A GENERAL TOOL FOR A BETTER USE OF DESIGN OR VERIFICATION MATHEMATICAL MODELS.

E. Ghiazza - ITALIMPIANTI
F. Giovani - ITALIMPIANTI

Abstract

It is quite common in desalination field to use more or less simple mathematical models (usually coded in standard procedural languages such as FORTRAN, C, etc.) both for design and for verification purposes.

For many reasons, during the evolution of a desalination plant construction, the need for calculation of design type (e.g. the heat exchange surfaces evaluation), of verification type (e.g. the evaluation of plant performances under certain operating conditions), and of hybrid type (e.g. the search for the proper operating conditions to reach desired performances) comes out.

The use of models of different kind is common to carry out these tasks, with corresponding purchase or development expenses and computer memory occupation.

By means of the proposed general software tool any available model can be used for design, verification or hybrid purposes, despite of the category the model belongs to.

A brief description of the tool, some examples and some considerations about its use and its advantages are reported. Moreover an application of the tool to a new optimization system for MSF plants is reported.

Introduction

The presented software tool has been developed in order to approach and deepen the three typical kinds of problems (design, verification and hybrid ones) commonly encountered in practice, and usually solved by proper mathematical models, as explained in the following.

To build the tool as generalized and robust as possible some basic guidelines have been observed, based on the principle that any software tool is generally developed to meet three important design criteria:

- *Correctness.* The tool must always yield the right answers, otherwise it is not effective. Moreover it must be designed to make errors immediately apparent.

- *Adaptability.* Most likely, the task performed by the tool will not remain the same forever. Thus, it must also be designed so that it can be easily modified to face new problems.
- *Speed.* Although speed is less important than accuracy, the tool must be designed for taking into account efficiency to speed recalculation.

Different kinds of problems in desalination field

Referring to the Fig. 1, a what-if analysis can be conducted, to understand which category the particular problem to solve belongs to; subject to different sets of input data and constraints (as highlighted by the lines direction), all the kinds of problems are usually solved using a proper model.

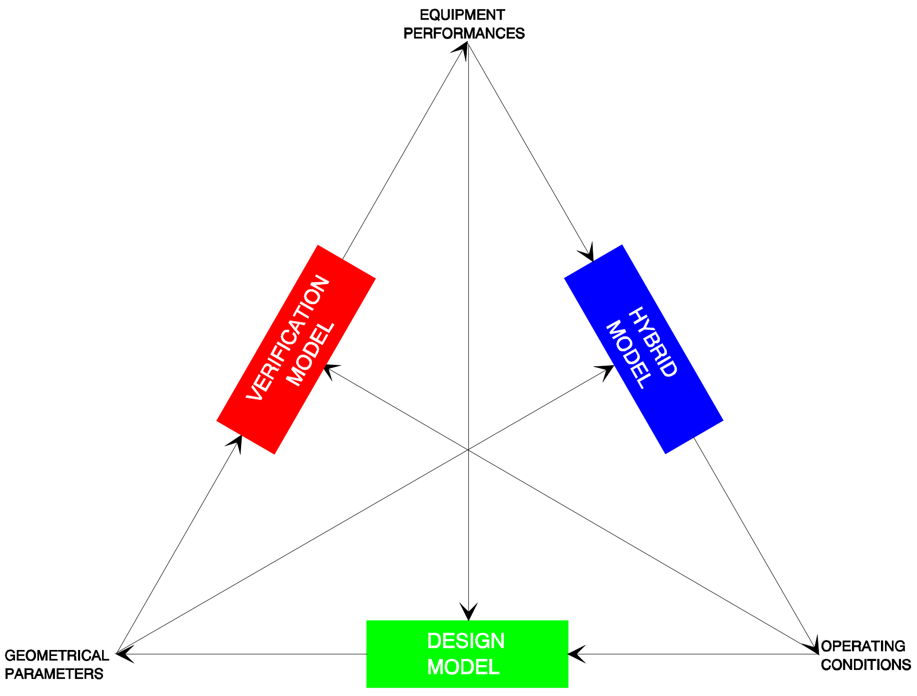


Fig. 1: Different categories of models

The three main categories of problems can be roughly outlined as follows:

Design problem:	starting from the desired equipment performances and fixed operating conditions, the proper geometrical parameters have to be evaluated.
Verification problem:	starting from fixed geometrical parameters and given operating conditions, the actual equipment performances have to be evaluated.

Hybrid problem:	starting from the desired equipment performances and fixed geometrical parameters, the proper operating conditions have to be evaluated.

It is worth mentioning that the "equipment" we are dealing with can be a particular device of the plant (e.g. the brine heater) or even the evaporator itself.

In Fig. 2 an example of this last case is outlined for the three types of problems.

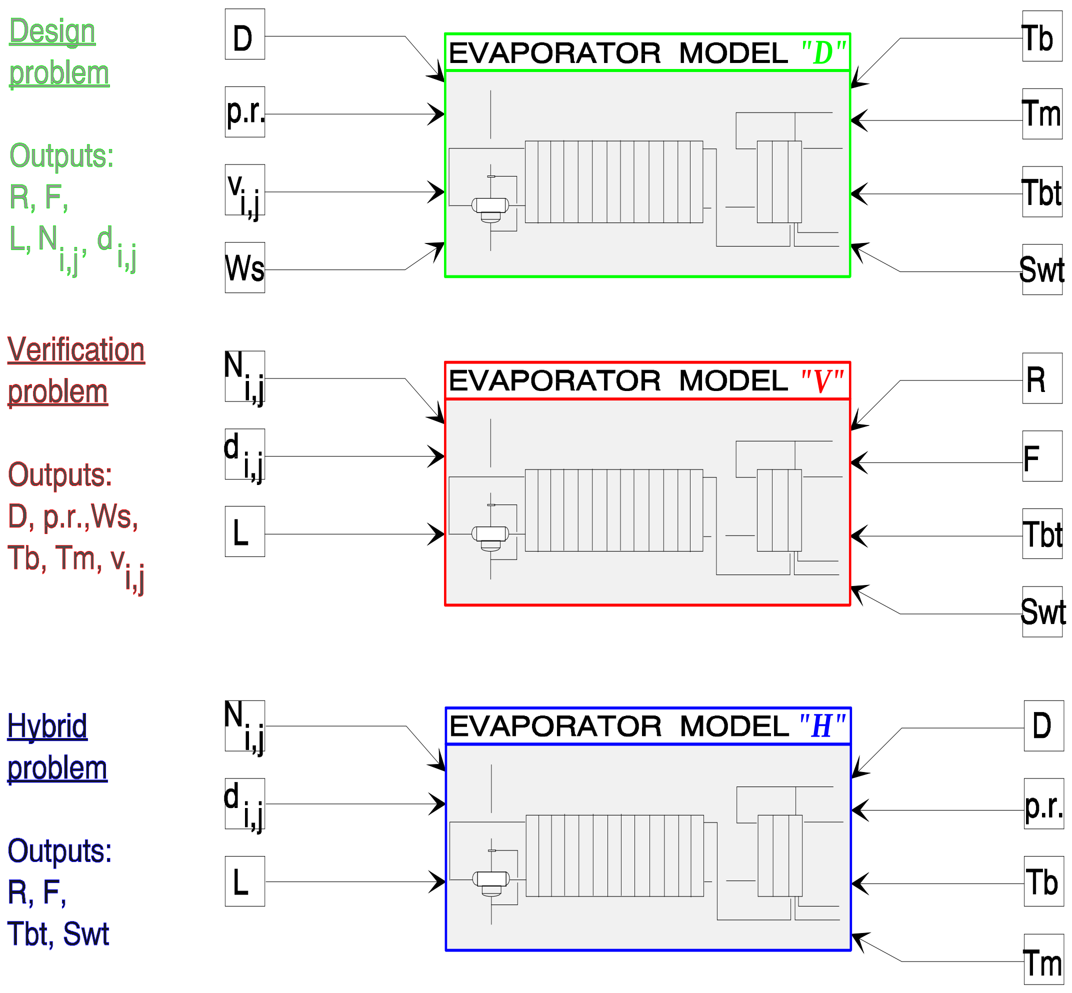


Fig. 2: Solving the three categories of problems with different evaporator models

D	Distillate output	Tbt	Top brine temperature
p.r.	Performance ratio	Swt	Reject inlet temperature
$v_{i,j}$	Tubes velocity	R	Brine recycle flow
Ws	Specific flow	F	Sea water to reject flow
Tb	Blow down temperature	L	Tubes length

Tm	Make up temperature	$N_{i,j}$	Tubes number
(i = recovery, j = reject)		$d_{i,j}$	Tubes diameter

From the above figure, it is clear that the same model cannot be used to solve all the types of problems, but on the other hand it can be seen that inputs and outputs are always the same, only mixed in different ways. Starting from this observation, the idea came out to have a tool linked to the model allowing the user to choose from a global set of variables and deciding which of them are inputs and which of them are outputs.

<u>The solving tool</u>

The basic principles of the proposed software tool are schematically outlined in Fig. 3.

Basically, the available model, whatsoever type it is (design, verification or hybrid), is used iteratively by the solving tool in order to get the desired outputs. The inputs to the model, starting from an initial set, are step by step changed by the solving tool following the best direction to reach the requested answer.

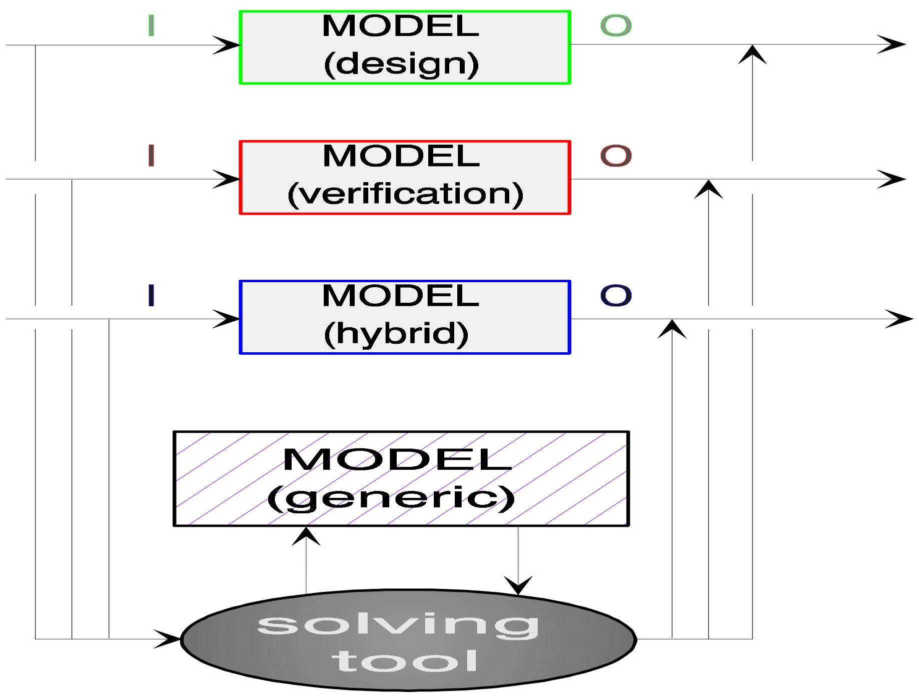


Fig. 3: Solving the three categories of problems with a generic evaporator model

The solution process implements a non-linear programming (NLP) Newton-like method to reach the target set of data acting on some given parameters, subjected to certain constraints, depending on the nature of the problem to solve.

In more detail:

Targets:	are the desired values for the given type of problem. They constitutes the model outputs.
Parameters:	are the changeable values to be modified in order to reach the targets. They constitutes the model inputs.
Constraints:	are the typical values of the parameters used to control the solution process, driving it to feasible values in the surrounding of them.

The basic equation of the solving algorithm is the following:

$$\left(\mathbf{T}-\mathbf{T}_{des}\right)^{\mathrm{T}} \times \mathbf{W}_{\mathrm{T}} \times \left(\mathbf{T}-\mathbf{T}_{des}\right) + \left(\boldsymbol{\Psi}-\boldsymbol{\Psi}_{con}\right)^{\mathrm{T}} \times \mathbf{W}_{\Psi} \times \left(\boldsymbol{\Psi}-\boldsymbol{\Psi}_{con}\right) \Rightarrow min.$$

$$\mathbf{T} = \mathbf{T}_0 + \mathbf{J} \times \left(\boldsymbol{\Psi}-\boldsymbol{\Psi}_0\right)$$

where:

$\mathbf{T}$	is a	n x 1	array of the actual outputs
$\mathbf{T}_{des}$	is a	n x 1	array of the desired outputs (targets)
$\mathbf{W}_{\mathrm{T}}$	is a	n x n	matrix of the targets weights
$\boldsymbol{\Psi}$	is a	m x 1	array of the actual model inputs (parameters)
$\boldsymbol{\Psi}_{con}$	is a	m x 1	array of the typical model inputs (constraints)
$\mathbf{W}_{\Psi}$	is a	m x m	matrix of the parameters weights

and:

$\mathbf{T}_0$	is a	n x 1	array of the outputs at the previous iteration
$\boldsymbol{\Psi}_0$	is a	m x 1	array of the model inputs at the previous iteration
$\mathbf{J}$	is a	n x m	matrix (Jacobian) of derivatives whose structure is:

$$\mathbf{J} = \begin{pmatrix} \partial t_1/\partial\psi_1 & \partial t_1/\partial\psi_2 & \cdot & \partial t_1/\partial\psi_{m-1} & \partial t_1/\partial\psi_m \\ \partial t_2/\partial\psi_1 & \cdot & \cdot & \cdot & \partial t_2/\partial\psi_m \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \partial t_{n-1}/\partial\psi_1 & \cdot & \cdot & \cdot & \partial t_{n-1}/\partial\psi_m \\ \partial t_n/\partial\psi_1 & \partial t_n/\partial\psi_2 & \cdot & \partial t_n/\partial\psi_{m-1} & \partial t_n/\partial\psi_m \end{pmatrix}$$

The presence of constraint values for the parameters is useful not only to keep the solution in a limited feasible domain, but also to avoid non-determination of the problem in the case that the number of parameters is greater than the number of the targets.

A very important role in the solution process is played by weights.

The meaning of weights is different for model outputs (targets) and for model inputs (parameters): for targets the higher the value of the weight, the more the algorithm will try to approach the desired target value; for parameters the higher the value of the weight, the more the algorithm will try to remain close to the typical parameter value (constraint).

A typical structure of a data file for the solving tool is given in Table 1, where `W` stands for weights and `C` stands for constraints.

T A R G E T S

TYPE	VALUE	W
Distillate prod. [t/h]	1900.0	1.0
Performance ratio [-]	8.0	1.0
Rec. tubes vel. [m/s]	2.0	1.0
Rej. tubes vel. [m/s]	2.0	1.0
Specific flow [t/h/m]	1000.0	1.0
Blow down temp. [°C]	40.0	1.0
Make up temp. [°C]	40.5	1.0

P A R A M E T E R S

TYPE	C	W	Δ	MIN	MAX
Top brine temp. [°C]	112.	-1.0	0.5	70.	130.
Brine recycle flow [t/h]	19850.0	0.0	150.	15000.	20000.
S.w. to rej. temp. [°C]	32.0	-1.0	0.5	15.	35.
S.w. to rej. flow [t/h]	17700.0	0.0	150.	15000.	20000.
Tubes length mult. [-]	1.0	0.2	.05	0.5	1.5
Rec. tubes num. mult. [-]	1.0	0.0	.05	0.5	1.5

Rec. tubes diam. mult. [-]	1.0	0.1	.05	0.5	1.5
Rej. tubes num. mult. [-]	1.0	0.0	.05	0.5	1.5
Rej. tubes diam. mult. [-]	1.0	0.1	.05	0.5	1.5

Tab. 1: A typical input data file for the solving tool

The file (main.dat) in this example refers to a case in which a verification evaporator model is used for design purposes. All the outputs weight values are set to 1.0, that means that all the requested targets must be matched exactly. This is in this case possible since the number of parameters that can be changed (weight > = 0.0) is not less than the number of targets to be reached; in fact only the top brine temperature and the sea water temperature are fixed (weight = -1.0). The parameters with a weight value of 0.0 are free to change, whilst the ones with a weight value > 0.0 are more or less constrained to a certain figure, depending on the weight value.

Only for the model input data (parameters), other three values are specified:

Δ	is the parameter increment used in Jacobian derivatives evaluation
MIN	is the minimum allowable value for the parameter
MAX	is the maximum allowable value for the parameter

The constraint values are used also as first trial parameters. The model input data that have not to be changed and other tuning parameters for the solving tool (max. number of iterations, tolerances, etc.) are stored in separate files (dati.dat, param.dat) (see fig. 4). A report output file (tool.o) is generated together with the usual output file of the model itself (model.o).

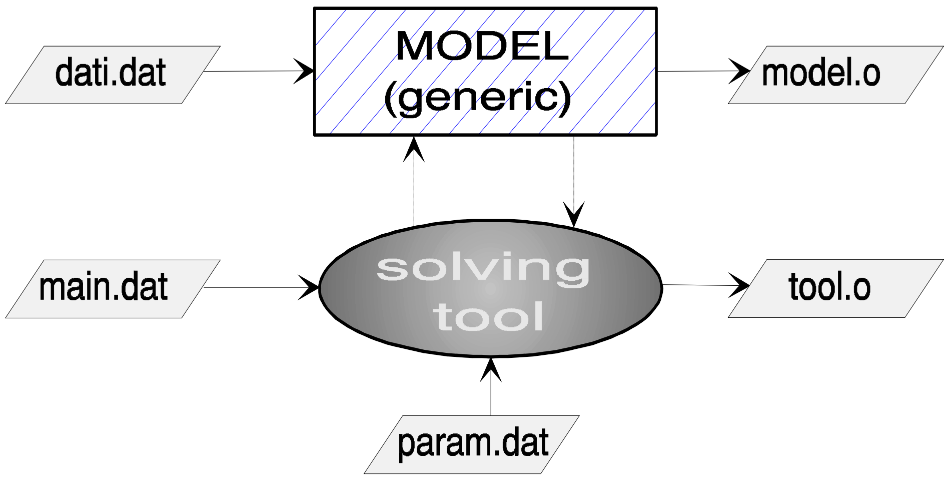


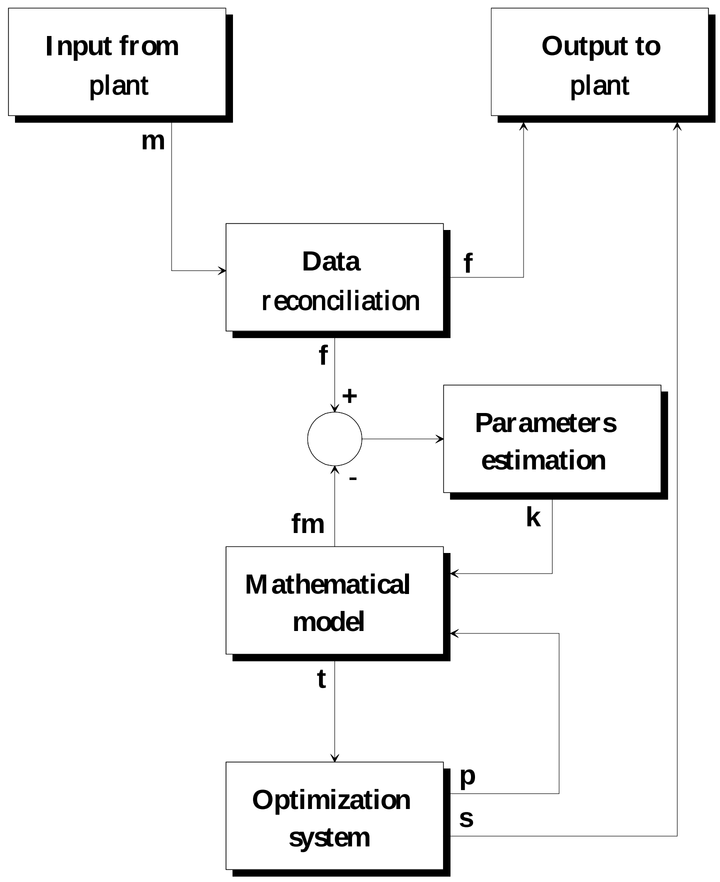
Fig. 4: A simplified software structure for the solving tool

In the above example the operating conditions (flows) and the geometrical parameters (tubes length, tubes diameter and tubes number both for recovery and for reject sections) are calculated by the proposed tool, so solving a design problem by means of a verification model.

If, after the design evaluation, verification run-outs are requested, the same system can be used just setting to -1.0 all the parameters weights (model inputs fixed) and to 0.0 all the targets weights (no target to be reached).

An application to optimize MSF plants operation:

The software package DROPS (**D**ata **R**econciliation and **Op**timization **S**ystem), developed by ITALIMPIANTI, is a good example of how the proposed tool can be applied to a so-called "hybrid" problem, corresponding in this case to an optimization problem.



Legend:	*m*	*measured values*
	f	*best estimates of measurements*
	fm	*true values from model*
	k	*model parameters*
	p	*optimization parameters (trial values of set points)*
	t	*targets*
	s	*set points*

Fig. 5: The functional structure of DROPS

In Fig. 5 the schematic functional structure of DROPS is outlined. This system has been developed in order to minimize the unit cost of the distillate produced by a MSF unit by properly tuning the operating conditions of the distiller (set points). The heart of DROPS, represented in Fig. 5 by the block “**Op**timization **S**ystem”, is essentially constituted by the proposed tool, working in this case on an hybrid problem where:

• Targets:	desired distillate production rate
	minimum specific steam consumption
	minimum specific chemicals consumption
	minimum specific electrical consumption
• Parameters:	top brine temperature
	brine recycle flow
	sea water to reject temperature (winter)
	sea water to reject flow
	sea water make up flow
• Constraints:	top brine temperature allowed values
	recovery tubes brine velocity allowed values
	sea water to reject temperature allowed values
	reject tubes sea water velocity allowed values
	brine recycle salinity allowed values
	process pumps capacity allowed values

Since it is mandatory that the mathematical model used by DROPS in conjunction with the proposed tool is kept constantly aligned with the actual status of the plant, a function of “parameters estimation” (mainly corresponding to fouling factors) is included in the system; this estimation is significant only when based on reliable measurement from the plant, hence the other main issue of DROPS is constituted by an internal function of data treatment represented in Fig. 5 by the block “**D**ata **R**econciliation”, with advanced filtering and error detection features.

Conclusions

Whilst at the moment the package DROPS is about to be released in its final version, after a period of test on the new desalination plant of Al Taweelah “B” in Abu Dhabi, it is worth mentioning that the proposed solving tool, as a commercial package for design and verification problem solution aid, is still living in a prototype stage.

The software prototype runs on a DEC Alpha workstation, under OpenVMS operating system. As a future development, it could be imported on other hardware platforms, such as last generation personal computers (e.g. 486/66 MHz or Pentium/90 MHz), running on WINDOWS systems.

The basic principles of the solving tool will mainly remain the same, but minor modification to it and to the software itself are going on. For instance, for the time being, the proposed tool (written in FORTRAN language), must be tailored for each customer by a preliminary work on software to accept the customer model, since its code needs to be linked together with the model’s one in order to build the executable file. This derives from the necessity of handling the input model as a function or a subroutine in the actual context. Moreover, the tool could be provided together with a specialized GUI (graphical user interface), still not available, in order to make the customization of the case study and the output data analysis more friendly and easier for the final user.

Furthermore, from a more specialized mathematical point of view, possible improvements can be made on the algorithm moving from the fact that generally the Newton-like method used requires less work, but for instance another NLP method, such as the conjugate gradient method or other methods not using derivatives could be worth trying if stepping through the iterations reveals slow progress between successive trial points. Also some more tuning parameters could be added to increase the algorithm flexibility.

Nevertheless, it must be kept that all these considerations will not invalidate the tool’s generality in approaching the depicted problems.