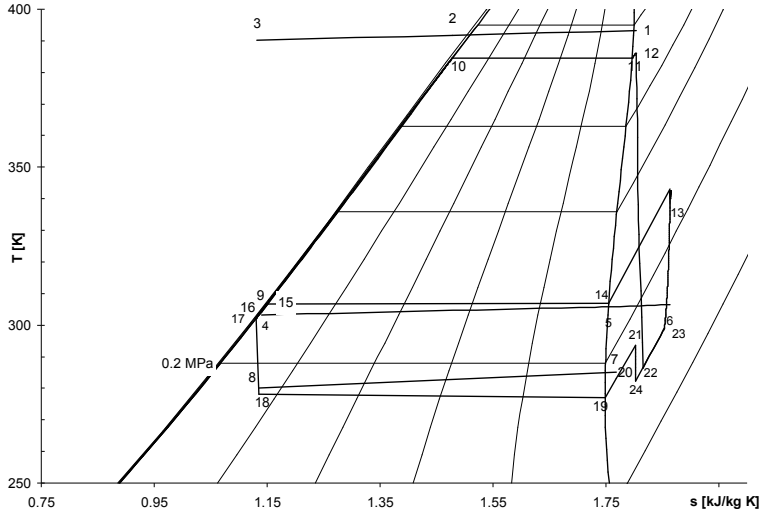

Figure 4.14: Ts diagram for the ejector cycle with R245fa and SLHX

Chapter 5

Experimental results

In this chapter experimental results, obtained by prototype plant realized at the test facility, set up in manufacturer plant of Frigel Firenze Spa, are presented.

Ejector refrigeration apparatus is described focusing on main building feature and measurement equipment in the first part of the chapter and then data analysis is presented and results evaluated.

5.1 Experimental apparatus

The results of a study of potential working fluids lead to R245fa being selected as the most suitable fluid, because of its relatively high critical temperature, moderate system pressure ratio, positively sloped saturated vapour curve, (that gives a dry expansion even starting with saturated vapour) and a reasonable GWP. The saturation temperatures at the generator, condenser and evaporator selected are 106°C, 39°C and 4°C respectively at design point. From design program results (§4), it is estimated that the jet-pump entrainment ratio would be approximately 0.44 and this give a system COP value of about 0.31. Therefore, for an evaporator cooling capacity of about 40 kW the vapour generator would require approximately 130 kW of heat input.

Starting from these data, plant components have been selected and found under a collaboration with Frigel Firenze Spa, at whose test facility assembly has been made and experimental tests performed.

According to research interest, plant is equipped with measurement instrumentation, which permits to obtain pressure trend along the ejector and working conditions of plate heat exchangers. A detailed description of used instrumentation and probes, and their placement over the plant are reported in §5.1.4.

Moreover, prototype plant is designed to permit substitution of primary nozzle and diffuser, to test different ejector solution. All other parts of the plant are unchanged and it is then possible to compare some geometric solution for ejector on the same configuration. Actually, two different diffuser profile are tested, obtained by two different building process and materials (§5.1.2).

5.1.1 Plant layout

Main components of the prototype plant are the three heat exchangers, ejector and feed pump (fig.(5.1)). An electronic expansion valve (EEV) is used to control liquid level in the evaporator. During components placing, several features have to be taken into account like to avoid feed pump cavitation and heat exchangers thermal losses, to position measurement probes, to permit easy connection with external water circuits and to allow access to diffuser for assembly and disassembly operations.

Vertical arrangement is chosen for heat exchangers placement, partly to reduce floor space occupied by the plant, but mainly to mitigate any flow cavitation problems at pump inlet, being in this way possible to place condenser in the upper part of the structure and pump at the bottom, increasing condensed refrigerant head. Consequently, main axis of ejector also has to be vertically aligned. The main drawback of described configuration is the elbow connecting the end of the diffuser to the condenser inlet. This elbow should have an adequate radius and a suitable straight connecting

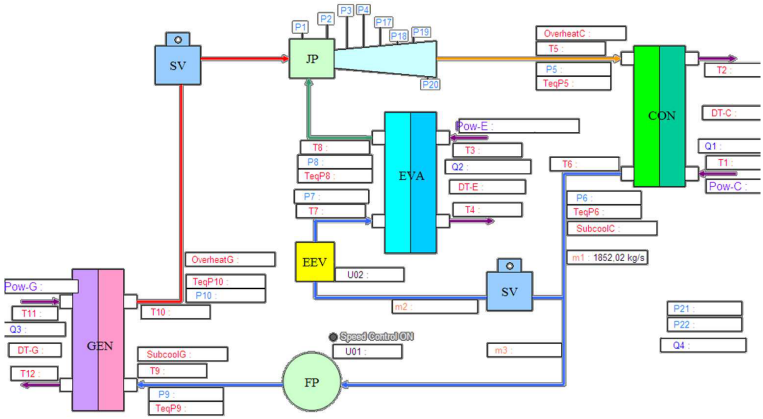

Figure 5.1: Scheme of the prototype plant extracted from plant control and monitoring program

piece upstream. Actually, the fluid velocity in this zone should be moderate (less than 10 m/s) and therefore the flow should not suffer significantly.

Behind the frame, from top to bottom, condenser, evaporator and generator, are seen (fig.(5.2)). Each exchanger rests on a couple of beams and is fixed to another couple of beams on the front. The connections for R245fa are on the front, while those for water are on the back, in order to facilitate the connection with the hot, cold and ambient water circuits. The pump stands on the base.

A remarkable result is the very high head and relatively low power of the pump. This makes the pump selection difficult, because normal units of such high head have usually a much higher power. A multistage centrifugal, magnetically driven, variable speed feed-pump is chosen.

5.1.2 Ejector description

Ejector is composed by different functional sections which, as said, can be identified as primary and secondary nozzle, mixing chamber and supersonic diffuser. From the building point of view, necessity

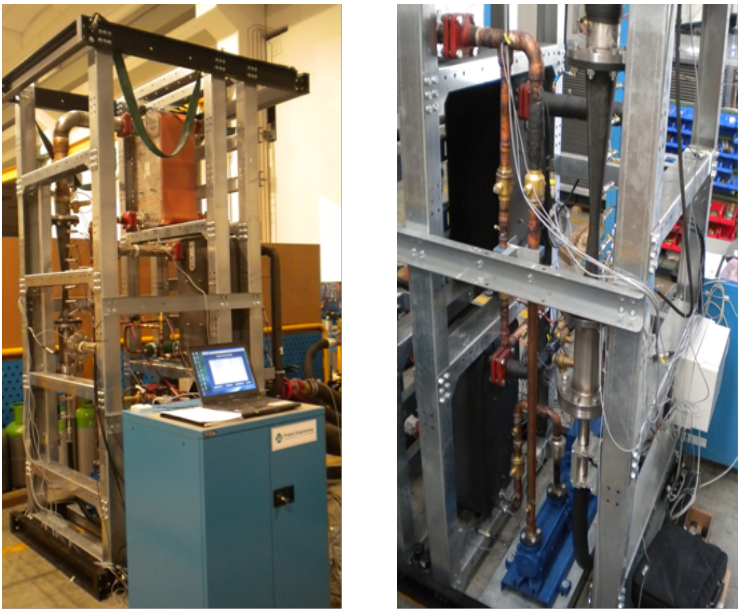

Figure 5.2: Photos of prototype plant from different sides, at Frigel Firenze Spa test facility

of components replacement and adjustment have to be taken into account. In particular, starting from a fixed Tee standard piece, connected on one side with the duct arriving from evaporator, secondary flow inlet, on other side with the diffuser and on the last one with the sliding tube of the primary nozzle, all other parts of the ejector are specifically designed.

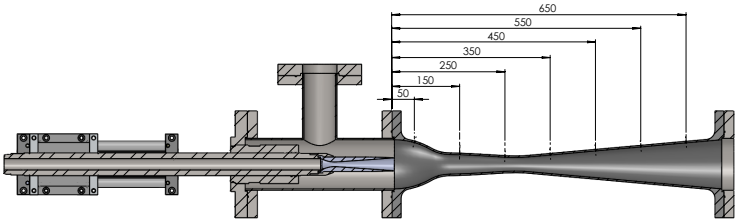

Figure 5.3: Complete design of ejector, including Tee piece, slit of the primary nozzle and diffuser

Primary nozzle is made by aluminium piece, worked to obtain internal and external profile defined according to design program model. The nozzle is fixed to a sliding tube, which can be moved through a commercial screw-operated slide in order to vary its axial position. On the other end, the tube is connected to a flexible pipe which supplies primary fluid from the generator. Flexible pipe enter the Tee piece from the bottom, while secondary fluid duct is linked to it horizontally.

On the upper side of the Tee, mixer-diffuser duct is mounted using a specifically realized flange. During description of the design program, mixing chamber and supersonic diffuser are ideally divided in two distinct sections along the fluid flow. From the point of view of their realization, they represent a single continuous duct whose internal surface is modelled to obtain desired section changes. Two solution for ejector mixer-diffuser duct are designed and realized whose main characteristics are summarize in tab.(5.1). The external shape may vary according to the needs of the production process, component strength and measurement probes seats.

First solution, realized in composite material, is completely mod-

	Ejector A	Ejector B
Area ratio	8.6	13.2
Diffuser throat diameter [mm]	30	37.1
Diffuser throat position [mm]	270	302.5
Diffuser length [mm]	735	735
Diffuser material	Carbon fibre	Aluminium

Table 5.1: Comparison of different solutions for mixer-diffuser duct realization

elled according to results of the design program. Composite material has the advantage of very high flexibility in surface shape modelling, good strength and possibility to realize the whole mixer-diffuser duct on a single piece. On the other hand, it presents quite high development cost to realize the mould which can be reduced only for quite large scale production. Surface roughness is quite low, ensuring low friction losses.

A difficult feature using composite material is the realization of the holes necessary to mount pressure probes. They require in fact a further piece manufacturing which has to be done accurately to avoid internal surface damaging and carbon fiber displacement. In fig.(5.4) an image of the composite material made diffuser is shown.

A second mixer-diffuser duct solution is proposed and realized, whose main geometric characteristics are summarized in tab.(5.1). It is based on a hybrid design procedure which maintains the *Constant Rate of Momentum Change* criterion for the design of the diffuser but with different inlet conditions determination. Inlet conditions of the supersonic diffuser are represented by the outlet ones of the mixing chamber. Mixing chamber design is probably the most critical part of the design program, being hypothesis on mixing efficiency and separation line between mixing fluids necessary to define duct external profile for that stretch. Moreover, no oblique shocks are taken into account in the design program, overestimating pressure values at the exit of the secondary nozzle and so affecting mixing chamber design.

It is decided to realize a diffuser using *CRMC* method starting

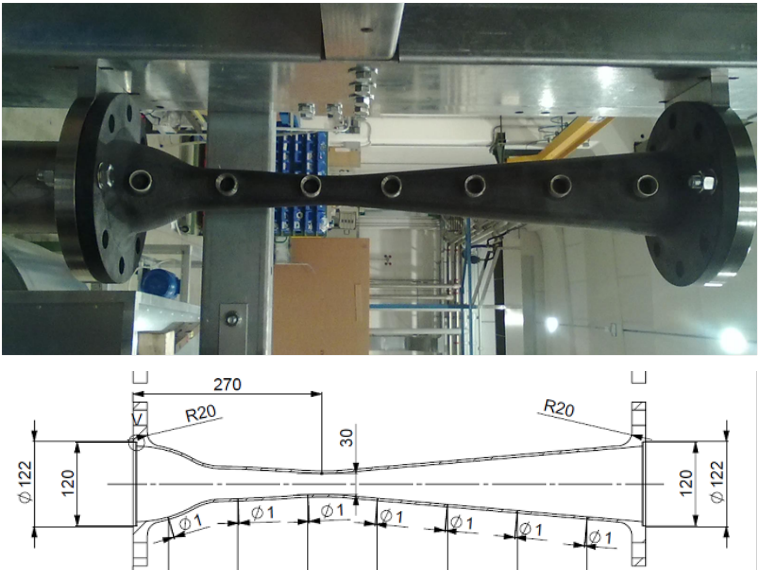

Figure 5.4: Photo and design of solution A of supersonic diffuser

from *CFD* results as its inlet conditions shown in tab.(5.2). Mean of the values obtained by numerical simulation over the outlet section of the mixing chamber, whose geometry is identical to that previously considered, is used to define inlet conditions of the diffuser and then procedure described in §4.8 is adapted to consider those inlet conditions.

	T [K]	P [kPa]	$\dot{m}$ [kg/s]
Primary nozzle inlet	379.25	1445.00	0.520
Secondary nozzle inlet	284.20	63.00	0.244
Mixing chamber outlet	326.40	44.77	0.764

Table 5.2: Inlet conditions for primary and secondary nozzle and outlet conditions at the exit of mixing chamber obtained by CFD simulation

Finally a mixer-diffuser duct with the same mixing chamber geometry but different supersonic diffuser one, even if based on the same design criterion, is obtained (fig.(5.5)).

Two metal pieces are shaped and connected to realize second version of mixer-diffuser duct. Use of aluminium pieces permits to reduce prototype costs and time, so it is selected for the second solution proposed for the diffuser.

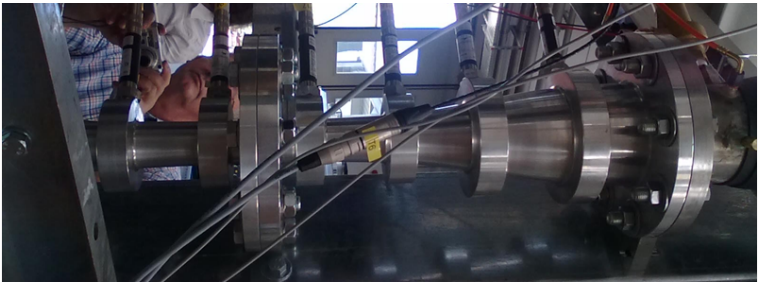

Figure 5.5: Photo of solution B of supersonic diffuser

5.1.3 Plate heat exchangers selection

Plate heat exchangers by GEA GmbH are used for the generator, evaporator and condenser. Their performance is modelled using a manufacturer's heat exchanger selection software. This provides data on their heat transfer capacity and pressure losses as well as giving the dimensions the heat exchangers. The generator is designed to be heated by water at 115°C produced by a system of seven electric heaters providing a maximum heat input of 154 kW, which is shown in fig.(5.6).

Characteristics of the selected plate heat exchangers are summarized in tab.(5.3)

	Generator	Evaporator	Condenser
Model	AE10-110	WP10L-250	WP10-120
Nom. temperature (R245fa) [°C]	106	5	39
Nom. pressure (R245fa) [bar]	14.6	0.6	2.4
Mass flow rate (R245fa) [kg/s]	0.52	0.36	0.88
Mean temperature difference [°C]	7	4.6	2.8
Nominal power [kW]	150	60	185

Table 5.3: Main characteristics of the selected plate heat exchangers

Figure 5.6: Water heating system for generator supply circuit

5.1.4 Measurement equipment

Measurement equipment is provided for the prototype plant which permits on one hand to give parameters to the control logic and on the other hand to validate design hypothesis and numerical simulation results.

Piezoresistive pressure transducer, produced by Keller and calibrated by the manufacturer for the employment range, are used to obtain pressure values. Two pressure probes are placed at inlet and outlet connection of each plate heat exchanger, one at the inlet of the Tee piece upstream secondary nozzle and seven along mixing chamber and diffuser external profile. These last probes are fixed on static pressure ports realized perpendicular drilling the duct profile and inserting a threaded cup at each port to mount sensor.

Temperature are obtained with Pt100 resistance thermometers, read by acquisition system and placed at inlet and outlet connection of heat exchangers.

Endress+Hauser Promag electromagnetic flow meters are used to measure all water flows of the external circuits.

Pump electric power consumption is measured by an electronic wattmeter.

A data acquisition system, developed by Frigel Firenze Spa, is used to record and store measured data and to calculate some relevant performance parameters. Data are registered every second and periodically saved in a file to avoid data losses, if acquisition is suddenly interrupted because malfunctioning.

5.2 Experimental results

Results described in this section are obtained by different test sessions performed during last year. Both ejector solution are evaluated, trying to replicate working condition of the refrigeration plant, compatibly with different climatic conditions (which affect condensation temperatures) and improvement made to acquisition system and test procedures.

For the same temperature and pressure levels at condenser, evaporator and generator, ejector refrigeration plant performances depend mainly of ejector design. Comparison of different ejectors with same boundary conditions permits to experimentally evaluate goodness of design criteria.

While two design solutions are proposed for diffuser, a single primary nozzle geometry is used. Primary nozzle geometry and position affect system performance [8], so, for the same primary nozzle geometry, its optimum is found and maintained during all tests, following procedure proposed by Eames *et al.* [13], for each ejector solution.

5.2.1 Positioning of NXP

Positioning of primary nozzle within the mixing chamber is important to determine performance of the system. Experimental tests were carried out to find the optimum nozzle exit position (NXP) for the ejectors under test. For each position, generator and evaporator heat rates were measured and system COP is calculated by them. Reference position (NXP = 0) is 120 mm downstream the junction

flange between Tee piece and the diffuser duct, which represents the position used by the design program to define ejector geometry. Fig.(5.7) shows COPs obtained for different positioning of the nozzle exit plane for both diffuser solutions adopted.

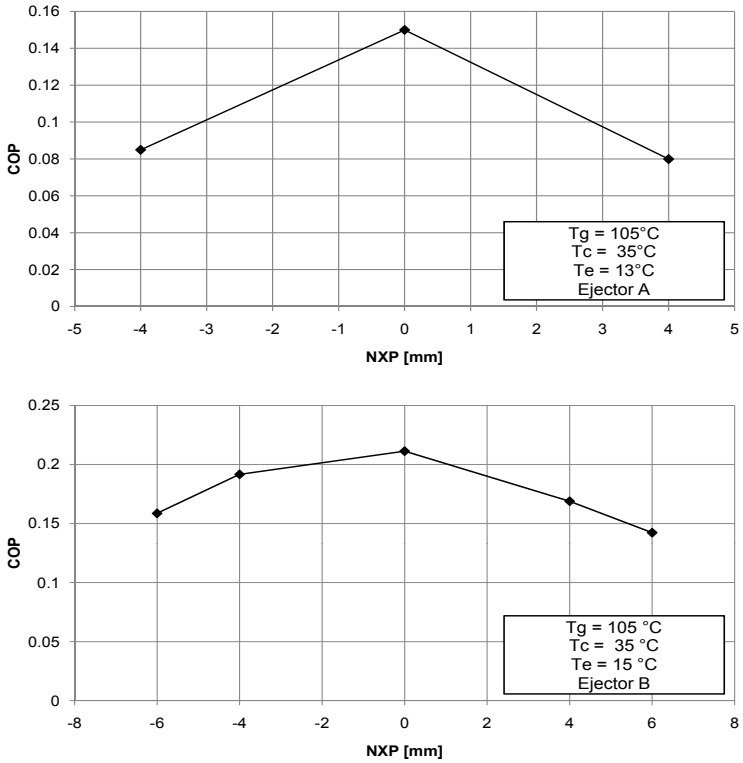

Figure 5.7: COP values for different positions of NXP for the two different types of diffuser

It has to be noted that best performances seems to be reached when reference position is set for primary nozzle exit plane.

5.2.2 Condenser pressure variation

With nozzles held at their design position, which, as seen in previous section, also represents their optimum position, condenser pressure

has been varied, maintaining evaporator and generator fixed. Generator temperatures were controlled by switching electrical heaters and three temperature levels were selected (100, 105 and 110°C), evaporator temperatures were changed from 7°C to 13°C.

Indeed some difficulties have arisen due to temperature level control, specially for condenser and evaporator. The first in fact provides water cooling for all refrigeration units under test at the facility and, during summer, air conditioning needs of the offices, so being difficult to maintain stable condensing temperature of the ejector refrigeration plant. For evaporator, controller of the water mass flow rate seems to respond too slowly to water temperature variations, so generating instabilities in fluid evaporation temperature. These instabilities do not appear to be due to refrigerant power variations caused by fluid side phenomena, being them present even when plant parameters are stable.

Representative working points of the plant are then the results of a reduction procedure, by first filtering data based on temperature values at evaporator and generator and condenser pressure and then time-averaging them over the selected intervals. Effects of this procedure on precision of presented values is analysed in §5.2.4.

Figg.(5.8) and (5.9) show COP changes ad function of condenser pressure obtained by experimental results respectively with solution A and B of the diffuser design. Both design solutions show the typical behaviour of ejector refrigerators, with a nearly constant COP, independent of condenser pressure, which is obtained for double chocking condition. For those condenser pressures range, secondary nozzle is chocked, in addition to the primary nozzle, being so its mass flow rate independent by outlet conditions. Growing condenser pressure, a critical pressure value is found over which system COP starts to decrease until pressure is such to reverse secondary flow. This *backflow* condition represents final limit of the plant working points.

Moreover it can be noted that higher COPs are obtained by increasing evaporator temperature and decreasing generator one. In

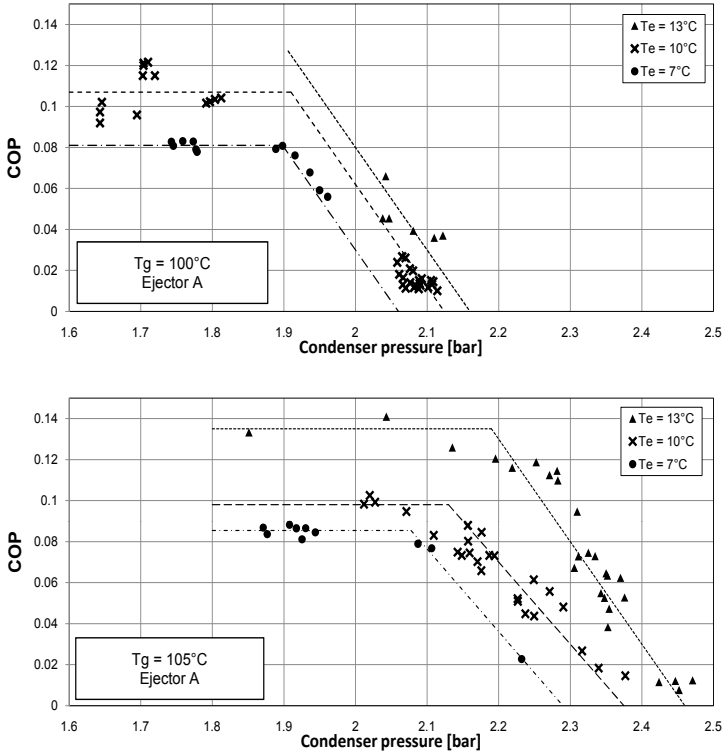

Figure 5.8: COP values for different condenser pressure for solution A of the diffuser

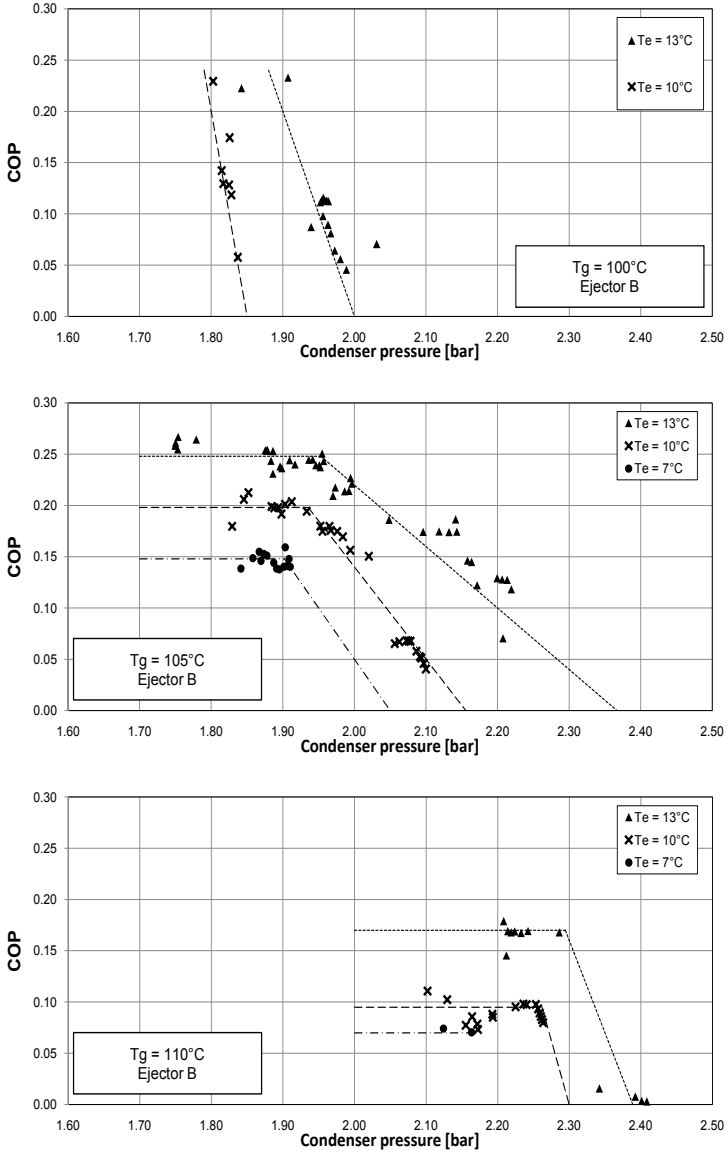

Figure 5.9: COP values for different condenser pressure for solution B of the diffuser

the first case increased COPs are associated with a slightly increase in critical pressure, while generator temperature reduction lowers the critical pressure, limiting plant working range in term of condenser pressure.

This can be better analysed looking at fig.(5.10), which summarizes COP values as a function of critical temperature for different values of evaporator and generator temperatures.

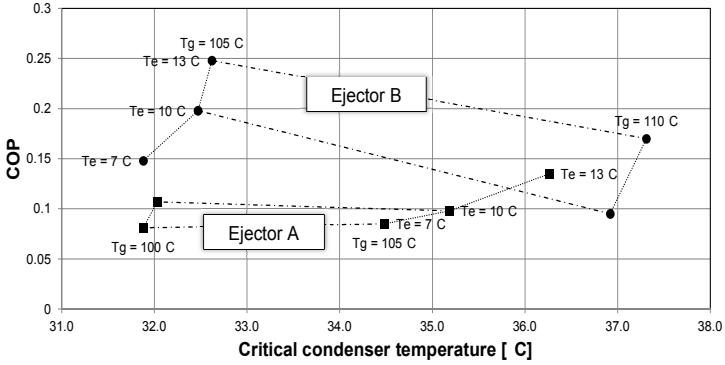

Figure 5.10: COP values as function of critical temperature for different evaporator and generator temperatures and both design solutions

Comparing behaviour of the two design solution, ejector B gives higher COP compared to ejector A, as expected when considering that the first has an higher area ratio. On the other hand it presents lower critical condenser pressure for the same generator and evaporator temperatures. Increased COP for the same generator conditions shows that higher secondary mass flow rate is accomplished by ejector B, limiting however its pressure ratio.

5.2.3 Pressure profiles along ejector

Pressure transducers, placed along mixing chamber and diffuser, measure static pressure values at the wall of the duct internal surface. These values can be compared with those expected from one-dimensional model of the design program to verify model assump-

tion and reliability. Fig.(5.11) shows the results of pressure measurements taken for different operating conditions.

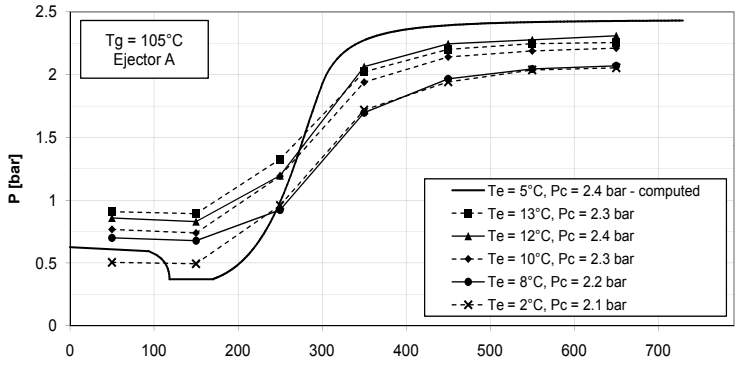

Figure 5.11: Wall pressure values along mixing chamber and diffuser compared to those expected by design program for solution A design

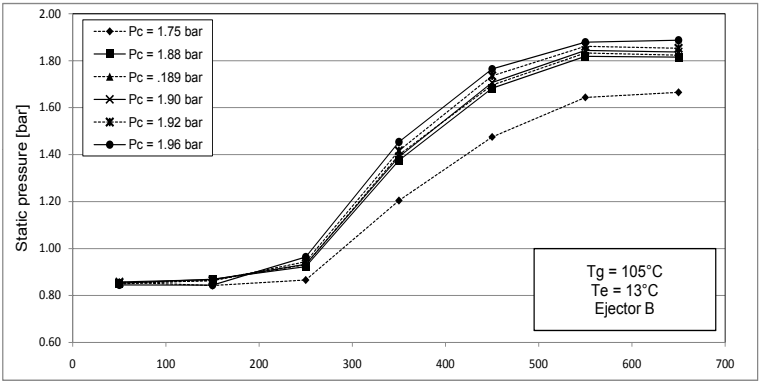

Figure 5.12: Wall pressure values along mixing chamber and diffuser for different condenser pressure and same temperatures at generator and evaporator

For ejector B, wall pressure profile is shown in fig.(5.12), which points out typical double chocking condition of the ejector for condenser pressure below its critical value. Changing condenser pressure modifies only second stretch of the diffuser, leaving the first three measured values unaffected.

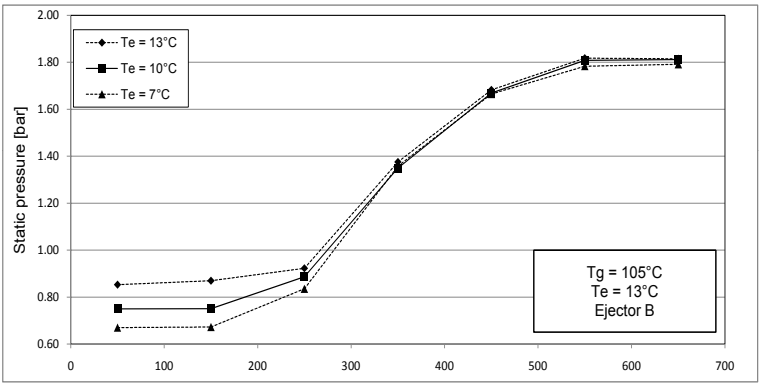

Figure 5.13: Wall pressure values along mixing chamber and diffuser for different values of evaporator temperature and same conditions at generator and condenser

Moreover, fig.(5.13) shows pressure values for the same condenser pressure and generator temperature but different values of evaporator temperature. Condenser pressure seems to strongly affect wall pressure profile after chocking condition is reached and shock probably happened, while weak dependence on evaporator temperature is shown.

5.2.4 Error analysis

As said in §5.2.2, plotted points are obtained by reducing data with a post-processing procedure. It is mainly based on a filtering over temperature values at generator and evaporator and on pressure values at the condenser, followed by time averaging of the data over the selected sets. Data filtering is necessary to refer COP values to the same test conditions, being difficult, during acquisition phases, to maintain stable operating conditions. While generator temperature level were indeed quite stable during plant operations, temperature controls showed some difficulties to stabilize temperature at evaporator and condenser, resulting in fluctuating values. Being plant performances dependent on condenser and evaporator

temperatures, set of values which presented same evaporating and condensing points over a time interval of some minutes have been selected.

Each single COP value is then the result of time-average of several instantaneous values, which takes into account for fluctuations. Two times the standard deviation associated to the time-average operation can be considered as error on each COP value [54]. Mean value of relative error associated to COP, considering all data sets for both design solutions, can be set to about 15%. It has to be noted that pressure and temperature measurement systems present significantly lower precision and accuracy errors than those resulting from fluctuating behaviour, so their contribution to overall error value on COP can be neglected.

Looking at wall pressure profiles along ejector, it seems they are less influenced by fluctuations of operating conditions, being static pressure values error within 2-3%. Each pressure point has been obtained as time-average over the same data intervals used for COP, so lower error can be interpreted as a reduced influence of the temperature fluctuations over flow behaviour at the duct wall, being probably their impact more relevant for the inner sections.

5.2.5 Results analysis

Measured values of COP are lower than those obtained by design program for diffuser solution A. The average measured value of COP is approximately 0.1 whilst the maximum value recorded is 0.12, measured with an evaporator temperature of 10°C(sat) and condenser temperature of 32°C(sat). For comparison, the simulation model estimates a COP of about 0.31 with evaporator at 5°C(sat) and the condenser at 40°C(sat). Plant mounting diffuser solution B shows very better results in term of COP, being values two times higher compared to solution A, for the same test conditions. It is then possible to conclude that modified *CRMC* method could be a good design criterion, even if some adjustments have to be done for the mixing chamber modelling. Solution A and B, in fact, differ only

for diffuser inlet conditions that are obtained as outlet conditions of the mixing stretch by one-dimensional model and CFD results respectively.

Looking at fig.(5.11), it seems that model used in design program gives too rapid pressure increase. Moreover pressure in mixing chamber is assumed to reach lower values than measured ones, being not considered oblique shocks starting from primary nozzle exit plane. Probably more complex modelling technique may be required to better match measured pressures and to obtain better agreement with COP values.

Because of the rapid variation in the chilled water temperature difference across the evaporator, precise evaluation of COP is difficult. This problem is thought to result from the large thermal inertia of the evaporator and the responsiveness of the chilled water flow control system. Although time-averaged measurements helped to smooth the data and gave more stable results COP data are still quite scattered.

The prototype refrigerator was shown to operate stably over a range of operating conditions. The minimum suction and maximum condenser pressures and temperatures achieved are acceptable to the proposed application. Feed pump behaviour is satisfactory, justifying vertical arrangement.

Chapter 6

Conclusions

During present work, an optimization program based on a multivariate approach is developed, which permits to define configuration of a whole ejector refrigeration system including heat exchangers and ejector. Program is capable to easily adapt to run with different working fluids, being properties computed with NIST REFPROP libraries included in the code.

Ejector models are proposed to design mixing chamber with a specific criterion and supersonic diffuser with a literature method (*CRMC*) modified introducing some innovative feature. Specifically, friction is included in flow equations contributing to define diffuser geometry itself.

Starting from program results and design models, two different solutions for ejector have been developed and experimentally tested in a prototype plant. An experimental ejector refrigeration system has been, in fact, realized at the test facility set up in manufacturer plant of Frigel Firenze Spa, which collaborates to system building and measurement instrumentation placement and calibration. System nominal refrigerant power is 40 kW, which is considered adequate for industrial temperature control application, but maximum of about 30 kW has been achieved with one of the two ejector solution mounted. The minimum suction and maximum condenser

pressures and temperatures achieved are however acceptable to the proposed application. The prototype refrigerator was shown to operate stably over a range of operating conditions.

One-dimensional models seems to be promising instrument for optimized development of ejection refrigeration plant. Many iterations required by multivariate optimization algorithms need in fact quite fast, but reliable ways to describe fluid behaviour through ejector.

Possibility to realize an ejector refrigeration plant, whose size is adequate to industrial application with a reasonable COP has been pointed out. Based on results obtained by this prototype, the development of a pilot plant can start and numerical simulation of the component validated.

Finally, some limits of the proposed models have been identified from the point of view of foreseen, compared to real behaviour of the ejector and of the evaluation of system performance. Starting from those limits some strategies have been recognized to improve models themselves.

Acknowledgements

Author wishes to thank Prof. Ian Eames of the University of Nottingham for his suggestions during ejector model definition and experimental tests organization.

Frigel Firenze Spa made prototype realization possible providing components, skills in plant assembly and control and test facility availability. Moreover, collaboration of Ing. Michele Livi and Ing. Filippo Malvolti was valuable during plant development and experimental data collection.

Author wishes to thank, for collaboration and results of CFD analysis produced, Ing. Simone Salvadori and Prof. Francesco Martelli.

Finally special thanks go to Prof. Giuseppe Grazzini and Ing. Adriano Milazzo for their suggestions and fruitful collaboration.

The prototype has been partly funded by Regione Toscana.

Bibliography

[1] J.M. Abdulateef, K. Sopian, M.A. Alghoul, and M.Y. Sulaiman. Review on solar-driven ejector refrigeration technologies. *Renewable and Sustainable Energy Reviews*, 13(6–7):1338–1349, 2009.

[2] S. Aphornratana and I.W. Eames. A small capacity steam-ejector refrigerator: experimental investigation of a system using ejector with movable primary nozzle. *International Journal of Refrigeration*, 20(5):352–358, 1997.

[3] Y. Bartosiewicz, Z. Aidoun, and Y. Mercadier. Numerical assessment of ejector operation for refrigeration applications based on cfd. *Applied Thermal Engineering*, 26(5–6):604–612, 2006.

[4] E. W. Beans. Computer solution to generalized one-dimensional flow. *Journal of Spacecraft and Rockets*, 7(12):1460–1464, 2009.

[5] L. Boumaraf and A. Lallemand. Modeling of an ejector refrigerating system operating in dimensioning and off-dimensioning conditions with the working fluids r142b and r600a. *Applied Thermal Engineering*, 29(2–3):265–274, 2009.

[6] M.J. Box. A new method of constrained optimization and a comparison with other methods. *The Computer Journal*, 8(1):42–52, 1965.

[7] K. Chunnanond and S. Aphornratana. Ejectors: applications in refrigeration technology. *Renewable and Sustainable Energy Reviews*, 8(2):129–155, 2004.

[8] K. Chunnanond and S. Aphornratana. An experimental investigation of a steam ejector refrigerator: the analysis of the pressure profile along the ejector. *Applied Thermal Engineering*, 24(2-3):311–322, 2004.

[9] K. Cizungu, A. Mani, and M. Groll. Performance comparison of vapour jet refrigeration system with environment friendly working fluids. *Applied Thermal Engineering*, 21(5):585–598, 2001.

[10] S. L. Dixon. *Fluid mechanics and thermodynamics of turbomachinery*. Butterworth-Heinemann, 1998.

[11] R. Dorantes and A. Lallemand. Prediction of performance of a jet cooling system operating with pure refrigerants or non-azeotropic mixtures. *International Journal of Refrigeration*, 18(1):21–30, 1995.

[12] I.W. Eames. A new prescription for the design of supersonic jet-pumps: the constant rate of momentum change method. *Applied Thermal Engineering*, 22(2):121–131, 2002.

[13] I.W. Eames, A.E. Ablwaifa, and V. Petrenko. Results of an experimental study of an advanced jet-pump refrigerator operating with r245fa. *Applied Thermal Engineering*, 27(17-18):2833–2840, 2007.

[14] I.W. Eames, S. Aphornratana, and H. Haider. A theoretical and experimental study of a small-scale steam jet refrigerator. *International Journal of Refrigeration*, 18(6):378–386, 1995.

[15] I.W. Eames, S. Aphornratana, and D.-W. Sun. The jet-pump cycle — a low cost refrigerator option powered by waste heat. *Heat Recovery Systems and CHP*, 15(8):711–721, 1995.

[16] Hisham El-Dessouky, Hisham Ettouney, Imad Alatiqi, and Ghada Al-Nuwaibit. Evaluation of steam jet ejectors. *Chemical Engineering and Processing: Process Intensification*, 41(6):551–561, 2002.

[17] J. Fischer. On ejector technology. *International Journal of Refrigeration*, (In press).

[18] G. Flügel. *The design of jet pumps.* NACA Technical Memorandum 982, 1941.

[19] I.E. Frank and R. Todeschini. *The data analysis handbook.* Elsiever, 1994.

[20] G. Grazzini. *Ottimizzazione termodinamica di frigoriferi.* Pitagora Editrice Bologna, 1999.

[21] G. Grazzini and A. Mariani. A simple program to design a multi-stage jet-pump for refrigeration cycles. *Energy Conversion and Management*, 39(16–18):1827–1834, 1998.

[22] G. Grazzini, A. Milazzo, and D. Paganini. Design of an ejector cycle refrigeration system. *Energy Conversion and Management*, 54(1):38–46, 2012.

[23] G. Grazzini, A. Milazzo, and S. Piazzini. Prediction of condensation in steam ejector for a refrigeration system. *International Journal of Refrigeration*, 34(7):1641–1648, 2011.

[24] G. Grazzini, S. Piazzini, and A. Rocchetti. Confronto e verifica di correlazioni per il dimensionamento di uno scambiatore a piastre. In *XXV Congresso Nazionale UIT sulla Trasmissione del Calore*, Trieste, 2007.

[25] G. Grazzini, S. Piazzini, and A. Rocchetti. Progetto ottimizzato di un impianto frigorifero ad eiezione. In *XXV Congresso Nazionale UIT sulla Trasmissione del Calore*, Trieste, 2007.

[26] G. Grazzini and A. Rocchetti. Influence of the objective function on the optimisation of a steam ejector cycle. *International Journal of Refrigeration*, 31(3):510–515, 2008.

[27] E.A. Groll. Ejector technology. *International Journal of Refrigeration*, 34(7):1543–1544, 2011.

[28] S. He, Y. Li, and R.Z. Wang. Progress of mathematical modeling on ejectors. *Renewable and Sustainable Energy Reviews*, 13(8):1760–1780, 2009.

[29] A. Hemidi, F. Henry, S. Leclaire, J.-M. Seynhaeve, and Y. Bartosiewicz. Cfd analysis of a supersonic air ejector. part i: Experimental validation of single-phase and two-phase operation. *Applied Thermal Engineering*, 29(8–9):1523–1531, 2009.

[30] A. Hemidi, F. Henry, S. Leclaire, J.-M. Seynhaeve, and Y. Bartosiewicz. Cfd analysis of a supersonic air ejector. part ii: Relation between global operation and local flow features. *Applied Thermal Engineering*, 29(14–15):2990–2998, 2009.

[31] B. K. Hodge and K. Koenig. *Compressible fluid dynamics with personal computer applications*. Prentice Hall, 1995.

[32] B.J. Huang, J.M. Chang, C.P. Wang, and V.A. Petrenko. A 1-d analysis of ejector performance. *International Journal of Refrigeration*, 22(5):354–364, 1999.

[33] S. Kakac and H liu. *Heat exchangers : selection, rating, and thermal design*. CRC Press, 1998.

[34] W. M. Kays and A. L. London. *Compact heat exchangers*. McGraw-Hill, 1984.

[35] J.H. Keenan and E.P. Neumann. A simple air ejector. *ASME Journal of Applied Mechanics Translation*, 64:A75–84, 1942.

[36] J.H. Keenan, E.P. Neumann, and F. Lustwerk. An investigation of ejector design by analysis and experiment. *ASME Journal of Applied Mechanics Translation*, 72:299–309, 1950.

[37] S.A. Klein and A.H. Harvey. *NIST Standard Reference Database 10: NIST/ASME Steam Properties*. National Institute of Standards and Technology, Standard Reference Data Program, 2.0 edition, 1996.

[38] E.W. Lemmon, M.L. Huber, and M.O. McLinden. *NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties - REFPROP*. National Institute of Standards and Technology, Standard Reference Data Program, 8.0 edition, 2008.

[39] D.G. Luenberger. *Linear and nonlinear programming*. Addison-Wesley Publishing Company, 1984.

[40] J. T. Munday and D. F. Bagster. A new ejector theory applied to steam jet refrigeration. *Industrial and Engineering Chemistry Process Design and Development*, 16(4):442–449, 1977.

[41] E. Nehdi, L. Kairouani, and M. Elakhdar. A solar ejector air-conditioning system using environment-friendly working fluids. *International Journal of Energy Research*, 32(13):1194–1201, 2008.

[42] M. Ouzzane and Z. Aidoun. Model development and numerical procedure for detailed ejector analysis and design. *Applied Thermal Engineering*, 23(18):2337–2351, 2003.

[43] V. O. Petrenko. Application of innovative ejector chillers and air conditioners operating with low boiling refrigerants in trigeneration systems. *International Seminar on ejector/jet-pump technology and application, Louvain-la-Neuve, Belgium, September 7–9*, 2009.

[44] K. Pianthong, W. Seehanam, M. Behnia, T. Sriveerakul, and S. Aphornratana. Investigation and improvement of ejector refrigeration system using computational fluid dynamics technique. *Energy Conversion and Management*, 48(9):2556–2564, 2007.

[45] C. Pollerberg, A.H.H. Ali, and C. Dötsch. Experimental study on the performance of a solar driven steam jet ejector chiller. *Energy Conversion and Management*, 49(11):3318–3325, 2008.

[46] W. H. Press, B. P. Flanney, S. A. Teukolsky, and W. T. Vetterling. *Numerical recipes - The art of scientific computing.* Cambridge University Press, 1986.

[47] E.D. Rogdakis and G.K. Alexis. Investigation of ejector design at optimum operating condition. *Energy Conversion and Management*, 41(17):1841–1849, 2000.

[48] E. Rusly, L. Aye, W.W.S. Charters, and A. Ooi. Cfd analysis of ejector in a combined ejector cooling system. *International Journal of Refrigeration*, 28(7):1092–1101, 2005.

[49] A. Selvaraju and A. Mani. Analysis of an ejector with environment friendly refrigerants. *Applied Thermal Engineering*, 24(5–6):827–838, 2004.

[50] T. K. Serghides. Estimate friction factor accurately. *Chemical Engineer Journal*, 91:63–64, 1984.

[51] A.H. Shapiro. *The dynamics and thermodynamics of compressible fluid flow.* John Wiley and Sons, 1953.

[52] W. Spendley, G.R. Hext, and F.R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962.

[53] Da-Wen Sun. Comparative study of the performance of an ejector refrigeration cycle operating with various refrigerants. *Energy Conversion and Management*, 40(8):873–884, 1999.

[54] J. R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements.* University Science Books, 1997.

[55] S. Varga, A. C. Oliveira, and B. Diaconu. Numerical assessment of steam ejector efficiencies using cfd. *International Journal of Refrigeration*, 32(6):1203–1211, 2009.

[56] Z.-Z. Wang and Z.-N. Zhao. Analysis of performance of steam condensation heat transfer and pressure drop in plate condensers. *International Journal of Heat and Mass Transfer*, 14(4):32–41, 1993.

[57] Y.-Y. Yan and T.-F. Lin. Evaporation heat transfer and pressure drop of refrigerant r-134a in a plate heat exchanger. *Journal of Heat Transfer*, 121(1):118–127, 1999.

[58] Y.-Y. Yan, H.-C. Lio, and T.-F. Lin. Condensation heat transfer and pressure drop of refrigerant r-134a in a plate heat exchanger. *International Journal of Heat and Mass Transfer*, 42(6):993–1006, 1999.

[59] Y. Zhu, W. Cai, C. Wen, and Y. Li. Simplified ejector model for control and optimization. *Energy Conversion and Management*, 49(6):1424–1432, 2008.

[60] M. J. Zucrow and J. D. Hoffman. *Gas dynamics*. John Wiley and Sons, 1976.

Appendix A

Optimization program code

```
program PlantDesign

use dfport
use parameters
use cmplx_vars
use print_outs
use quit
use REFPROP8_mod

implicit none

logical :: makedirqq

! ======================================================================
c-> working directory
resi=getcwd(diri)
diri=trim(diri)//'\'

c-> Input
call readdat()

c-> Call SETUP to initialize REFPROP8 and set the pure fluid component name
c-> Define fluid name
icomp=1
hf(1)='R134a.fld'
hfmix='hmx.bnc'
hrf='DEF'
call SETUP (icomp,hf,hfmix,hrf,ierr,herr)
if (ierr.ne.0) write (*,*) herr

write(*,'(a,$)')'How_many_runs?_'
   66 continue
read(*,*)nr
if((nr.lt.1).or.(nr.gt.9))then
  write(*,'(a,$)')'Enter_a_number_between_1_and_9:_'
  goto 66
end if
```

```
c-> multi-RUN
do ir=1,nr
c-> gestione files
  write(ic,'(i1)')ir
  diro='OUTPUT_'//ic
  resl=makedirqq(trim(diro))
  diro=trim(diri)//trim(diro)//'\'

  file=trim(diro)//'skip.log'
  open(logu,file=trim(file),form='formatted')

  file=trim(diro)//'Randoms.dat'
  open(lrd,file=trim(file),form='formatted')

c-> Punti scartati
  file=trim(diro)//'Failure.dat'
  open(lko,file=trim(file),form='formatted')
  write(lko,2)'nt_g','mg_h','pg_c','dt_g','mp_1','mp_2','meis',
     -   'pe_n','np_c','mc_c','tc_n'
    2   format(11(a10,'___'))

c-> Punti buoni
  file=trim(diro)//'CplxPts.dat'
  open(lok,file=trim(file),form='formatted')
  write(lok,3)'nt_g','mg_h','pg_c','dt_g','mp_1','mp_2','meis',
     -   'pe_n','np_c','mc_c','tc_n','C.O.P.'
    3   format(12(a10,'___'))

c-> Generatore
  file=trim(diro)//'Generat.dat'
  open(lsg,file=trim(file),form='formatted')
  write(lsg,4)'IN:_mh','mc','ph','pc','Th_in','Tc_in','DT_sh',
     -   'Ntubi','D_int','Spess','OUT:_Tc_ou',
     -   '**_Q_**','**_D_p_**','**_W_pmp_**'
    4   format(14(a11,'__'))

c-> Evaporatore
  file=trim(diro)//'FlashEv.dat'
  open(lev,file=trim(file),form='formatted')
  write(lev,5)'IN:_Wf','m_lin','T_lin','p_lin','Loss','h_cond',
     -   'p_cond','p_eva','OUT:_m_eva',
     -   '**_m_s_v_**','**_m_ev_**','**_Sup_**','**_W_p_**'
    5   format(13(a11,'__'))

c-> Eiettore
  file=trim(diro)//'Ejector.dat'
  open(lej,file=trim(file),form='formatted')
  select case(ej_flag1)
  case(0)
c - modello standard
    write(lej,6)'IN:_p_gen_','T_gen_','m_g_1_','m_g_2_',
     -      'p_evap','m_evap','p_cond','T_cond',
     - '**_m_p1_**','**_A_g1_**','**_m_s1/m_p1_**','*_p_o1/p_s1',
     - '**_m_p2_**','**_A_g2_**','**_m_s2/m_p2_**','*_p_o2/p_s2',
     - '**_m_p__**','**_A_ex_**','**_m_ev/m_p__**','*_p_ex/p_ev'
    6   format(20(a11,'__'))
  case(1)
c- modello Eames
    write(lej,8)'IN:_p_gen_','T_gen_','m_gen_',
     -    'p_evap_','m_evap_','p_cond_','T_cond_',
     - '**__m_p_**_','**__A_g_**_','**__m_s_**_','*___p_ex__*',
     - '**__m_t_**_','**_A_ex_**_','*_m_s/m_p_*','*_p_ex/p_i_'
    8   format(15(a11,'__'))

  case default
    stop 'ERROR:_ej_flag1.'
```

```
  end select

c-> Condensatore
  file=trim(diro)//'Condens.dat'
  open(lsc,file=trim(file),form='formatted')
  write(lsc,7)'IN: fr_s ','T_s_in','T_nom ','fr_w ','T_w_in',
     -   'p_w_ou','n_pas ','OUT: p_s_in','h_s_ou','p_s_ou'
    7   format(10(a11,' '))

c-> Scambiatori a piastre (Frigel)
  file=trim(diro)//'PHEdata.dat'
  open(lpe,file=trim(file),form='formatted')

c-> risoluzione Complex
  call updt_bounds(1)
  call sub_complex(1)
c
c-> chiusura files
  close(logu)
  close(lrd)
  close(lko)
  close(lok)
  close(lsg)
  close(lev)
  close(lej)
  close(lsc)
        close(lpe)
end do
c
end


subroutine sub_complex(iflag)
c iflag=+1 -> maximum search
c iflag=-1 -> minimum search
use cmplx_vars
use parameters
c
implicit none
c
integer :: iflag,itgt,irmv,icp,it,j,itop,ibot,nrep,ite
integer :: ir,jolly
c
c-------------------------------------------------------------------------------
c
c-> setup
c random matrix
call random_seed()
call random_number(r)
c Complex StartUp (forzo un punto)
r(1,:)=0.5
c reset auxiliary complex points (just explicit variables)
x(2:ncp,1:nxe)=0.0
obj_fun(1:ncp)=0.0
c
c-> define auxiliary complex points (just explicit variables)
ir=0
icp=1
jolly=0
do while(icp.le.ncp)
  jolly=jolly+1
  if(ir.lt.ncp)then
    ir=ir+1
  else
    call random_number(r)
    ir=1
  end if
  do j=1,nxe
```

```
      x(icp,j)=bound_l(j)+r(ir,j)*(bound_u(j)-bound_l(j))
    end do
    call updt_objfun(icp)
    if(obj_fun(icp).gt.0.0)icp=icp+1
end do
c
c
c-> main complex loop
c   ==================
it=1
nrep=1
do while((it.lt.itmax).and.(nrep.le.mrep))
c find point with highest function value
   itop=locmax(obj_fun(:),ncp)
c find point with lowest function value
   ibot=locmin(obj_fun(:),ncp)
c first convergency criterion
   if(obj_fun(itop)-obj_fun(ibot).ge.epsln)nrep=0
   nrep=nrep+1
c
   if(nrep.le.mrep)then
c switch max/min search (according to iflag)
      itgt=((1+iflag)*itop+(1-iflag)*ibot)/2
      irmv=((1+iflag)*ibot+(1-iflag)*itop)/2
c replace the point with worst function value while it is worst
      call centroid(irmv,ncp)
      x(irmv,1:nxe)=(1.0+alpha)*xc(1:nxe)-alpha*x(irmv,1:nxe)
      call checkbounds(irmv,ncp)
      call updt_objfun(irmv)
      ite=1
      do while((locworst(iflag).eq.irmv).and.(ite.le.10))
x(irmv,1:nxe)=0.5*(x(irmv,1:nxe)+xc(1:nxe))
call checkbounds(irmv,ncp)
call updt_objfun(irmv)
ite=ite+1
      end do
      if(locworst(iflag).eq.irmv)then
x(irmv,1:nxe)=x(itgt,1:nxe)
obj_fun(irmv)=obj_fun(itgt)-0.5*epsln*float(iflag)
      end if
   end if
   it=it+1
end do
c
itgt=locworst(-iflag)
call updt_objfun(itgt)
call best_output(itgt)
c
write(*,*)
write(*,*)
write(*,*)'Done!'
c
return
end


subroutine centroid(iskip,ipmax)
 use cmplx_vars
c
implicit none
c
integer :: iskip,ipmax
integer :: i,j
real div
c
c-----------------------------------------------------------------------
c
div=1.0/float(ipmax-1)
```

```
do j=1,nxe
  xc(j)=0.0
  do i=1,ipmax
    xc(j)=xc(j)+x(i,j)
  end do
  xc(j)=(xc(j)-x(iskip,j))*div
end do
c
return
end


subroutine updt_objfun(icp)
c
use dflib
use cmplx_vars
use cycle_vars
use parameters
use pt_dataset
use print_outs
c
implicit none
c
integer :: icp
logical :: scarta
c
! ========================================================================
c
c- setup
obj_fun(icp)=0.0

mg_h=    x(icp, 1)
pg_c=    x(icp, 2)
dt_g=    x(icp, 3)
mp_1=    x(icp, 4)
meis=    x(icp, 5)
te_n=    x(icp, 6)
mc_c=    x(icp, 7)
tc_n=    x(icp, 8)
np_c=int(x(icp, 9))
np_e=int(x(icp,10))
np_g=int(x(icp,11))
c
c- printouts (per debug)
cread(lrd,1)nt_g,mg_h,pg_c,dt_g,mp_1,mp_2,meis,pe_n,np_c,mc_c,tc_n
write(lrd,1)nt_g,mg_h,pg_c,dt_g,mp_1,mp_2,meis,pe_n,np_c,mc_c,tc_n
    1 format(i9,'',7(e15.7),i9,'',2(e15.7))
c
c- calcolo componenti impianto completo
call componenti(scarta)
if(scarta)then
c
  write(lko,2)nt_g,mg_h,pg_c,dt_g,mp_1,mp_2,meis,pe_n,np_c,mc_c,
     -  tc_n
    2   format(i10,'___',e13.3,f13.1,f13.2,3e13.3,f13.1,i10,'___',
     -   e13.2,f13.2)
else
  obj_fun(icp)=q_frg/(q_gen+w_gen+w_eva+w_phe+w_pmp)
        if (obj_fun(icp).eq.0.0) then
              pause
        end if
c
  write(lok,3)nt_g,mg_h,pg_c,dt_g,mp_1,mp_2,meis,pe_n,np_c,mc_c,
     -  tc_n,obj_fun(icp)
    3   format(i10,'___',e13.3,f13.1,f13.2,3e13.3,f13.1,i10,'___',
     -   e13.2,f13.2,e13.3)
c  call good_output(icp)
c
```

```
  call best_output(icp)!output completo per il best-pt
end if
c
return
end


subroutine checkbounds(icp,icm)
use cmplx_vars
c
implicit none
c
integer :: icp,icm,j
logical :: repeat
c
! ============================================================================
c
repeat=.true.
do while(repeat)
  do j=1,nxe
    if(x(icp,j).lt.bound_l(j))then
x(icp,j)=bound_l(j)+delta
    else if(x(icp,j).gt.bound_u(j))then
x(icp,j)=bound_u(j)-delta
    end if
  end do
  repeat=.false.
c
  do j=nxe+1,nxt
    if((x(icp,j).lt.bound_l(j)).or.(x(icp,j).gt.bound_u(j)))then
call centroid(icp,icm)
x(icp,1:nxe)=0.5*(x(icp,1:nxe)+xc(1:nxe))
repeat=.true.
    end if
  end do
end do
c
return
end


     subroutine componenti(scarta)
! ================================================
c
       use cycle_vars
       use parameters
       use pt_dataset
       use Gdat
       use SCdat
       use quit
       use WSCall
         use REFPROP8Call
         use GEA_mod
c
       implicit none
c
       real,parameter :: toll=1.0e-3
       integer,parameter :: nit=50
c
       logical :: scarta
c
       real :: mg_c,hgic
       real :: tcou,dcou
         real :: teou
c
       real :: me,mex,em
       integer :: it
c
       real :: dum,dlth,dum2
```

```
      real :: tgam,pgam
c
c=====================================================
c
c-> startup
      scarta=.false.
      skip=.false.
      write(*,*)
      write(logu,*)'-------------------------------------------------'
c
      mg_c=mp_1+mp_2
      tgic=tc_n
c
      tcou=tc_n
      call REFRsatur_t(tcou,pcou,dcou,dum)
      hcou=REFRenth_td(tcou,dcou)
c
      pcin=pcou
c
      me=2.0e-3
      em=1.0
      it=1
        call REFRsatur_t(te_n,pe_n,dum,dum2)

c codice di lettura dei dati dalle tabelle GEA (l'equivalente codice
c di lettura delle variabili indipendenti posizionato in sub_complex)
c-> lettura dati generatore
if(GEAInputEna(3).eq.1) then
call GenReadGEAPHE
write(*,'(a,$)')'SG'//trim(GeaGen%PHEname)
w_gen=(GeaGen%WaterTotPdrop*1.0e+3)*GeaGen%WaterFlowRate/1.0e+3
pg_c=GeaGen%RefrPout*1.0e+5     !bar -> Pa
tgoc=GeaGen%RefrTout+273.15     !degC -> K
mg_c=GeaGen%RefrFlowRate
q_gen=GeaGen%PHEpower*1.0e+3    !kW -> W
if(skip)then
 scarta=.true.
 write(*,*)
 return
end if
write(*,'(a,$)').'
end if
c-> lettura dati evaporatore
if(GEAInputEna(1).eq.1) then
call EvaReadGEAPHE
write(*,'(a,$)')'EV'//trim(GeaEva%PHEname)
w_eva=(GeaEva%WaterTotPdrop*1.0e+3)*GeaEva%WaterFlowRate/1.0e+3
pe_n=GeaEva%RefrPout*1.0e+5     !bar -> Pa
me=GeaEva%RefrFlowRate
if(skip)then
 scarta=.true.
 write(*,*)
 return
end if
write(*,'(a,$)').'
end if
c-> lettura dati condensatore
if(GEAInputEna(3).eq.1) then
call ConReadGEAPHE
write(*,'(a,$)')'SC'//trim(GeaCon%PHEname)
w_phe=(GeaCon%WaterTotPdrop*1.0e+3)*GeaCon%WaterFlowRate/1.0e+3
pcou=GeaCon%RefrPout*1.0e+5     !bar -> Pa
tcou=GeaCon%RefrTout+273.15     !degC -> K
dcou=REFRdens_tp(tcou,pcou)
hcou=REFRenth_td(tcou,dcou)
if(GeaCon%RefrTotPdrop.gt.0.0) then
 pcin=(GeaCon%RefrPout*1.0e+5)+(GeaCon%RefrTotPdrop*1.0e+3)
```

```
      else
      _pcin=(GeaCon%RefrPout*1.0e+5)
      end_if
      tcin=GeaCon%RefrTin+273.15_____!degC_->_K
      if(skip)then
      _scarta=.true.
      _write(*,*)
      _return
      end_if
      write(*,'(a,$)')'.'
      end_if
      c->_calcolo_di_gamma_e_degli_altri_esponenti_isentropici
      _tgam=(tgoc+tcin)/2.0
      _pgam=(pg_c+pcin)/2.0
      _REFRgam=REFRgamma_tp(tgoc,pg_c)
      _REFRgm1_g=(REFRgam-1.0)/REFRgam
      _REFRg_gm1=REFRgam/(REFRgam-1.0)

      c====>_LOOP_sulla_portata_proveniente_dall'evaporatore (me)
            do while((em.gt.toll).and.(it.le.nit).and.(.not.skip))
              mex=me
              if(it.eq.1)mex=1.0

              if (GEAInputEna(3).eq.0) then
              write(*,'(a,$)')'SG'
      !         call generatore(mg_h,mg_c,p_g_h,pg_c,tghin,tgic,dt_g,nt_g,
      !     -                    dint,spes,tgoc)
                call GenPHE(mg_h,mg_c,p_g_h,pg_c,tghin,tgic,dt_g,np_g,!hp,smp,
           &                  tgoc)
              if(skip)then
                scarta=.true.
                write(*,*)
                return
              end if
              write(*,'(a,$)')'.'
              end if
      c
              if (GEAInputEna(1).eq.0) then
              write(*,'(a,$)')'EV'

      !         call evapf(q_frg,meis,tehin,p_e_h,dpeis,hcou,pcou,pe_n,me)
                call EvaPHE(meis,me,p_e_h,pe_n,tehin,te_n,hcou,0.0,np_e,!hp,smp,
           &                  teou)
              if(skip)then
                scarta=.true.
                write(*,*)
                return
              end if
              write(*,'(a,$)')'.'
              end if
      c
      c-> eiettore
              write(*,'(a,$)')'EJ'
              select case(ej_flag1)
              case(0)
                call ej2_design(pg_c,tgoc,mp_1,mp_2,pe_n,me,pcin,tcin)
              case(1)
                call eje_design_eames(pg_c,tgoc,mg_c,pe_n,me,pcin,tcin)
              case default
                stop 'ERROR:_ej_flag1'
              end select
              if(skip)then
                scarta=.true.
                write(*,*)
                return
              end if
              write(*,'(a,$)')'.'
```

```
c
c    ==============================
         if (GEAInputEna(3).eq.0) then
         mc_h=mg_c+me
         write(*,'(a,$)')'SC'
         call phe(mc_h,tcin,tc_n,mc_c,tcwi,pcwo,np_c,pcin,hcou,pcou)
         if(skip)then
           scarta=.true.
           write(*,*)
           return
         end if
         write(*,'(a,$)')'.'
         end if
c
         dcou=REFRdens_ph(pcou,hcou)
         dlth=(pg_c-pcou)/dcou
         hgic=hcou+dlth
         tgic=REFRtemp_ph(pg_c,hgic)
         w_pmp=mg_c*dlth
         em=abs(1.0-me/mex)
         it=it+1
       end do
c
       write(*,*)'_OK.'
c
       return
       end


subroutine Eje_design_Eames(pgen,tgen,mgen,peva,meva,pcon,tcon)
c
       use quit
       use TFData
       use REFPROP8Call
c
       implicit none
c
       integer,parameter :: iflg=0
       real,parameter :: ptol=0.5
c
       real :: pgen,tgen,mgen,peva,meva,pcon,tcon
c
       real :: px,rhux
       real :: d0p,d0s,dum
       real :: dp,sp,ep,pl,pr,pm
c
         integer :: test
c-> setup
       p0p=pgen
       t0p=tgen
       mp1=mgen
       mp2=0.0
       p0s=peva
       ms1=meva
c
       d0p=REFRdens_tp(t0p,p0p)
       h0_g=REFRenth_td(t0p,d0p)
       call REFRsatur_p(t0s,p0s,dum,d0s)
       h0_e=REFRenth_td(t0s,d0s)
c
c-> dimensionamento gola
       call chock(p0p,t0p,eta1,iflg,px,rhux)
       ag1=mp1/rhux
       ag2=0.0
c
c-> flusso secondario
c - condizione di chocking
       call chock(p0s,t0s,1.0,iflg,px,rhux)
```

```
c
      pl=px
      pr=p0s
      dp=0.5*(pr-pl)          !alias: -(pr-pl)/20  --
      p2=pr                                         !
      ep=-1.0                                       !
      do while((ep.lt.0.0).and.(dp.gt.0.01))        !
        dp=-0.1*dp           !<----------------------
        do while((ep.lt.0.0).and.(p2.gt.pl))
          pr=p2
          p2=p2+dp
          call solvEames()
          if(skip)return
          ep=p5-pcon
        end do
        dp=-0.1*dp
        do while((ep.gt.ptol).and.(p2.lt.pr))
          pl=p2
          p2=p2+dp
          call solvEames()
          if(skip)return
          ep=p5-pcon
        end do
      end do
c
c-> controllo convergenza
      if((ep.gt.ptol).or.(ep.lt.0.0))then
        skip=.true.
          write(logu,*)'EJ-Eje_design_eames:_controllo_finale.'
        return
      end if
c
      tcon=t5
     call CRMCdffsr()

      return
      end


      subroutine solvEames()
c
      use quit
      use TFData
c
      implicit none
c
      integer,parameter :: iflg=4
c
      real :: rhux
        integer :: test
c
c=====================================================
c
c-> dimensionamento sezione di uscita primario
      call xpans(p0p,t0p,p2,eta1,iflg,t21,d21,u21,m21,rhux,h21)
      if(skip)return
      a21=mp1/rhux
c-> dimensionamento sezione di uscita secondario
      call xpans(p0s,t0s,p2, 1.0,iflg,t22,d22,u22,m22,rhux,h22)
      if(skip)return
      a22=ms1/rhux
c-> diffusore-mixer
      call mixexpand(p2,mp1,h21,a21,u21,
     -                    ms1,h22,a22,u22,
     -                p5,mtt,h0m1,a5 ,u5 ,t5,d5,m5,test)
      if(skip)return
      return
      end
```

```
      subroutine chock(p0,t0,eta,iflg,p,rhu)
c
      use REFPROP8Call
c
      implicit none
c
      real,parameter :: tol=1.0e-6
      integer,parameter :: item=100
c
      integer :: iflg,ite
      real :: p0,t0,eta,rhu
      real :: tau,esp
      real :: ts,tt,d0,h0,s0
      real :: p,err,dl,ren
      real :: tis,dis,his
      real :: tre,dre,hre
      real :: vel,ssp,mch
c
c=============================================
c
        tau=0.5*(REFRgam+1.0)
        esp=REFRgam/(REFRgam-1.0)
        p  =p0*(1.0-1.0/eta*(1.0-1.0/tau))**esp
        rhu=sqrt(REFRgam*tau/(REFRrgas()*t0))*p
c
      return
      end


      subroutine xpans(p0,t0,p,eta,iflg,tre,dre,vel,mch,rhu)
c
      use REFPROP8Call
      use quit
c
      implicit none
c
      integer :: iflg
      real :: p0,t0,p,eta,tre,dre,vel,mch,rhu
      real :: ts,tt,d0,h0,s0
      real :: tis,dis,his,tri
      real :: hre,ssp
      real :: tau,esp
      real :: dl,ren
        real :: pstart,tstart,hstart,sstart,cstart,dstart
        real :: pnew,tnew,hnew,snew,cnew,dnew
        real :: runew,pratio
        integer :: count
        real :: enth
c
c=============================================
c
      if(p0/p.le.1.0)then
        skip=.true.
          write(logu,*)'EJ-xpans:_ingresso_alla_subroutine.'
        return
      end if

        tri=(p/p0)**((REFRgam-1.0)/REFRgam)          !Tis/T0
        tre=t0*(1.0+eta*(tri-1.0))
        mch=sqrt(2.0*(t0/tre-1.0)/(REFRgam-1.0))
        rhu=sqrt(REFRgam/(REFRrgas()*tre))*p*mch
        dre=p/(REFRrgas()*tre)
        vel=rhu/dre
      end select
c
      return
      end
```

```
      subroutine mixexpand(pi,m1,h01,sez1,u1,
     -                           m2,h02,sez2,u2,
     -                     po,m,h0,sezo,uo,to,do,mcho,test)
c
c
      use parameters
      use quit
      use TFData
      use REFPROP8Call
        use Gea_mod
c
      implicit none
c
      real,parameter :: tol=1.0e-6
      integer,parameter :: nt=35
c
      real :: pi,m1,h01,sez1,u1,m2,h02,sez2,u2,                    !input
     -        po,m,h0,sezo,uo,to,do,mchi,mcho                      !output
c
      real :: ui,ti,di
c aux
      real :: rut,ss,xi,xa
      real :: pt,tt,t0,px,tx,dx
      real :: ruo,rux,rum,dp,rou,mch
      real :: er,f0
      integer :: iflg,it
        integer :: test
        integer :: count
c
        real :: pstart,hstart,tstart,dstart,sstart,cstart,cend,pratio
        real :: deltac,hid
        real :: pnew,hnew,tnew,dnew,snew,runew,cnew,cnew2
c=======================================================
c
c-> setup
      m=m1+m2
      h0=(m1*h01+m2*h02)/m
      ui=(m1*u1+m2*u2)/m
c
      ui=etam*ui                          !cfr. Eames (1995)
c
        ti = REFRtemp_ph(pi,h0)
        di = REFRdensV_ph(pi,h0)
        rou = di*ui
        mch = ui/REFRsspd_td(ti,di)
      ai=m/rou
      mchi=mch
c
      sezo=ao

          hstart = h0
        pstart = pi
        tstart = ti
        dstart = di
        cstart = ui
c
        cend = 10.0
        deltac = (cstart - cend) / 50
        sstart = REFRentr_ph(pstart,hstart)
        cnew = cstart
        count = 0
        do while (count.lt.50)
                cnew = cnew - deltac
                hnew = hstart + (cstart**2.0-cnew**2.0)/2.0
                hid = hstart + etad * (hnew-hstart)
                pnew = REFRpres_hs(hid,sstart)
                tnew = REFRtemp_ph(pnew,hnew)
```

```
                snew = REFRentr_ph(pnew,hnew)
                dnew = REFRdens_ph(pnew,hnew)
                runew = dnew * cnew
                mch = cnew/REFRsspd_td(tnew,dnew)
                rum = max(rum,runew)
                count = count + 1
          end do
        rut=m/ao                              !target rho*u
        if(rut.gt.rum)then
          skip=.true.
          return
        end if
          golaDffsr = m/rum
          sezo = m/runew
          uo = cnew
          to = tnew
          do = dnew
          mcho = mch
          p5 = pnew
c
c-> assegnazioni per l'output
      p4=pi
      u4=ui
      t4=ti
      d4=di
      a4=ai
      m4=mchi
c     ptot=pt
c
      return
      end
```

Appendix B

Design program code

```
Attribute VB_Name = "Modulo3"
Sub Calc()

'jet pump refrigeration cycle

'Cycle points
'1 Hot water at generator inlet
'2 Hot water at generator pinch point
'3 Hot water at generator outlet
'4 Water at condenser inlet
'5 Water at condenser pinch point
'6 Water at condenser outlet
'7 Cold water at evaporator inlet
'8 Cold water at evaporator outlet
'9 R134a at generator inlet (= pump exit)
'10 R134a evaporation
'11 end of evaporation
'12 Superheated R134a at generator exit (=primary nozzle inlet)
'13 R134a at condenser inlet (=diffuser exit)
'14 start of R134a condensation
'15 end of condensation
'16 subcooled R134a at condenser exit (=pump inlet)
'17 liquid at SLHX exit (= start of lamination)
'18 R134a at evaporator inlet (= end of lamination)
'19 evaporation
'20 Superheated R134a at evaporator exit
'21 R134a vapour at secondary inlet
'22 Choked secondary flow
'23 Mixed fluid (diffuser inlet)
'24 End of primary expansion
'25 Primary nozzle throat

Dim T(25), P(25), s(25), h(25), rho(25), x(25), M(25), c(25), omega(25)
T0 = 273.15
cp_w = 4.186
pgre = 3.141592654

'data
With Worksheets("Data")
DTgen = .Cells(3, 2).Value        'minimum delta T at generator (K)
DTcond = .Cells(4, 2).Value       'minimum delta T at condenser (K)
DTevap = .Cells(5, 2).Value       'minimum delta T at evaporator (K)
```

```
DTsh = .Cells(6, 2).Value          'superheating at generator (K)
c0 = .Cells(7, 2).Value            'minimum fluid transfer velocity (m/s)
eta1 = .Cells(8, 2).Value          'primary expansion efficiency
eta2 = .Cells(9, 2).Value          'secondary expansion efficiency
etamix = .Cells(10, 2).Value       'mixing section efficiency
etap = .Cells(11, 2).Value         'feed pump efficiency
P_hw = .Cells(12, 2).Value         'thermal power at generator [kW]
fluido = .Cells(13, 2).Value       'fluid (according to NIST fluid names)
T(1) = .Cells(14, 2).Value + T0    'hot water at generator inlet [C]
T(4) = .Cells(15, 2).Value + T0    'water at condenser inlet [C]
T(7) = .Cells(16, 2).Value + T0    'cold water return [C]
T(8) = .Cells(17, 2).Value + T0    'cold water at deliver [C]
M(1) = .Cells(18, 2).Value         'water flow rate at generator [kg/s]
M(4) = .Cells(19, 2).Value         'water flow rate at condenser [kg/s]
DTsurr = .Cells(20, 2).Value       'superheating at evaporator exit [C]
DTsubc = .Cells(21, 2).Value       'subcooling at condenser exit [C]
DPcond = .Cells(22, 2).Value       'pressure loss at condenser [MPa]
DPev = .Cells(23, 2).Value         'pressure loss at evaporator [MPa]
DPgen = .Cells(24, 2).Value        'pressure loss at generator [MPa]
DPrigliq = .Cells(25, 2).Value     'pressure loss at SLHX (liquid side) [
    MPa]
DPrigvap = .Cells(26, 2).Value     'pressure loss at SLHX (vapour side) [
    MPa]
epsrig = .Cells(27, 2).Value       'SLHX efficiency
Dgen = .Cells(28, 2).Value         'internal diameter of the pipe from
    generator [mm]
Ds_int = .Cells(29, 2).Value       'internal diameter of the secondary
    anulus [mm]
Ds_est = .Cells(30, 2).Value       'external diameter of the secondary
    anulus [mm]
Dcond = .Cells(31, 2).Value        'diameter at condenser inlet [mm]
Length = .Cells(32, 2).Value       'length of the CRMC profile

End With

'calculation
omg = 0.5
Domg = 0.05 * omg
Flag = 0
omega(12) = pgre * (Dgen / 2) ^ 2
omega(21) = pgre * ((Ds_est - Ds_int) / 2) ^ 2
omega(13) = pgre * (Dcond / 2) ^ 2

'Fixed temperatures and pressures
T(16) = T(4) + DTcond
T(18) = T(8) - DTevap
P(18) = Pressure(fluido, "Tvap", "SI", T(18))
P(19) = P(18) - 0.8 * DPev
T(19) = Temperature(fluido, "Pvap", "SI", P(19))
h(19) = Enthalpy(fluido, "Pvap", "SI", P(19))
T(20) = T(19) + DTsurr
If T(20) > T(7) - DTevap Then T(20) = T(7) - DTevap
P(20) = P(19) - 0.2 * DPev
h(20) = Enthalpy(fluido, "pt", "SI", P(20), T(20))

'Initialization
T(10) = T(1) - 2 * DTgen
T(15) = T(4) + DTcond + DTsubc
P(15) = Pressure(fluido, "Tliq", "SI", T(15))
h(15) = Enthalpy(fluido, "Tliq", "SI", T(15))
P(14) = P(15) + DPcond * 0.75
T(14) = Temperature(fluido, "Pvap", "SI", P(14))
h(14) = Enthalpy(fluido, "Pvap", "SI", P(14))

10
P(10) = Pressure(fluido, "Tliq", "SI", T(10))
h(10) = Enthalpy(fluido, "Tliq", "SI", T(10))
```

```
P(11) = P(10) - DPgen / 3
T(11) = Temperature(fluido, "Pvap", "SI", P(11))
T(12) = T(11) + DTsh
If T(12) > T(1) - DTgen Then T(12) = T(1) - DTgen
P(12) = P(11) - DPgen / 3
h(12) = Enthalpy(fluido, "PT", "SI", P(12), T(12))

20
P(16) = P(15)
h(16) = Enthalpy(fluido, "pt", "SI", P(16), T(16))
Cliq = (1 + omg) * cp(fluido, "Ph", "SI", P(16), h(16))
Cvap = omg * cp(fluido, "Ph", "SI", P(20), h(20))
Cmin = Cvap
If Cliq < Cvap Then
    Cmin = Cliq
End If
Qrigmax = Cmin * (T(16) - T(20))
h(17) = h(16) - epsrig * Qrigmax / (1 + omg)
P(17) = P(16) - DPrigliq
T(17) = Temperature(fluido, "ph", "SI", P(17), h(17))
s(17) = Entropy(fluido, "ph", "SI", P(17), h(17))
P(9) = P(10) + DPgen / 3
hid = Enthalpy(fluido, "ps", "SI", P(9), s(17))
h(9) = h(17) + (hid - h(17)) / etap
T(9) = Temperature(fluido, "ph", "SI", P(9), h(9))
s(9) = Entropy(fluido, "ph", "SI", P(9), h(9))
mp = P_hw / (h(12) - h(9))
ms = mp * omg
T(5) = T(4) + (ms + mp) * (h(14) - h(16)) / cp_w / M(4)
T14new = T(5) + DTcond
If Abs(T(14) - T14new) > 0.0001 Then
    T(14) = T14new
    P(14) = Pressure(fluido, "Tvap", "SI", T(14))
    h(14) = Enthalpy(fluido, "Tvap", "SI", T(14))
    P(15) = P(14) - DPcond * 0.75
    T(15) = Temperature(fluido, "Pliq", "SI", P(15))
    GoTo 20
End If
T(2) = T(1) - mp * (h(12) - h(10)) / cp_w / M(1)
T10new = T(2) - DTgen
If Abs(T10new - T(10)) > 0.001 Then
    T(10) = T10new
    GoTo 10
End If

h(18) = h(17)
s(18) = Entropy(fluido, "ph", "SI", P(18), h(18))
h(21) = h(20) + epsrig * Qrigmax / omg
P(21) = P(20) - DPrigvap
T(21) = Temperature(fluido, "ph", "SI", P(21), h(21))
s(21) = Entropy(fluido, "ph", "SI", P(21), h(21))
rho(21) = Density(fluido, "ph", "SI", P(21), h(21))
c(21) = ms / rho(21) / (omega(21) / 1000000)
P(22) = Prcrit(fluido, P(21), h(21), c(21), eta2)
hid = Enthalpy(fluido, "ps", "SI", P(22), s(21))
h(22) = h(21) - (h(21) - hid) * eta2
T(22) = Temperature(fluido, "ph", "SI", P(22), h(22))
s(22) = Entropy(fluido, "ph", "SI", P(22), h(22))
rho(22) = Density(fluido, "ph", "SI", P(22), h(22))
s(12) = Entropy(fluido, "ph", "SI", P(12), h(12))
rho(12) = Density(fluido, "ph", "SI", P(12), h(12))
c(12) = mp / rho(12) / (omega(12) / 1000000)
P(24) = P(22)
hid = Enthalpy(fluido, "Ps", "SI", P(24), s(12))
dh = (h(12) - hid) * eta1
h(24) = h(12) - dh
c(24) = (c(12) ^ 2 + 2000 * (h(12) - h(24))) ^ 0.5
```

```
c(22) = (c(21) ^ 2 + 2000 * (h(21) - h(22))) ^ 0.5
htp = h(24) + c(24) ^ 2 / 2000
hts = h(22) + c(22) ^ 2 / 2000
P(23) = P(24)
c(23) = etamix ^ 0.5 * (c(24) + omg * c(22)) / (1 + omg)
h(23) = (htp + omg * hts) / (1 + omg) - c(23) ^ 2 / 2000
T(23) = Temperature(fluido, "ph", "SI", P(23), h(23))
rho(23) = Density(fluido, "ph", "SI", P(23), h(23))
omega(23) = (mp + ms) / rho(23) / c(23)
rin = (omega(23) / pgre) ^ 0.5 * 1000

rho(24) = Density(fluido, "Ph", "SI", P(24), h(24))
omega(22) = ms / rho(22) / c(22)
omega(24) = mp / rho(24) / c(24)
r1p = (omega(24) / WorksheetFunction.Pi) ^ 0.5
r1s = ((omega(22) + omega(24)) / WorksheetFunction.Pi) ^ 0.5
h1s = r1s - r1p
'etamix = 0.95
rough = 0.00005
Call Mixing(mp, rho(24), ms, rho(22), r1p, h1s, c(24), c(22), omg, rin,
      etamix, fluido, P(23), rough)

P(13) = P(14) + DPcond * 0.25
Angle = 5
Pnew = CrmcDffsr(fluido, P(23), c(23), h(23), rin, rough, 5, 2, Length,
      5, 100, Dcond, h(13))
DeltaP = Pnew - P(13)

If Abs(DeltaP) < 0.0001 Or Domg < 0.000001 Then GoTo 30
If DeltaP > 0 Then
    If Flag < 0 Then Domg = 0.75 * Domg
    omg = omg + Domg
    Flag = 1
Else
    If Flag > 0 Then Domg = 0.75 * Domg
    omg = omg - Domg
    Flag = -1
End If
GoTo 10

30
'other properties
s(10) = Entropy(fluido, "Tliq", "SI", T(10))
s(11) = Entropy(fluido, "Pvap", "SI", P(11))
h(11) = Enthalpy(fluido, "Pvap", "SI", P(11))
T(13) = Temperature(fluido, "ph", "SI", P(13), h(13))
s(13) = Entropy(fluido, "ph", "SI", P(13), h(13))
rho(13) = Density(fluido, "ph", "SI", P(13), h(13))
c(13) = (mp + ms) / rho(13) / (omega(13) / 1000000)
s(14) = Entropy(fluido, "Tvap", "SI", T(14))
s(15) = Entropy(fluido, "Tliq", "SI", T(15))
s(16) = Entropy(fluido, "pt", "SI", P(16), T(16))
rho(16) = Density(fluido, "pt", "SI", P(16), T(16))
s(19) = Entropy(fluido, "Pvap", "SI", P(19))
s(20) = Entropy(fluido, "pt", "SI", P(20), T(20))
s(23) = Entropy(fluido, "ph", "SI", P(23), h(23))
T(24) = Temperature(fluido, "Ph", "SI", P(24), h(24))
s(24) = Entropy(fluido, "Ph", "SI", P(24), h(24))
rho(24) = Density(fluido, "Ph", "SI", P(24), h(24))
P(25) = Prcrit(fluido, P(12), h(12), c(12), eta1)
hid = Enthalpy(fluido, "ps", "SI", P(25), s(12))
h(25) = h(12) - (h(12) - hid) * eta1
T(25) = Temperature(fluido, "ph", "SI", P(25), h(25))
s(25) = Entropy(fluido, "ph", "SI", P(25), h(25))
rho(25) = Density(fluido, "ph", "SI", P(25), h(25))
c(25) = (c(12) ^ 2 + 2000 * (h(12) - h(25))) ^ 0.5
```

```
'water circuits
T(2) = T(10) + DTgen
mhw = mp * (h(12) - h(10)) / cp_w / (T(1) - T(2))
T(3) = T(2) - mp * (h(10) - h(9)) / cp_w / mhw
T(5) = T(14) - DTcond
mw_c = (mp + ms) * (h(14) - h(16)) / cp_w / (T(5) - T(4))
T(6) = T(5) + (mp + ms) * (h(13) - h(14)) / cp_w / mw_c
mcw = ms * (h(20) - h(18)) / cp_w / (T(7) - T(8))
s(1) = s(12)
s(2) = s(10)
s(3) = s(9)
s(4) = s(16)
s(5) = s(14)
s(6) = s(13)
s(7) = s(20)
s(8) = s(18)

'global results
P_hw = mp * (h(12) - h(9))
P_cw = ms * (h(20) - h(18))
P_p = (ms + mp) * (h(9) - h(17))
P_cond = (ms + mp) * (h(13) - h(16))
'Pp_gen = mhw * 100 / 1000
'Pp_ev = mcw * 7300 / 1000
'Pp_cond = mw_c * 49000 / 1000
COP = P_cw / (P_hw + P_p)
COP_c = (1 - T(4) / T(1)) / ((T(4) - T(7)) / T(7))
'COP1 = P_cw / (P_hw + P_p + Pp_gen + Pp_ev + Pp_cond)
etaII = COP / COP_c
omega(23) = omega(23) * 1000000
omega(22) = ms / rho(22) / c(22) * 1000000
omega(24) = mp / rho(24) / c(24) * 1000000
omega(25) = mp / rho(25) / c(25) * 1000000

With Worksheets("Results")
For i = 1 To 25
.Cells(i, 1).Value = T(i)
.Cells(i, 2).Value = s(i)
.Cells(i, 3).Value = P(i)
.Cells(i, 4).Value = h(i)
.Cells(i, 5).Value = omega(i)
Next

'global results
.Cells(1, 12).Value = omg
.Cells(2, 12).Value = COP
.Cells(3, 12).Value = P_cw
.Cells(4, 12).Value = mp
.Cells(5, 12).Value = ms
.Cells(6, 12).Value = mhw
.Cells(7, 12).Value = mcw
.Cells(8, 12).Value = mw_c
.Cells(9, 12).Value = rho(16)
.Cells(10, 12).Value = mp / rho(16) * 3600
.Cells(11, 12).Value = P(9) - P(16)
.Cells(12, 12).Value = P_p
.Cells(13, 12).Value = (P(9) - P(16)) * 1000000 / rho(16) / 9.81
.Cells(14, 12).Value = P_cond
.Cells(15, 12).Value = etaII
.Cells(2, 16).Value = COP1

'primary expansion
n = 30
Tp = T(12)
sp = s(12)
pp = P(12)
hp = h(12)
```

```
rhop = Density(fluido, "ph", "SI", pp, hp)
cc = c(12)
pr = (P(12) / P(24)) ^ 0.1
.Cells(n, 1).Value = Tp
.Cells(n, 2).Value = sp
.Cells(n, 3).Value = pp
.Cells(n, 4).Value = hp
.Cells(n, 5).Value = rhop
.Cells(n, 6).Value = cc
For i = 1 To 10
pp = pp / pr
hp = h(12) - (h(12) - Enthalpy(fluido, "ps", "SI", pp, s(12))) * eta1
Tp = Temperature(fluido, "Ph", "SI", pp, hp)
sp = Entropy(fluido, "Ph", "SI", pp, hp)
rhop = Density(fluido, "ph", "SI", pp, hp)
cc = (c(12) ^ 2 + 2000 * (h(12) - hp)) ^ 0.5
.Cells(n + i, 1).Value = Tp
.Cells(n + i, 2).Value = sp
.Cells(n + i, 3).Value = pp
.Cells(n + i, 4).Value = hp
.Cells(n + i, 5).Value = rhop
.Cells(n + i, 6).Value = cc
Next

'secondary expansion
n = 45
Tp = T(21)
sp = s(21)
pp = P(21)
hp = h(21)
rhop = rho(21)
cc = c(21)
pr = (P(21) / P(22)) ^ 0.1
.Cells(n, 1).Value = Tp
.Cells(n, 2).Value = sp
.Cells(n, 3).Value = pp
.Cells(n, 4).Value = hp
.Cells(n, 5).Value = rhop
.Cells(n, 6).Value = cc
For i = 1 To 10
pp = pp / pr
hp = h(21) - (h(21) - Enthalpy(fluido, "ps", "SI", pp, s(21))) * eta2
Tp = Temperature(fluido, "Ph", "SI", pp, hp)
sp = Entropy(fluido, "Ph", "SI", pp, hp)
rhop = Density(fluido, "ph", "SI", pp, hp)
cc = (c(21) ^ 2 + 2000 * (h(21) - hp)) ^ 0.5
.Cells(n + i, 1).Value = Tp
.Cells(n + i, 2).Value = sp
.Cells(n + i, 3).Value = pp
.Cells(n + i, 4).Value = hp
.Cells(n + i, 5).Value = rhop
.Cells(n + i, 6).Value = cc
Next

'pump
n = 60
Tp = T(17)
sp = s(17)
pp = P(17)
hp = h(17)
pr = (P(9) / P(17)) ^ 0.25
.Cells(n, 1).Value = Tp
.Cells(n, 2).Value = sp
.Cells(n, 3).Value = pp
.Cells(n, 4).Value = hp
For i = 1 To 4
pp = pp * pr
```

```
hp = h(17) + (Enthalpy(fluido, "ps", "SI", pp, s(17)) - h(17)) / etap
Tp = Temperature(fluido, "Ph", "SI", pp, hp)
sp = Entropy(fluido, "Ph", "SI", pp, hp)
.Cells(n + i, 1).Value = Tp
.Cells(n + i, 2).Value = sp
.Cells(n + i, 3).Value = pp
.Cells(n + i, 4).Value = hp
Next

'liquid heating
n = 70
Tp = T(9)
sp = s(9)
diT = (T(10) - T(9)) / 10
.Cells(n, 1).Value = Tp
.Cells(n, 2).Value = sp
For i = 1 To 9
Tp = Tp + diT
sp = Entropy(fluido, "PT", "SI", P(9), Tp)
.Cells(n + i, 1).Value = Tp
.Cells(n + i, 2).Value = sp
Next
Tp = T(10)
sp = s(10)
.Cells(n + 10, 1).Value = Tp
.Cells(n + 10, 2).Value = sp

'superheating
n = 85
Tp = T(11)
sp = s(11)
.Cells(n, 1).Value = Tp
.Cells(n, 2).Value = sp
diT = (T(12) - T(11)) / 10
For i = 1 To 10
Tp = Tp + diT
sp = Entropy(fluido, "PT", "SI", P(11), Tp)
.Cells(n + i, 1).Value = Tp
.Cells(n + i, 2).Value = sp
Next

End With
End Sub


Attribute VB_Name = "Modulo5"
Function Mixing(mp, rhop, ms, rhos, r1p, h1s, c1p, c1s, omg, r2, eta,
    fluid, pmix, rough)
'*******************************
'Author:     Dario Paganini (DP)
'Purpose:    Define mixing chamber profile
'Version:    1.4 (2011/09/05) - (DP) removed eta assignment into the
    function
'                code
'            1.3 (2011/02/17) - (DP) added an iterative cycle to compute
     angle
'                of the inlet profile
'            1.2 (2011/02/07) - (DP) modified output worksheet reference
'            1.1 (2011/01/05) - (DP) added input parameters including
    mixing
'                effciency
'                converted from subroutine to function
'            1.0 (2010/12/03) - (DP) firts version
'*******************************
'*******************************
'INPUT
'mp      inlet primary flowrate [kg/s]
'rhop    inlet primary density [kg/m^3]
```

```
'ms       inlet secondary flowrate [kg/s]
'rhos     inlet secondary density [kg/m^3]
'r1p      primary inlet radius [m]
'h1s      secondary inlet enthalpy [KJ/kg]
'c1p      inlet primary speed [m/s]
'c1s      inlet secondary speed [m/s]
'omg      entrainment ratio
'r2       outlet radius [m]
'eta      mixing efficiency
'fluid    fluid code
'pmix     mixing pressure [MPa]
'rough    realtive roughness
'
'OUTPUT
'outputs are printed on the "Mixing" worksheet.
'*******************************

'Write input variables
With Worksheets("Mixing")
    .Cells(3, 2).Value = mp
    .Cells(4, 2).Value = ms
    thetap = .Cells(5, 2).Value
    .Cells(6, 2).Value = r1p
    .Cells(7, 2).Value = h1s
    .Cells(8, 2).Value = c1p
    .Cells(9, 2).Value = c1s
    .Cells(10, 2).Value = omg
    c2 = eta ^ 0.5 * (c1p + omg * c1s) / (1 + omg)
    .Cells(11, 2).Value = c2
    .Cells(12, 2).Value = (r2 / 1000)
    lmix = ((r2 / 1000) - r1p) / Tan(thetap * WorksheetFunction.Pi /
        180)
    .Cells(13, 2).Value = lmix
    .Cells(3, 6).Value = rhop
    .Cells(4, 6).Value = rhos
End With
thetap = thetap * WorksheetFunction.Pi / 180
'Define mixing section variables
Dim x(), rp(), mx(), mjx(), msx(), omegajx(), cjx(), rhojx(), omegasx()
    , omegax(), rx(), xs() As Double

'Define data array
'eta = 0.95
n = 10
ReDim x(n)
ReDim rp(n)
ReDim mx(n)
ReDim mjx(n)
ReDim msx(n)
ReDim omegajx(n)
ReDim cjx(n)
ReDim rhojx(n)
ReDim omegasx(n)
ReDim omegax(n)
ReDim rx(n)
ReDim xs(n)

'Assign intitial values
10
x(0) = 0
rp(0) = r1p
mx(0) = 0
mjx(0) = mp
msx(0) = ms
omegajx(0) = WorksheetFunction.Pi * rp(0) ^ 2
cjx(0) = c1p
rhojx(0) = mjx(0) / (omegajx(0) * cjx(0))
```

```
omegasx(0) = msx(0) / (rhos * c1s)
omegax(0) = omegajx(0) + omegasx(0)
rx(0) = (rp(0) ^ 2 + omegasx(0) / WorksheetFunction.Pi * Cos(alpha)) ^
    0.5
xs(0) = x(0) + (rx(0) - rp(0)) * Tan(alpha)

'Assign values
For i = 1 To n
    x(i) = x(i - 1) + (lmix / n)
    rp(i) = r1p + (x(i) * Tan(thetap))
    mx(i) = 2 * ms * x(i) * Tan(thetap) * (r1p + (x(i) * Tan(thetap) /
        2)) / ((r2 / 1000) ^ 2 - r1p ^ 2)
    mjx(i) = mp + mx(i)
    msx(i) = ms - mx(i)
    omegajx(i) = WorksheetFunction.Pi * rp(i) ^ 2
    cjx(i) = (c1s + (mp * (c1p - c1s) / (mjx(i)))) * eta ^ 0.5
    rhojx(i) = mjx(i) / (omegajx(i) * cjx(i))
    omegasx(i) = msx(i) / (rhos * c1s)
    omegax(i) = omegajx(i) + omegasx(i)
    rx(i) = (rp(i) ^ 2 + omegasx(i) / WorksheetFunction.Pi * Cos(alpha)
        ) ^ 0.5
    xs(i) = x(i) + (rx(i) - rp(i)) * Tan(alpha)
Next

alphanew = -Atn((rx(1) - rx(0)) / (x(1) - x(0)))
If Abs(alphanew - alpha) > 0.0001 Then
    alpha = alphanew
    GoTo 10
End If

'Conical section (frcitionless) added to the end of mixing chamber
'to activate the change reading from "Mixing" worksheet in CrmcDffsr
    function
'must be shifted one row down (from indexes 25-26 to indexes 26-27)
x1 = x(n)
x2 = x1 + (x(n) - x(n - 1))
dx = x2 - x1
rr1 = rx(n)
rr2 = rr1 + (rx(n) - rx(n - 1))
omegax1 = omegax(n)
omegax2 = WorksheetFunction.Pi * rr2 ^ 2
domegax = omegax2 - omegax1
v1 = cjx(n)
mach = v1 / SpeedOfSound(fluid, "dp", "SI", rhojx(n), pmix)
cpConic = cp(fluid, "dp", "SI", rhojx(n), pmix)
cvConic = cv(fluid, "dp", "SI", rhojx(n), pmix)
gammaConic = cpConic / cvConic
muConic = Viscosity(fluid, "dp", "SI", rhojx(n), pmix)
ReConic = 10 ^ 6 * rhojx(n) * v1 * 2 * rr1 / muConic
RelRough = rough
fSergConic = fSerg(RelRough, ReConic)
Dv = domegax * v1 / (omegax1 * (mach ^ 2 - 1)) + gammaConic * (mach ^ 2
    / (1 - mach ^ 2)) * v1 * fSergConic * dx / rr1
v2 = v1 + Dv

'Print results
With Worksheets("Mixing")
    .Range("A16:A26").ClearContents
    .Range("B16:B26").ClearContents
    .Range("C16:C26").ClearContents
    .Range("D16:D26").ClearContents
    .Range("E16:E26").ClearContents
    .Range("F16:F26").ClearContents
    .Range("G16:G26").ClearContents
    .Range("H16:H26").ClearContents
    .Range("I16:I26").ClearContents
    .Range("J16:J26").ClearContents
```

```
    .Range("K16:K26").ClearContents
    .Range("L16:L26").ClearContents
    For i = 0 To n
        .Cells(16 + i, 1).Value = x(i)
        .Cells(16 + i, 2).Value = rp(i)
        .Cells(16 + i, 3).Value = mx(i)
        .Cells(16 + i, 4).Value = mjx(i)
        .Cells(16 + i, 5).Value = msx(i)
        .Cells(16 + i, 6).Value = omegajx(i)
        .Cells(16 + i, 7).Value = cjx(i)
        .Cells(16 + i, 8).Value = rhojx(i)
        .Cells(16 + i, 9).Value = omegasx(i)
        .Cells(16 + i, 10).Value = omegax(i)
        .Cells(16 + i, 11).Value = rx(i)
        .Cells(16 + i, 12).Value = xs(i)
    Next
    .Cells(16, 13).Value = alpha
    .Cells(16 + i, 1).Value = x2
    .Cells(16 + i, 4).Value = mjx(n)
    .Cells(16 + i, 7).Value = v2
    .Cells(16 + i, 8).Value = (mjx(n) / v2 / omegax2)
    .Cells(16 + i, 10).Value = omegax2
    .Cells(16 + i, 11).Value = rr2
    .Cells(16 + i, 12).Value = x2
End With

End Function


Attribute VB_Name = "Modulo1"
'**********************************
'***          Module header         ***
'**********************************
'Authors:    Dario Paganini (DP)
'Contents:   Diffuser design function and related functions
'Suboutines and functions included:
'-----------------------------------
'Name              Version
'-----------------------------------
'CrmcDffsr         2.2 (2011/11/14)
'fdAdx             1.0 (2011/06/23)
'fdVdx             1.0 (2011/07/04)
'fM                1.0 (2011/07/04)
'Vcrmc             1.1 (2011/07/05)
'dVdxcrmc          1.0 (2011/07/04)
'Acon              1.0 (2011/07/05)
'dAdxConic         1.0 (2011/07/05)
'frho              1.0 (2011/07/04)
'fh                1.0 (2011/07/04)
'fRe               1.0 (2011/07/04)
'RK_NR             2.2 (2011/07/08)
'RK4_NR            1.0 (2011/04/21)
'derivs            2.1 (2011/11/14)
'**********************************
'**********************************

'**********************************
'Define public variables for the module
'**********************************
Dim Pi
Dim fluid, mDffsr, RelRough, _
    v_inlet, h_inlet, _
    dvdx_mix, dvdx_crmc, _
    Lcrmc, Lconic, Lratio, AngleConic, OutSectDiam, _
    derivs_flag
'**********************************

Function CrmcDffsr(fld, pInput, cInput, hInput, rInput, RelRough,
```

```
    cOutCrmc, _
                    StepCrmc, Length, MaxAngle, nCellConic, dOut, hOut)
'**********************************
'Author:     Dario Paganini (DP)
'Purpose:    Compute supersonic diffuser profile using CRMC method
'Version:    1.0 (2010/09/23) - (DP) firts version
'Version:    1.1 (2010/12/30) - (DP) bug removed for EndCRMCIndex
    assignment
'Version:    1.2 (2011/02/07) - (DP) added code to joint mixing section
    and
'                diffuser with continuous derivative
'Version:    1.3 (2011/02/17) - (DP) code to joint mixing section and
    diffuser
'                modified: read one row shifted data
'Version:    1.4 (2011/06/22) - (DP) assign velocity change parameters
    instead
'                read them from worksheet ("Mixing" worksheet not
    present)
'Version:    1.5 (2011/07/05) - (DP) integration method replaced by RK
    method
'                from Numerical Recipes - temporary test version
'Version:    2.0 (2011/07/07) - (DP) removed previous integration
    methods and
'                replaced by Runge-Kutta integration from Numerical
    Recipes
'Version:    2.1 (2011/07/07) - (DP) change input variable fluid in fld
    (to
'                avoid scope conflict)
'Version:    2.2 (2011/11/14) - (DP) corrected friction factor f
    assignment
'                (Fanning friction factor expected, Darcy passed)
'**********************************
'**********************************
'INPUT
'fld          NIST database code
'pInput       Pressure at the inlet section [MPa]
'cInput       Speed at the inlet section [m/s]
'hInput       Enthalpy at the inlet section [kJ/kg]
'rInput       Inlet radius [mm]
'RelRough     Relative roughness
'cOutCrmc     Output speed of the CRMC length [m/s]
'StepCrmc     Control volume length along the CRMC length [mm]
'LCrmc        Length of the CRMC part [mm]
'MaxAngle     Maximum inclination of the exit profile [degree]
'             Before MaxAngle: CRMC profile; After MaxAngle: conic
    profile
'nCellConic   Number of cells of the conic length
'dOut         Exit diameter of the diffuser [mm]

'OUTPUT
'CrmcDffsr    Pressure at the outlet section [MPa]
'hOut         Enthalpy at the outlet section [kJ/kg]
'**********************************

'Assign constant
Pi = WorksheetFunction.Pi

'**********************************
'Inputs unit conversion and steps number assignment
'**********************************
'Input unit conversion
'Step length [mm]
dx = StepCrmc
'Diffuser length [m]
Lcrmc = Length / 1000
'Number of cells
nCells = 1000 * Lcrmc / dx
```

```
'Double the cells for midpoint step analisys
nCellsConic = 2 * nCellConic
'Output section diameter [m]
OutSectDiam = dOut / 1000
fluid = fld

'*********************************
'Assign speeds along the CRMC length
'*********************************
'use first nMixDff step to joint with mixing region imposing continous
    derivative
Lratio = 2 / 3
nMixDff = Int(nCells * Lratio)
cOutOffset = 0 ' -20
'Starting velocity derivative (read from "Mixing")
With Worksheets("Mixing")
    x1 = .Cells(26, 1).Value
    x2 = .Cells(27, 1).Value
    v1 = .Cells(26, 7).Value
    v2 = .Cells(27, 7).Value
End With
'x1 = 0.047334863
'x2 = 0.052068349
'v1 = 280.0152429
'v2 = 277.3151838

dvdx_mix = (v2 - v1) / (x2 - x1)
dvdx_crmc = (cOutCrmc - cInput + cOutOffset) / Lcrmc

'*********************************
CrmcSection:
'*********************************

    'Define variables for crmc stretch
    '*******************************
    Dim xCrmc(), rCrmc(), aCrmc(), tanCrmc(), _
        pCrmc(), tCrmc(), rhoCrmc(), sCrmc(), hCrmc(), _
        machCrmc(), cCrmc(), ReCrmc(), gammaCrmc(), muCrmc(), fCrmc(),
            _
        dVdx() As Double
    '*******************************
    ReDim xCrmc(nCells), rCrmc(nCells), aCrmc(nCells), tanCrmc(nCells),
        _
        pCrmc(nCells), tCrmc(nCells), rhoCrmc(nCells), sCrmc(nCells),
            _
        hCrmc(nCells), machCrmc(nCells), cCrmc(nCells), ReCrmc(nCells
            ), _
        gammaCrmc(nCells), muCrmc(nCells), fCrmc(nCells), dVdx(nCells
            )
    '*******************************

    'Assign initial values
    aInput = Pi * (rInput / 1000) ^ 2
    v_inlet = cInput
    h_inlet = hInput
    'Assign mass flow rate
    rhoInput = Density(fluid, "ph", "SI", pInput, hInput)
    mDffsr = rhoInput * cInput * aInput

    'RK integration
    derivs_flag = "Crmc"
    Call RK_NR(aInput, 0, Lcrmc, dx / 1000, xCrmc, aCrmc)
    'Call RK_NR(aCrmc(0), xCrmc(0), xCrmc(nCells), dx / 1000, xCrmcTest
        , aCrmcTest)
    derivs_flag = ""

    'Thermodynamic properties computing
```

```
    For i = 0 To nCells - 1
        cCrmc(i) = Vcrmc(xCrmc(i))
        'Compute thermodynamic properties
        rhoCrmc(i) = frho(cCrmc(i), aCrmc(i), mDffsr)
        hCrmc(i) = fh(cCrmc(i), h_inlet, v_inlet)
        pCrmc(i) = Pressure(fluid, "dh", "SI", rhoCrmc(i), hCrmc(i))
        tCrmc(i) = Temperature(fluid, "dh", "SI", rhoCrmc(i), hCrmc(i))
        sCrmc(i) = Entropy(fluid, "ph", "SI", pCrmc(i), hCrmc(i))
        'Compute driving potentials variables
        machCrmc(i) = fM(cCrmc(i), SpeedOfSound(fluid, "ph", "SI",
            pCrmc(i), hCrmc(i)))
        rCrmc(i) = (aCrmc(i) / Pi) ^ 0.5
        dVdx(i) = dVdxcrmc(xCrmc(i))
        gammaCrmc(i) = cp(fluid, "ph", "SI", pCrmc(i), hCrmc(i)) / cv(
            fluid, "ph", "SI", pCrmc(i), hCrmc(i))
        'Compute Fanning friction factor
        muCrmc(i) = Viscosity(fluid, "dh", "SI", rhoCrmc(i), hCrmc(i))
        ReCrmc(i) = fRe(cCrmc(i), rhoCrmc(i), rCrmc(i), muCrmc(i))
        fCrmc(i) = fSerg(RelRough, ReCrmc(i)) / 4
        rCrmc(i + 1) = (aCrmc(i + 1) / Pi) ^ 0.5
        tanCrmc(i) = Atn((rCrmc(i + 1) - rCrmc(i)) / (xCrmc(i + 1) -
            xCrmc(i))) * 180 / Pi
        If tanCrmc(i) > MaxAngle Then
            StartCRMCIndex = 0
            EndCRMCIndex = i
            GoTo ConicalSection
        End If
    Next
    'Assign index if maximum angle value not reached
    StartCRMCIndex = 0
    EndCRMCIndex = i - 1

'*********************************
ConicalSection:
'*********************************
    'Define variables for conic stretch
    '*****************************
    Dim xConic(), rConic(), aConic(), tanConic(), _
        pConic(), tConic(), rhoConic(), sConic(), hConic(), _
        machConic(), cConic(), ReConic(), gammaConic(), muConic(),
            fConic(), _
        dAdx() As Double
    '*****************************
    ReDim xConic(nCellsConic), rConic(nCellsConic), aConic(nCellsConic)
        , _
         tanConic(nCellsConic), pConic(nCellsConic), tConic(
             nCellsConic), _
         rhoConic(nCellsConic), sConic(nCellsConic), hConic(
             nCellsConic), _
         machConic(nCellsConic), cConic(nCellsConic), ReConic(
             nCellsConic), _
         gammaConic(nCellsConic), muConic(nCellsConic), fConic(
             nCellsConic), _
         dAdx(nCellsConic)
    '*****************************

    'Compute conical section length
    AngleConic = Atn((rCrmc(EndCRMCIndex) - rCrmc(EndCRMCIndex - 1)) /
        (dx / 1000)) * 180 / Pi
    Lconic = ((OutSectDiam / 2) - rCrmc(EndCRMCIndex)) / Tan(AngleConic
        * Pi / 180)
    'Define dx along conical section [mm]
    dxConic = Lconic / nCellsConic * 1000
    drConst = dxConic * Tan(AngleConic * Pi / 180) / 1000

    'RK integration
    derivs_flag = "Conic"
```

```
    Call RK_NR(cCrmc(EndCRMCIndex), 0, Lconic, dxConic / 1000, xConic,
         cConic)
    derivs_flag = ""

    'Thermodynamic properties computing
    For i = 0 To nCellsConic
        aConic(i) = Acon(xConic(i))
        'Compute thermodynamic properties
        rhoConic(i) = frho(cConic(i), aConic(i), mDffsr)
        hConic(i) = fh(cConic(i), h_inlet, v_inlet)
        pConic(i) = Pressure(fluid, "dh", "SI", rhoConic(i), hConic(i))
        tConic(i) = Temperature(fluid, "dh", "SI", rhoConic(i), hConic(
            i))
        sConic(i) = Entropy(fluid, "ph", "SI", pConic(i), hConic(i))
        'Compute driving potentials variables
        machConic(i) = fM(cConic(i), SpeedOfSound(fluid, "ph", "SI",
            pConic(i), hConic(i)))
        rConic(i) = (aConic(i) / Pi) ^ 0.5
        dAdx(i) = dAdxConic(xConic(i))
        gammaConic(i) = cp(fluid, "ph", "SI", pConic(i), hConic(i)) /
            cv(fluid, "ph", "SI", pConic(i), hConic(i))
        'Compute Fanning friction factor
        muConic(i) = Viscosity(fluid, "dh", "SI", rhoConic(i), hConic(i
            ))
        ReConic(i) = fRe(cConic(i), rhoConic(i), rConic(i), muConic(i))
        fConic(i) = fSerg(RelRough, ReConic(i)) / 4
    Next

'Compute efficiency
Deltah = hConic(i - 1) - hCrmc(0)
hid = Enthalpy(fluid, "ps", "SI", pConic(i - 1), sCrmc(0))
DeltahId = hid - hCrmc(0)
eta = DeltahId / Deltah

'*********************************
'Print results
'*********************************
With Worksheets("CRMC_Diffuser")
    .Range("a:a").ClearContents
    .Range("b:b").ClearContents
    .Range("c:c").ClearContents
    .Range("d:d").ClearContents
    .Range("e:e").ClearContents
    .Range("f:f").ClearContents
    .Range("g:g").ClearContents
    .Range("h:h").ClearContents
    .Range("i:i").ClearContents
    .Range("j:j").ClearContents
    .Range("k:k").ClearContents
    .Range("l:l").ClearContents
    .Range("m:m").ClearContents
    '.Range("N:N").ClearContents
'Assign header - first row
    .Cells(1, 1).Value = "x_[mm]"
    .Cells(1, 2).Value = "dx_[mm]"
    .Cells(1, 3).Value = "r_[mm]"
    .Cells(1, 4).Value = "T_[K]"
    .Cells(1, 5).Value = "P_[MPa]"
    .Cells(1, 6).Value = "h_[kJ/kg]"
    .Cells(1, 7).Value = "rho_[kg/m3]"
    .Cells(1, 8).Value = "c_[m/s]"
    .Cells(1, 9).Value = "Mach"
    .Cells(1, 10).Value = "s_[kJ/kg_K]"
    .Cells(1, 11).Value = "Re"
    .Cells(1, 12).Value = "Serghide_friction"
    .Cells(1, 13).Value = "Eta"
    .Cells(2, 13).Value = eta
```

```
    .Cells(1, 14).Value = "dv/dx [1/s]"
    For i = 0 To EndCRMCIndex
        .Cells(i + 2, 1).Value = xCrmc(i) * 1000
        .Cells(i + 2, 2).Value = dx
        .Cells(i + 2, 3).Value = rCrmc(i) * 1000
        .Cells(i + 2, 4).Value = tCrmc(i)
        .Cells(i + 2, 5).Value = pCrmc(i)
        .Cells(i + 2, 6).Value = hCrmc(i)
        .Cells(i + 2, 7).Value = rhoCrmc(i)
        .Cells(i + 2, 8).Value = cCrmc(i)
        .Cells(i + 2, 9).Value = machCrmc(i)
        .Cells(i + 2, 10).Value = sCrmc(i)
        .Cells(i + 2, 11).Value = ReCrmc(i)
        .Cells(i + 2, 12).Value = fCrmc(i)
    Next
    For i = 0 To nCellsConic
        .Cells(i + EndCRMCIndex + 2, 1).Value = (xCrmc(EndCRMCIndex) +
            xConic(i)) * 1000
        .Cells(i + EndCRMCIndex + 2, 2).Value = dxConic
        .Cells(i + EndCRMCIndex + 2, 3).Value = rConic(i) * 1000
        .Cells(i + EndCRMCIndex + 2, 4).Value = tConic(i)
        .Cells(i + EndCRMCIndex + 2, 5).Value = pConic(i)
        .Cells(i + EndCRMCIndex + 2, 6).Value = hConic(i)
        .Cells(i + EndCRMCIndex + 2, 7).Value = rhoConic(i)
        .Cells(i + EndCRMCIndex + 2, 8).Value = cConic(i)
        .Cells(i + EndCRMCIndex + 2, 9).Value = machConic(i)
        .Cells(i + EndCRMCIndex + 2, 10).Value = sConic(i)
        .Cells(i + EndCRMCIndex + 2, 11).Value = ReConic(i)
        .Cells(i + EndCRMCIndex + 2, 12).Value = fConic(i)
    Next
End With

'*********************************
'Assign output values
'*********************************
hOut = hConic(i - 2)
CrmcDffsr = pConic(i - 2)

End Function

Function fdAdx(M, A, V, dVdx, r, f, gam)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute driving potentials function for dA/dx
'Version:     1.0 (2011/06/23) - (DP) firts version
'*********************************
'*********************************
'INPUT
'M       Mach number
'A       Section
'V       Gas speed
'dVdx    Gas speed derivative
'D       Duct diameter
'f       Fanning friction factor
'gam     specific heat ratio
'OUTPUT
'fdAdx   value of the driving potential function
'*********************************

fdAdx = ((M ^ 2 - 1) * A * dVdx / V) + _
        (Pi * M ^ 2 * r * f)
        '(Pi * gam * M ^ 2 * r * f)


End Function

Function fdVdx(M, A, V, dAdx, r, f, gam)
```

```
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute driving potentials function for dV/dx
'Version:     1.0 (2011/07/04) - (DP) firts version
'*********************************
'*********************************
'INPUT
'M        Mach number
'A        Section
'V        Gas speed
'dAdx     Section derivative
'D        Duct diameter
'f        Fanning friction factor
'gam      specific heat ratio
'OUTPUT
'fdVdx    value of the driving potential function
'*********************************

fdVdx = (V * dAdx / A / (M ^ 2 - 1)) + _
        (M ^ 2 * V * f / r / (1 - M ^ 2))
        '(gam * M ^ 2 * V * f / r / (1 - M ^ 2))

End Function

Function fM(V, Vs)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute Mach number (function of x)
'Version:     1.0 (2011/07/04) - (DP) firts version
'*********************************
'*********************************
'INPUT
'V        gas speed
'Vs       speed of sound
'OUTPUT
'M        value of the Mach number
'*********************************

fM = V / Vs

End Function

Function Vcrmc(x)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute velocity (function of x)
'Version:     1.0 (2011/07/04) - (DP) firts version
'Version:     1.1 (2011/07/05) - (DP) function name modified (Vel ->
     Vcrmc)
'*********************************
'*********************************
'INPUT
'x        position along the duct
'OUTPUT
'Vcrmc    value of the gas speed
'*********************************

xb = (Lcrmc * Lratio)
If (x < xb) Then
    Vcrmc = v_inlet + (dvdx_mix * x) + _
            ((dvdx_crmc - dvdx_mix) / (2 * xb)) * x ^ 2
Else
    Vb = v_inlet + (dvdx_mix * xb) + _
         ((dvdx_crmc - dvdx_mix) / (2 * xb)) * xb ^ 2
    Vcrmc = Vb + dvdx_crmc * (x - xb)
End If
```

```
End Function

Function dVdxcrmc(x)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute velocity derivative
'Version:     1.0 (2011/07/04) - (DP) firts version
'*********************************
'*********************************
'INPUT
'x        position along the duct
'OUTPUT
'dVdx     value of the gas speed derivative
'*********************************

If (x < (Lcrmc * Lratio)) Then
    dVdxcrmc = dvdx_mix + ((dvdx_crmc - dvdx_mix) / (Lcrmc * Lratio)) *
        x
Else
    dVdxcrmc = dvdx_crmc
End If

End Function

Function Acon(x)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute section (function of x)
'Version:     1.0 (2011/07/05) - (DP) firts version
'*********************************
'*********************************
'INPUT
'x        position along the duct
'OUTPUT
'Vcrmc    value of the gas speed
'*********************************

r = (OutSectDiam / 2) - Tan(AngleConic * Pi / 180) * (Lconic - x)
Acon = Pi * r ^ 2

End Function

Function dAdxConic(x)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute section derivative
'Version:     1.0 (2011/07/05) - (DP) firts version
'*********************************
'*********************************
'INPUT
'x        position along the duct
'OUTPUT
'dVdx     value of the gas speed derivative
'*********************************

r = (OutSectDiam / 2) - Tan(AngleConic * Pi / 180) * (Lconic - x)
dAdxConic = 2 * Pi * r * Tan(AngleConic * Pi / 180)

End Function

Function frho(V, A, mp)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute density
'Version:     1.0 (2011/07/04) - (DP) firts version
'*********************************
'*********************************
```

```
'INPUT
'V        gas speed
'A        section
'mp       mass flow rate
'OUTPUT
'rho      gas density
'*********************************

frho = mp / V / A

End Function

Function fh(V, h_inlet, v_inlet)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute total enthalpy
'Version:     1.0 (2011/07/04) - (DP) firts version
'*********************************
'*********************************
'INPUT
'V            gas speed
'hinit        duct inlet total enthalpy
'v_inlet      duct gas speed at inlet section
'OUTPUT
'h            gas total enthalpy
'*********************************

fh = h_inlet + (v_inlet ^ 2 - V ^ 2) / 2000

End Function

Function fRe(V, rho, r, mu)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute Reynolds
'Version:     1.0 (2011/07/04) - (DP) firts version
'*********************************
'*********************************
'INPUT
'V        gas speed
'rho      density
'D        duct diameter
'mu       viscosity
'OUTPUT
'Re       Reynolds number
'*********************************

fRe = 10 ^ 6 * rho * V * 2 * r / mu

End Function

'*********************************
'***          Module header         ***
'*********************************
'Authors:     Adriano Milazzo (AM), Dario Paganini (DP)
'Contents:    Runge-Kutta integration subroutines
'Suboutines and functions included:
'-------------------------------------
'Name              Version
'-------------------------------------
'RK_NR             2.1 (2011/07/04)
'RK4_NR            1.0 (2011/04/21)
'derivs            2.0 (2011/07/05)
'*********************************
'*********************************

Sub RK_NR(vstart, x1, x2, h, xx, y)
```

```
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Runge-Kutta integration driver
'Version:     1.0 (2011/04/23) - (DP) first version
'Version:     2.0 (2011/05/12) - (DP) input parameter change: h instead
     of nstep,
'                nstep is computed using h
'Version:     2.1 (2011/07/04) - (DP) change V variable in RK_V
'Version:     2.2 (2011/07/08) - (DP) nstep computed using round function
      insead
'                int function
'*********************************
'*********************************
' Runge-Kutta integration from "Numerical Recipes" William H. Press et
     al.
'*********************************
'*********************************
'INPUT
'vstart
'x1       start position
'x2       end position
'h        integration step length
'OUTPUT
'xx       array of the output abscissas
'y        array of the integrated values
'*********************************

RK_V = vstart
y(0) = RK_V
xx(0) = x1

x = x1
nstep = Round(Abs((x2 - x1) / h))

For k = 1 To nstep
    Call derivs(x, RK_V, Dv)
    Call RK4_NR(RK_V, Dv, x, h, RK_V)
    If (x + h = x) Then MsgBox ("Error in stepsize")
    x = x + h
    xx(k) = x
    y(k) = RK_V
Next

End Sub

Sub RK4_NR(y, dydx, x, h, yout)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Runge-Kutta integration step
'Version:     1.0 (2011/04/21) - (DP) first version
'*********************************
'*********************************
' Runge-Kutta integration from "Numerical Recipes" William H. Press et
     al.
'*********************************
'*********************************
'INPUT
'y        function value at x
'dydx     derivatives value at x
'x        position
'h        step dimension
'yout     function value at x+h
'i        alement index of the integrated array
'OUTPUT
'*********************************

Dim dym, dyt, yt As Double
```

```
hh = h * 0.5
h6 = h / 6
xh = x + hh

'First step
yt = y + hh * dydx
'Second step
Call derivs(xh, yt, dyt)
yt = y + hh * dyt
'Third step
Call derivs(xh, yt, dym)
yt = y + h * dym
dym = dym + dyt
'Fourth step
Call derivs(x + h, yt, dyt)
yout = y + h6 * (dydx + dyt + 2# * dym)

End Sub


Sub derivs(x, y, dydx)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute derivatives dydx at x
'Version:     1.0 (2011/04/21) - (DP) first version
'Version:     2.0 (2011/07/05) - (DP) modified for CrmcDffsr function
'                 added flag to select derivative (used as public
    variable)
'Version:     2.1 (2011/11/14) - (DP) corrected friction factor f
    assignment
'                 (Fanning friction factor expected, Darcy passed)
'*********************************
'*********************************
'INPUT
'
'OUTPUT
'*********************************

Select Case derivs_flag
    Case "Crmc"
        'Assign speed and section
        A = y
        V = Vcrmc(x)
        dVdx = dVdxcrmc(x)
        'Compute thermodynamic properties
        rho = frho(V, A, mDffsr)
        h = fh(V, h_inlet, v_inlet)
        P = Pressure(fluid, "dh", "SI", rho, h)
        'Copmpute driving potentials variables
        M = fM(V, SpeedOfSound(fluid, "ph", "SI", P, h))
        r = (A / Pi) ^ 0.5
        gam = cp(fluid, "ph", "SI", P, h) / cv(fluid, "ph", "SI", P, h)
        'Compute Fanning friction factor
        mu = Viscosity(fluid, "dh", "SI", rho, h)
        Re = fRe(V, rho, r, mu)
        f = fSerg(RelRough, Re) / 4
        'Compute driving potentials function
        dydx = fdAdx(M, A, V, dVdx, r, f, gam)
    Case "Conic"
        'Assign speed and section
        V = y
        A = Acon(x)
        dAdx = dAdxConic(x)
        'Compute thermodynamic properties
        rho = frho(V, A, mDffsr)
        h = fh(V, h_inlet, v_inlet)
```

```
        P = Pressure(fluid, "dh", "SI", rho, h)
        'Copmpute driving potentials variables
        M = fM(V, SpeedOfSound(fluid, "ph", "SI", P, h))
        r = (A / Pi) ^ 0.5
        gam = cp(fluid, "ph", "SI", P, h) / cv(fluid, "ph", "SI", P, h)
        'Compute Fanning friction factor
        mu = Viscosity(fluid, "dh", "SI", rho, h)
        Re = fRe(V, rho, r, mu)
        f = fSerg(RelRough, Re) / 4
        'Compute driving potentials function
        dydx = fdVdx(M, A, V, dAdx, r, f, gam)
    Case Else
        derivs_err = 1
        derivs_msg = "Error in derivative selection option"
End Select

End Sub


Function fSerg(RelRough, ReynoldsNum)
'Friction Factor calulated by T.K. Serghide's implementation of
    Steffenson
'Reference Chemical Engineering March 5, 1984
Dim A As Single
Dim B As Single
Dim c As Single
'Note that in Visual Basic "Log" stands for Natural Log
A = -0.86858896 * Log(RelRough / 3.7 + 12 / ReynoldsNum)
B = -0.86858896 * Log(RelRough / 3.7 + (2.51 * A / ReynoldsNum))
c = -0.86858896 * Log(RelRough / 3.7 + (2.51 * B / ReynoldsNum))
fSerg = (A - ((B - A) ^ 2) / (c - (2 * B) + A)) ^ -2

End Function
```

Appendix C

Nozzle efficiency code

```
Attribute VB_Name = "Modulo1"
'**********************************
'Authors:    Adriano Milazzo (AM), Dario Paganini (DP)
'Contents:   Subroutines to compute supersonic nozzle (eventually with
     normal
'            shock wave) starting assigning inlet to outlet pressure
     ratio
'Suboutines and functions included:
'-----------------------------------
'Name              Version
'-----------------------------------
'NozSim            1.6 (2012/11/16)
'nozzle            3.0 (2011/05/12)
'nozzleProp        1.1 (2011/05/16)
'ShockJump         1.1 (2011/04/07)
'DmdxSonic         not defined
'NozGeoDef         1.0 (2011/04/21)
'**********************************
'**********************************

Public Pi, GAM, PEX, f, ISHOCK, X_arr(), XM_arr(), PSTAG_arr(), PS_arr
     (), _
    TSTAG_arr(), TS_arr(), XMINAREA, NozRin, NozRout, NozRth, NozAlpha,
         _
    NozRfillet, xchange, ychange, dmdxmax, xsp, DXSAVE

Sub NozSim()
'**********************************
'Author:     Dario Paganini (DP)
'Purpose:    Solve nozzle for different back pressure values
'Version:    1.0 (2011/04/06) - (DP) first version
'Version:    1.1 (2011/04/21) - (DP) added nozzle geometry definition
'Version:    1.2 (2011/05/11) - (DP) removed Xsonic subroutine call (XSP
'                computed analiticaly)
'Version:    1.3 (2011/05/20) - (DP) removed bug in Pbstar computing
'Version:    1.4 (2011/06/16) - (DP) added threshold to treat as
     isentropic if
'                f < 0.0005
'                added automatic version printing on worksheet
'                removed unused code
'Version:    1.5 (2011/09/15) - (DP) Dimensional properties introduced
'                modified NozProp worksheet and data reading from it
```

```
'Version:    1.6 (2012/11/16) - (DP) Added decimation on printed output
    data
'                to reduce exectuion time
'*********************************
'*********************************
' Algorithms and relations are got from B.K.Hodge and K.Koenig "
    Compressible
' fluid dynamics with personal computer applications"
'*********************************
'*********************************
'INPUT
'no inputs
'
'OUTPUT
'output are printed on worksheets.
'*********************************
    'Define program version
    NozSimVer = "1.6 (2012/11/16)"

    'Constants definition
    Pi = 3.1415926

    'Define nozzle geometry
    Call NozGeoDef(XMINAREA, XMAX)

    'Data read from worksheet
    With Worksheets("NozProp")
        GAM = .Cells(3, 2).Value
        cp = .Cells(4, 2).Value
        f = .Cells(5, 2).Value
        'Read initial stagnation properties
        Tstag0 = .Cells(3, 4).Value
        Pstag0 = .Cells(4, 4).Value
        'Read backpressure
        Pb = .Cells(5, 4).Value
        'Clear previously computed data
        .Range("A10:A65000").ClearContents
        .Range("B10:B65000").ClearContents
        .Range("C10:C65000").ClearContents
        .Range("D10:D65000").ClearContents
        .Range("E10:E65000").ClearContents
        .Range("F10:F65000").ClearContents
        .Range("G10:G65000").ClearContents
        .Range("H10:H65000").ClearContents
        .Range("I10:I65000").ClearContents
        .Range("J10:J65000").ClearContents
    End With

    'Pressure ratio
    PREF = Pb / Pstag0


    'Compute sonic point location
    If (f < 0.0005) Then
        'Number of integration steps between minarea and inlet section
        DXnum = 100
        xsp = XMINAREA
        dx = XMINAREA / DXnum
        f = 0
    Else
        'Number of integration steps between minarea and throat section
        DXnum = 2
        xsp = XMINAREA + NozRfillet * (1 / ((4 / GAM / GAM / f / f) -
            1)) ^ 0.5
        dx = (xsp - XMINAREA) / DXnum
    End If
    If (xsp > xchange) Then Stop 'Sonic point in the conical section
```

```
'Write integration step on worksheet
With Worksheets("NozProp")
    .Cells(6, 2).Value = dx
End With

'Call function to determine dM/dx at sonic point (M=1)
dmdxmax = DmdxSonic(xsp, dx)

'Backward integration from sonic point to find initial Mach number
Call nozzle(-1, xsp, dmdxmax, -dx, xsp, 0, 1, xback, mback)

'Assign properties initial values
nback = UBound(xback)
xi = xback(nback)
XMI = mback(nback)
Tstagi = TSTAGFN(0#)
Tstati = TSTAGFN(0#) / ET(XMI)
Pstagi = 1#
Pstati = Pstagi / (ET(XMI) ^ (GAM / (GAM - 1)))

Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi, xback,
    mback, Tstatback, Tstagback, Pstatback, Pstagback)

'Call three times the nozzle subroutine to define backpressure
    intervals
'for which different shock waves behaviour is present
'Find Pbd value (shock-free flow)
xshock = -1
Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1, xfor, mfor)
Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi, xfor, mfor
    , Tstat, Tstag, Pstat, Pstag)
Pbd = Pstat(UBound(Pstat))

'Find Pbstar value (normal shock immediately after the throat)
xshock = xsp + dx * (1 + 0.01) 'Add 1% to dx to avoid rounding
    error
Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1, xfor, mfor)
Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi, xfor, mfor
    , Tstat, Tstag, Pstat, Pstag)
Pbstar = Pstat(UBound(Pstat))

'Find Pbc value (normal shock at the exit plane)
xshock = XMAX
Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1, xfor, mfor)
Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi, xfor, mfor
    , Tstat, Tstag, Pstat, Pstag)
Pbc = Pstat(UBound(Pstat))
'Stop

'Identify shock wave configuration
If (PREF <= 1) And (PREF >= Pbstar) Then
    ShockWaveString = "SUBSONIC_FLOW"
End If
If (PREF < Pbstar) And (PREF >= Pbc) Then
    ShockWaveString = "SHOCK_IN_NOZZLE"
End If
If (PREF < Pbc) And (PREF > Pbd) Then
    ShockWaveString = "OBLIQUE_SHOCK_AT_EXIT"
End If
If (PREF = Pbd) Then
    ShockWaveString = "SHOCK-FREE_FLOW"
End If
If (PREF < Pbd) Then
    ShockWaveString = "OUTSIDE_EXPANSION_WAVE"
End If
```

```
    Select Case ShockWaveString
        Case "SUBSONIC_FLOW"
            'Call nozzle(-1, -1, dmdxmax, dx, 0, XMAX, XMI / 100, xsub,
                msub)
            'Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi,
                xsub, msub, Tstatsub, Tstagsub, Pstatsub, Pstagsub)
            'nsub = UBound(xsub)
            ISHOCK = -1

        Case "SHOCK_IN_NOZZLE"
            xshock = xsp + dx * (1 + 0.01) 'Add 1% to dx to avoid
                rounding error
            DELTAX = (XMAX - xsp) / 10
            flag = 0
            Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1, xfor,
                mfor)
            Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi,
                xfor, mfor, Tstat, Tstag, Pstat, Pstag)

1000
            DELTAP = Pstat(UBound(Pstat)) - PREF
            If Abs(DELTAP) < 0.0001 Or DELTAX < dx Then GoTo 1001
            If DELTAP > 0 Then
                If flag < 0 Then DELTAX = 0.5 * DELTAX
                xshock = xshock + DELTAX
                Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1,
                    xfor, mfor)
                Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi
                    , xfor, mfor, Tstat, Tstag, Pstat, Pstag)
                flag = 1
            Else
                If flag > 0 Then DELTAX = 0.5 * DELTAX
                xshock = xshock - DELTAX
                Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1,
                    xfor, mfor)
                Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi
                    , xfor, mfor, Tstat, Tstag, Pstat, Pstag)
                flag = -1
            End If

            GoTo 1000
1001
            nfor = UBound(xfor)

        Case "OBLIQUE_SHOCK_AT_EXIT"
            xshock = -1
            Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1, xfor,
                mfor)
            Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi,
                xfor, mfor, Tstat, Tstag, Pstat, Pstag)
            nfor = UBound(xfor)
            ISHOCK = -1

        Case "SHOCK-FREE_FLOW"
            xshock = -1
            Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1, xfor,
                mfor)
            Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi,
                xfor, mfor, Tstat, Tstag, Pstat, Pstag)
            nfor = UBound(xfor)
            ISHOCK = -1

        Case "OUTSIDE_EXPANSION_WAVE"
            xshock = -1
            Call nozzle(xshock, xsp, dmdxmax, dx, xsp, XMAX, 1, xfor,
                mfor)
```

```
            Call nozzleProp(xi, XMI, Tstati, Tstagi, Pstati, Pstagi,
                 xfor, mfor, Tstat, Tstag, Pstat, Pstag)
            nfor = UBound(xfor)
            ISHOCK = -1

        Case Else
            ERFLAG = 1
            ERMSG = "ERROR_IN_PRESSURE_RANGE"
    End Select

    If (ShockWaveString <> "SUBSONIC_FLOW") Then
        ntot = nback + nfor
        'Properties array definition
        ReDim X_arr(ntot)
        ReDim XM_arr(ntot)
        ReDim PSTAG_arr(ntot)
        ReDim PS_arr(ntot)
        ReDim TSTAG_arr(ntot)
        ReDim TS_arr(ntot)
        For k = 0 To ntot
            If (k > nback) Then
                X_arr(k) = xfor(k - nback)
                XM_arr(k) = mfor(k - nback)
                PSTAG_arr(k) = Pstag(k - nback)
                PS_arr(k) = Pstat(k - nback)
                TSTAG_arr(k) = Tstag(k - nback)
                TS_arr(k) = Tstat(k - nback)
            Else
                X_arr(k) = xback(nback - k)
                XM_arr(k) = mback(nback - k)
                PSTAG_arr(k) = Pstagback(nback - k)
                PS_arr(k) = Pstatback(nback - k)
                TSTAG_arr(k) = Tstagback(nback - k)
                TS_arr(k) = Tstatback(nback - k)
            End If
        Next
        'Compute isentropic effciency
        eta = (1 - TS_arr(ntot)) / (1 - (PS_arr(ntot) ^ ((GAM - 1) /
             GAM)))
        With Worksheets("NozProp")
            .Cells(1, 4).Value = NozSimVer
            .Cells(4, 6).Value = xsp
            .Cells(4, 7).Value = dmdxmax
            .Cells(4, 8).Value = XM_arr(0)
            .Cells(4, 9).Value = xshock
            .Cells(4, 10).Value = eta
            .Cells(7, 1).Value = ShockWaveString
            For I = 0 To ntot
                If ((I Mod 100) = 0) Then
                .Cells(10 + I / 100, 1).Value = I + 1
                .Cells(10 + I / 100, 2).Value = X_arr(I)
                .Cells(10 + I / 100, 3).Value = XM_arr(I)
                .Cells(10 + I / 100, 4).Value = PSTAG_arr(I) * Pstag0
                .Cells(10 + I / 100, 5).Value = PS_arr(I) * Pstag0
                .Cells(10 + I / 100, 6).Value = TSTAG_arr(I) * Tstag0
                .Cells(10 + I / 100, 7).Value = TS_arr(I) * Tstag0
                End If
                If (I = ISHOCK) And (ISHOCK > 0) Then
                    .Cells(10 + I, 8).Value = "NORMAL_SHOCK_WAVE"
                End If
            Next
        End With
    Else
    '     ntot = nsub
    '     ReDim X_arr(nsub)
    '     ReDim XM_arr(nsub)
    '     ReDim PSTAG_arr(nsub)
```

```
'     ReDim PS_arr(nsub)
'     ReDim TSTAG_arr(nsub)
'     ReDim TS_arr(nsub)
'     For k = 0 To nsub
'         X_arr(k) = xsub(k)
'         XM_arr(k) = msub(k)
'         PSTAG_arr(k) = Pstagsub(k)
'         PS_arr(k) = Pstatsub(k)
'         TSTAG_arr(k) = Tstagsub(k)
'         TS_arr(k) = Tstatsub(k)
'     Next
'     With Worksheets("NozProp")
'         For I = 0 To ntot
'             .Cells(10 + I, 1).Value = I + 1
'             .Cells(10 + I, 2).Value = X_arr(I)
'             .Cells(10 + I, 3).Value = XM_arr(I)
'             .Cells(10 + I, 4).Value = PSTAG_arr(I)
'             .Cells(10 + I, 5).Value = PS_arr(I)
'             .Cells(10 + I, 6).Value = TSTAG_arr(I)
'             .Cells(10 + I, 7).Value = TS_arr(I)
'         Next
'     End With
    End If

    'Stop

End Sub

Sub nozzle(xshock, xsp, dmdxmax, h, xstart, xend, ystart, xout, yout)
'*********************************
'Author:      Adriano Milazzo (AM), Dario Paganini (DP)
'Purpose:     Simulate fluid flow through converging-diverging nozzle
'Version:     1.0 (2011/03/22) - (AM,DP) first version
'Version:     2.0 (2011/04/05) - (DP) added XSHOCK as input
'                pressure and temperature values passed as Public array
    instead
'                of printed on worksheet
'Version:     2.1 (2011/04/07) - (DP) XSP and DMDXMAX passed as
    parameters
'                removed call to subroutine Xsonic and DmdxSonic
'Version:     3.0 (2011/05/12) - (DP) integration subroutine called
    change
'                I/O parameters modified
'*********************************
'*********************************
' Algorithms and relations are got from B.K.Hodge and K.Koenig "
    Compressible
' fluid dynamics with personal computer applications"
'*********************************
'*********************************
'INPUT
'xshock      normal shock position (negative value if no shock present)
'xsp         sonic point position (negative value if subsonic flow)
'dmdxmax     dM/dx value at sonic point
'h           integration step
'xstart      integration starting point
'xend        integration ending point
'ystart      function starting value
'
'OUTPUT
'xout        positions array
'yout        function values array
'*********************************
    If ((xshock < 0) Or (xshock > xend)) Then    'Without shock
        nstep = Int(Abs((xend - xstart) / h))
        ReDim xout(nstep), yout(nstep)
        Call RK_NR(ystart, xstart, xend, h, xout, yout)
```

```
    Else                                                    'With shock
        'Forward integration from throat to shock
        nstep1 = Int(Abs((xshock - xstart) / h))
        Dim x1(), y1() As Double
        ReDim x1(nstep1), y1(nstep1)
        Call RK_NR(ystart, xstart, xshock, h, x1, y1)
        'Forward integration from shock to end
        yshock = y1(nstep1)
        Call ShockJump(IIS, I, yshock, xshock, TS, PS, TSI, PSI)
        nstep2 = Int(Abs((xend - xshock) / h))
        Dim x2(), y2() As Double
        ReDim x2(nstep2), y2(nstep2)
        Call RK_NR(yshock, xshock, xend, h, x2, y2)
        nstep = nstep1 + nstep2
        ReDim xout(nstep), yout(nstep)
        For k = 0 To nstep
            If k < nstep1 Then
                xout(k) = x1(k)
                yout(k) = y1(k)
            Else
                xout(k) = x2(k - nstep1)
                yout(k) = y2(k - nstep1)
            End If
        Next
    End If

End Sub

Sub nozzleProp(xi, mi, Tstati, Tstagi, Pstati, Pstagi, x, m, Tstat,
    Tstag, Pstat, Pstag)
'**********************************
'Author:     Adriano Milazzo (AM), Dario Paganini (DP)
'Purpose:    Compute properties in the nozzle starting from Mach numbers
'Version:    1.0 (2011/05/13) - (DP) first version
'Version:    1.1 (2011/05/16) - (DP) removed xshock and xsp as input
    parameters
'**********************************
'**********************************
' Algorithms and relations are got from B.K. Hodge and K. Koenig "
    Compressible
' fluid dynamics with personal computer applications"
'**********************************
'**********************************
'INPUT
'xi          initial position
'mi          initial Mach number
'
'OUTPUT
'Tstat       Static temperature
'Tstag       Stagnation temperature
'Pstat       Static pressure
'Pstag       Stagnation pressure
'**********************************
    nstep = UBound(x)
    ReDim Tstat(nstep), Tstag(nstep), Pstat(nstep), Pstag(nstep)

    For k = 0 To nstep
        T2OT1 = TSTAGFN(x(k)) * ET(mi) / TSTAGFN(xi) / ET(m(k))
        Tstat(k) = T2OT1 * Tstati
        Tstag(k) = TSTAGFN(x(k))
        P2OP1 = MDOT(x(k)) * AREA(xi) * mi * Sqr(T2OT1) / (MDOT(xi) *
            AREA(x(k)) * m(k))
        Pstat(k) = P2OP1 * Pstati
        Pstag(k) = P2OP1 * Pstagi * (ET(m(k)) / ET(mi)) ^ (GAM / (GAM -
            1))
    Next
```

```
End Sub

Function DmdxSonic(xsp, dx)
    'EVALUATION OF DMDXMAX FOR SONIC POINT EVALUATION
    XSPP = xsp + dx / 10
    XSPM = xsp - dx / 10
    DGDX = (G(AREADP(XSPP), FRICDP(XSPP, f), HEATDP(XSPP), MASSDP(XSPP)
        , XSPP, 1!, GAM) - G(AREADP(XSPM), FRICDP(XSPM, f), HEATDP(
        XSPM), MASSDP(XSPM), XSPM, 1!, GAM)) / (XSPP - XSPM)
    CTERM = (GAM + 1!) * DGDX / 8!
    BTERM = (GAM + 1!) * (2! * GAM / 8!) * (FRICDP(xsp, f) + HEATDP(xsp
        ) + 2! * MASSDP(xsp))
    DMDX1 = -0.5 * BTERM + 0.5 * Sqr(BTERM * BTERM - 4! * CTERM): DMDX2
        = -0.5 * BTERM - Sqr(BTERM * BTERM - 4! * CTERM)
    dmdxmax = DMDX1 'DMDXMAX FOR SUPERSONIC FLOW CHOSEN
    DmdxSonic = dmdxmax
End Function

Sub ShockJump(IIS, I, XM, x, TS, PS, TSI, PSI)
'*********************************
'Author:      Adriano Milazzo (AM), Dario Paganini (DP)
'Purpose:     Compute properties through shock wave
'Version:     1.0 (2011/03/22) - (AM,DP) first version
'Version:     1.1 (2011/04/07) - (DP) properties saved on arrays instead
    of
'                printed on worksheet
'*********************************
'*********************************
' Algorithms and relations are got from B.K.Hodge and K.Koenig "
    Compressible
' fluid dynamics with personal computer applications"
'*********************************
'*********************************
'INPUT
'IIS      shock wave flag
'I        iteration index
'XM       Mach number before shock
'X        shock position
'TS       Static temperature after shock
'PS       Static pressure after shock
'TSI      Static temperature before shock
'PSI      Static temperature before shock
'
'OUTPUT
'properties are saved on Public properties arrays
'*********************************
    'SHOCK WAVE JUMP CONDITIONS
    IIS = 1
    MO = XM
    test = ((XM * XM + 2! / (GAM - 1!)) / (2! * GAM / (GAM - 1!) * XM *
        XM - 1!))
    XM = Sqr((XM * XM + 2! / (GAM - 1!)) / (2! * GAM / (GAM - 1!) * XM
        * XM - 1!))
    'PS = PS * (2! * GAM / (GAM + 1!) * MO * MO - (GAM - 1!) / (GAM +
        1!))
    'TS = TS * ET(MO) / ET(XM)
    'PSTAG = PS * ET(XM) ^ PEX
    'DELS = Log(TS / TSI) - (GAM - 1!) * Log(PS / PSI) / GAM

    'X_arr(I - 1 + IIS) = x
    'xM_arr(I - 1 + IIS) = XM
    'PSTAG_arr(I - 1 + IIS) = PSTAG
    'PS_arr(I - 1 + IIS) = PS
    'TSTAG_arr(I - 1 + IIS) = TSTAG
    'TS_arr(I - 1 + IIS) = TS

End Sub
```

```
Sub NozGeoDef(xTh, xout)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Nozzle geometry definition
'Version:     1.0 (2011/04/21) - (DP) first version
'*********************************
'*********************************
' Frigel nozzle geometry
'*********************************
'*********************************
'INPUT
'
'OUTPUT
'xTh          Position of the minimum area section
'xOut         Nozzle length (x-axis origin in 0)
'*********************************

With Worksheets("NozGeo")
    NozRin = .Cells(3, 1).Value
    NozRth = .Cells(3, 2).Value
    NozRout = .Cells(3, 3).Value
    NozRfillet = .Cells(3, 4).Value
    NozAlpha = .Cells(3, 5).Value
End With
'Find throat position
xTh = (NozRfillet ^ 2# - (NozRth + NozRfillet - NozRin) ^ 2#) ^ 0.5
'Find start position for linear profile
xchange = xTh + NozRfillet * Sin(NozAlpha * Pi / 180)
ychange = NozRth + NozRfillet * (1 - Cos(NozAlpha * Pi / 180))
'Find nozzle length
xout = Round(xchange + ((NozRout - ychange) / Tan(NozAlpha * Pi / 180))
    )
'Print nozzle profile values
NozStepLen = 0.5
NozStepNum = Int(xout / NozStepLen)
xtemp = 0
With Worksheets("NozGeo")
    .Range("G2:G10000").ClearContents
    .Range("H2:H10000").ClearContents
    For I = 0 To NozStepNum
        xtemp = (I * NozStepLen)
        .Cells(2 + I, 7).Value = xtemp
        If (xtemp < xchange) Then
            theta = WorksheetFunction.Asin((xtemp - xTh) / NozRfillet)
            .Cells(2 + I, 7).Value = xtemp
            .Cells(2 + I, 8).Value = NozRth + NozRfillet * (1 - Cos(
                theta))
        Else
            .Cells(2 + I, 7).Value = xtemp
            .Cells(2 + I, 8).Value = ychange + Tan(NozAlpha * Pi / 180)
                * (xtemp - xchange)
        End If
    Next
End With

End Sub


Attribute VB_Name = "Modulo2"
'*********************************
'Authors:     Adriano Milazzo (AM), Dario Paganini (DP)
'Contents:    Driving potential function and useful functions definitions
'Suboutines and functions included:
'-----------------------------------
'Name              Version
'-----------------------------------
'AREADP            1.0 (2011/04/21)
```

```
'HEATDP          not defined
'DIA1            1.0 (2011/04/21)
'MASSDP          not defined
'FRICDP          1.0 (2011/04/21)
'TSTAGFN         not defined
'AREA            1.0 (2011/04/21)
'MDOT            not defined
'G               not defined
'FM              not defined
'MT              not defined
'ET              not defined
'*********************************
'*********************************

Function AREADP(a)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute area driving potential
'Version:     1.0 (2011/04/21) - (DP) first version
'*********************************
'*********************************
'INPUT
'a       abscissa
'OUTPUT
'*********************************
    If (a < xchange) Then
        theta = WorksheetFunction.Asin((a - XMINAREA) / NozRfillet)
        r = NozRth + NozRfillet * (1 - Cos(theta))
        AREADP = (2 / r) * _
            ((a - XMINAREA) / (NozRfillet ^ 2# - (a - XMINAREA) ^ 2#) ^
                0.5)
    Else
        r = ychange + Tan(NozAlpha * Pi / 180) * (a - xchange)
        AREADP = (2 / r) * Tan(NozAlpha * Pi / 180)
    End If

    'AREADP = -Pi * Sin(Pi * a) / (1! + 0.5 * Cos(Pi * a))
End Function
Function HEATDP(a)
    HEATDP = 0!
End Function
Function DIA1(a)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute section diameter
'Version:     1.0 (2011/04/21) - (DP) first version
'*********************************
'*********************************
'INPUT
'a       abscissa
'OUTPUT
'*********************************
    If (a < xchange) Then
        theta = WorksheetFunction.Asin((a - XMINAREA) / NozRfillet)
        r = NozRth + NozRfillet * (1 - Cos(theta))
        DIA1 = 2 * r
    Else
        r = ychange + Tan(NozAlpha * Pi / 180) * (a - xchange)
        DIA1 = 2 * r
    End If
    'Stop

    'DIA1 = 1! + 0.5 * Cos(Pi * a)
End Function
Function MASSDP(a)
    MASSDP = 0!:
End Function
```

```
Function FRICDP(a, f)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute friction driving potential
'Version:     1.0 (2011/04/21) - (DP) first version
'*********************************
'*********************************
'INPUT
'a        abscissa
'OUTPUT
'*********************************
    If (a < xchange) Then
        theta = WorksheetFunction.Asin((a - XMINAREA) / NozRfillet)
        r = NozRth + NozRfillet * (1 - Cos(theta))
        FRICDP = 4! * f / (2 * r)
    Else
        r = ychange + Tan(NozAlpha * Pi / 180) * (a - xchange)
        FRICDP = 4! * f / (2 * r)
    End If
    'Stop

    'FRICDP = 4! * f / (1! + 0.5 * Cos(Pi * a))
End Function
Function TSTAGFN(a)
    TSTAGFN = 1!
End Function
Function AREA(a)
'*********************************
'Author:      Dario Paganini (DP)
'Purpose:     Compute friction factor
'Version:     1.0 (2011/04/21) - (DP) first version
'*********************************
'*********************************
'INPUT
'a        abscissa
'OUTPUT
'*********************************
    If (a < xchange) Then
        theta = WorksheetFunction.Asin((a - XMINAREA) / NozRfillet)
        r = NozRth + NozRfillet * (1 - Cos(theta))
        AREA = Pi * r ^ 2
    Else
        r = ychange + Tan(NozAlpha * Pi / 180) * (a - xchange)
        AREA = Pi * r ^ 2
    End If
    'Stop
    'AREA = 0.25 * Pi * (1! + 0.5 * Cos(Pi * a)) ^ 2
End Function
Function MDOT(a)
    MDOT = 1!
End Function
Function G(DPA, DPF, DPH, DPM, a, CM, GAM)
    G = 2! * (-DPA + 0.5 * GAM * CM * CM * DPF + 0.5 * (1! + GAM * CM *
        CM) * DPH + (1! + GAM * CM * CM) * DPM)
End Function

'DIFFERENTIAL EQUATION DEFINITION
Function FM(AET, AMT, DPA, DPF, DPH, DPM, a, CM, GAM)
    FM = CM * AET / AMT * (-DPA + 0.5 * GAM * CM * CM * DPF + 0.5 * (1!
        + GAM * CM * CM) * DPH + (1! + GAM * CM * CM) * DPM)
End Function

'USEFUL FUNCTION DEFINITIONS
Function MT(CM)
    MT = 1! - CM * CM
End Function
Function ET(CM)
```

```
    ET = 1! + 0.5 * (GAM - 1!) * CM * CM
End Function
```