REVISTA MEXICANA DE INGENIERIA QUIMICA Vol. 3 (2004) 161-175 **AMIDIQ**

DESIGNING COMPONENTS FOR DYNAMIC PROCESS SIMULATION

DISEÑO DE COMPONENTES PARA LA SIMULACION DE PROCESOS DINAMICOS

F. Alarcón-Gálvez [1], J. González-Herrera [2], R. Guzmán-Sánchez [3], E. Morales-Manzanares [4], C. Montiel-Maldonado [5] and D. Juárez-Romero [6*]

[1] *Centro Nacional de Investigación y Desarrollo Tecnológico (Cenidet)*
[2] *Instituto Mexicano del Petróleo Av. 100 Metros, México, D.F.*
[3] *ESIQIE, IPN Unidad Profesional "Adolfo López Mateos"*
[4] *ITESM Campus Morelos, Av. Paseo de la Reforma 182-A, Morelos*
[5] *Departamento de Ing. Química, Facultad de Química Cd. Universitaria 04510, UNAM*
[6] *Centro de Investigación en Ingeniería y Ciencias Aplicadas, UAEM, Av. Universidad 1001, Col. Chamilpa, C. P. 62210 Cuernavaca, Morelos, México.*

Abstract

This work shows a prototype system for the construction of dynamic models for simulation of process plant networks with an improved warranty of producing a well posed system of differential algebraic equations. Every model contains a set of specified procedures that are coded in the C programming language, and it is automatically analysed with similar principles to those of the algorithmic differentiation. Moreover, the scheme to connect two pieces of equipment to ensemble the equations and associated iteration matrix is presented. The proof of these concepts is showed through dynamic simulations of a water supply cycle, and a drum-superheater cycle.

Keywords: model building, dynamic model testing, flowsheet composing.

Resumen

Este trabajo presenta un sistema prototipo para la construcción y simulación de modelos dinámicos de redes de plantas de proceso que garantiza la producción de un sistema correctamente planteado de ecuaciones àlgebro-diferenciales. Cada modelo contiene un conjunto de procedimientos específicos codificados en C y es automáticamente analizado con principios similares a los utilizados en la diferenciación algorítmica. También se presenta el mecanismo para el ensamblado de ecuaciones y matrices de iteración asociadas para componentes de equipo. La prueba de estos conceptos se muestra por medio de simulaciones dinámicas de un sistema de aprovisionamiento de agua y un ciclo de supercalentamiento en un tanque.

Palabras clave: construcción de modelos, prueba de modelos dinámicos, generación de diagramas de proceso.

1. Introduction

To support projects for the development of process plant simulators such as those of the Training Operators Centre, CAOI in Tula, Hidalgo, and demonstration projects at the Electrical Research Institute, Mexico (Zanobetti 1989; Resendiz y Nagore, 1994), several academic undertakings have been executed. Solving of hydraulic networks with high interaction required robust nonlinear equation solver. Also when a process unit for dynamic simulation is coupled to another, it can increase, maintain or decrese its stability, thus, we concluded that implicit Differential-Algebraic solvers are needed (Alarcón *et al.,* 2001).

In the present work, we developed tools for the analysis of dynamic models, which are represent by DAE equations, (Molina *et al*, 1999) and will compose a process network The present work is based in two premises aiming to represent a process *by connecting process units, which can be analyzed and solved*:

The first premise is the decompositions of the network suggested by Bär *et al.* (1993), Eq. 1, which allows us to represent a process network by connecting the process units,

*Autor para la correspondencia *E-mail:* djuarez@buzon.uaem.mx
Tel/fax: (55) 73297084

through streams, given a configuration of the existing plant.

$$Process = process - units + streams + configuration \quad (1)$$

The second premise is based on the computational graph used to represent internally a computer code, Eq. 2:

$$Process\ unit = operators + variables + control\ flow \quad (2)$$

This allows us to analyze different characteristics of a process unit following the computational graph. This graph, which is produced while the program is executed, describes the behaviour of variables that are modified by the operators in accordance with the control flow:

From these premises, the development of the paper is as follows: the second section shows a model decomposition based on process units, their characteristics, and their form of analysis. Section three is related to the type of connections used to build process networks and the form of evaluating physical properties. Section fourth deals with the network composition, the solution method, and implementation tools. The fifth section shows results of applying this scheme, and finally conclusions derived from this work are presented.

2. Process units

Some of the main issues related to object oriented modeling are (Holibaugh, 1991): to define representation techniques to asses if a model has a high fidelity behavior, to test if a model is properly formed and to verify its completeness and consistency. To care for these issues, we required the following characteristics of a model:

Operability: To represent the full set of operating regions as the reference unit i.e.: starting up, transition (forward and reverse flow), and malfunctions. This feature is essential to train operators for dealing with failures or unsafe conditions.

Configurability: To allow the same changes in unit sizes and feasible connections as the reference unit.

Observability: To be able to obtain the state of the model from the model outputs. This characteristic allows the validation of the model using the measurements obtained in the reference process unit.

Traceability: To be able to trace different characteristics of the model. For instance, the matrix that relates equations to variables (incidence matrix) used for the detection of arithmetic errors. This requirement also needs that the model's code is clearly written with expressive and standard language features.

2.1 Model characteristics

We use three views to describe a model:

- Phenomenological view
- Mathematical view
- Computational view

Each view will be discussed in the following subsections

2.1.1 Phenomenological view

This view represents the model as the actual unit works. Consider the working regions of a drum for steam generation, Fig. 1:

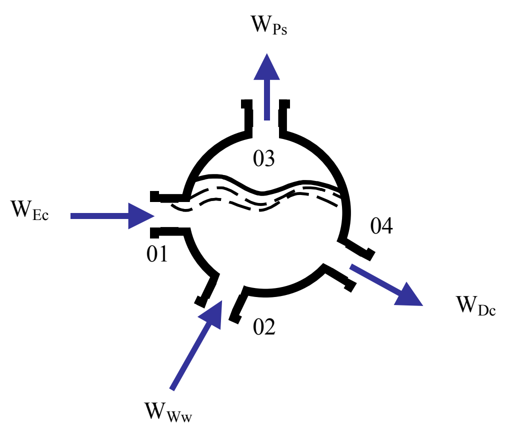


Fig. 1. Drum model.

- During start up: Filling, heating, emptying.
- During transition: Evaporating, steam generation, steam released to atmosphere
- During malfunctions: Leaking due to couple rupture, loosing pressure due to a down-stream failure, changing steam pressure gradient due to up-stream oscillations.

The outcome is that a model, which represents all these regions, must contain conditional clauses to delimit the equations according to each region. Some of these conditional clauses can be nested.

Balance equations for this model can be obtained applying Bernoulli's Equation for two phases:

$$\frac{dM_l + dM_v}{dt} - \sum W_i = r \qquad \text{Mass (3)}$$

$$P_{in} - P_{out} + g \sum \rho \Delta Z = r \qquad \text{Momentum (4)}$$

$$\frac{dH_l M_l + dH_v M_v}{dt} - \sum H_i W_i = r \qquad \text{Energy (5)}$$

Where:

M = Mass	H = Enthalpy
ρ = Density	g = Gravity
P = Pressure	W = Mass flow
ΔZ = Height	R = Residuals vector

We represent balance equations as residuals to allow a wide range of operation conditions without assuming a specific causality. The terms of these equations are evaluated according to the working regions.

2.1.2. Mathematical view

This view provides the required functionality for the model. For this purpose we use the following classification of equations (Ponton and Gawtrhop, 1991):

Balance equations. These represent changes in mass, momentum and energy.

Transfer mechanism equations. These define the rate of transfer into, out of, or between phases.

Constitutive relationships: These equations relate intensive variables in terms of extensive variables.

Constraints: These expressions limit the application of the equations (maximum or minimum flows), or verify that adequate values are used as arguments of arithmetic expressions.

The model equations representing the behavior of a unit can be grouped in a mathematical form as:

$$V = a(u, x, y, p)$$

Intermediate equations (6)

$$A(y)\,\frac{dy(t)}{dt} - f(t, u, v, x, y) = r_d$$

Differential equations (7)

$$G(t, x, y, u) = r_a$$

Algebraic equations (8)

$$Z = h(p, u, v, x, y)$$

Signal equations (9)

Where:

p ≡ unit parameters, which do not vary with time.

u(t) ≡ external continuous variables.

v(t) ≡ intermediate variables that are used to store the value of some computations.

x(t) ≡ algebraic state variables that do not have accumulation, but change with time.

y(t) ≡ differential state variables, which have accumulation, or time derivatives

z ≡ signals, variables whose value can be displayed in panels or registers.

r_a ≡ algebraic residuals.

r_d ≡ differential residuals.

a, f, g, h are real valued functions.

A(y) is a nonsingular real matrix.

The *procedural* (explicit) form of intermediate equations allows efficiency in the calculation, while the *declarative* form of the balance equations (the equation presents a relationship between the variables) allows the flexibility required for the specified variables.

2.1.3 Computational view

The computational view is implemented by coding the equations in a set of procedures. To increase understanding of the code among users, and to detect possible coding errors we use a common model description. This is composed by procedures and variables with a specific naming.

2.1.4 Naming of model's procedures

The user writes the model in standard C language with the name of the unit as its file name **`EeEquipo.c`** and header **`EeEquipo.h`**. (Ee is replaced with two literals representing the referred unit). As we shall see ahead, once the model is successfully tested it is automatically converted in a C++ model class. Every process unit is built within a set of procedures that contain the different types of equations, coded by the developer with predefined names, Table 1.

Table 1. Standard procedures in a model.

Procedure Name	Purpose
Scale()	Defines geometry and capacity parameters, **p**.
Start (Time)	Sets initial, **x***, and starting conditions, $\mathbf{y}^0$.
Behave (Time, EeR[])	Evaluates the intermediate variables, and evaluates the residuals of the conservation equations, $\mathbf{r}_a$, $\mathbf{r}_d$.
Operate (Time)	Modifies external variables, **u**.
Signal (Time)	Evaluates signal variables, **z**.

2.1.5 Naming of types of model's variables

The variables allowed are of basic type (integers, floats) or arrays. No composite types like structures are allowed. State variables, inputs, parameters and signals are global variables.

The intermediate variables are local, since they are only used in the model. Other variables are global. The use of global variables allows using the same form of invoking different type of models. The user can also code other procedures required for the model.

The following nomenclature that defines the different types of variables is adopted:

a *global variable name* is composed by 8 characters with a general form Ee##ΦΦ φT.

```
Tk 01 Ht V A
 |   |   |  |  |_____Type     : Algebraic
 |   |   |  |________Phase    : Vapor
 |   |   |___________Property : Enthalpy
 |   |_______________Port     : 01
 |___________________Unit     : Tank
```

a *local variable name* is composed by six characters since it does not have the unit identifier.

2.2 Physical properties

The requirements of the functions constructed to evaluate physical properties of fluids are:

- *Completeness*. To cover all the range of working regions.
- *Explicitness*. To be explicit in the required physical properties to avoid inner iterations that could deteriorate the global iteration cycle.
- *Functionality*. To obtain properties and their first derivatives by continuous expressions, since some expressions in the model might also require derivatives. Newton method requires continuity in the derivatives to maintain the direction of convergence.

- *Accuracy*. To compute the values of physical properties to fulfil the required precision.

To fulfil these requirements, we approximate fluid properties by piecewise-continuous polynomials in terms of pressure and enthalpy. Given the value of the state variables, a model evaluates its required physical properties. For the saturated phase:

$$\varphi_s = \varphi_s(\sqrt{P}) \tag{10}$$

Otherwise: $$\varphi = \varphi(\sqrt{P}, H) \tag{11}$$

First derivatives with respect to the state variables are also evaluated. First derivatives are evaluated as the derivatives of these polynomials. These derivatives have only slight discontinuities in junctions between segments of every polynomial, which do not affect the iteration process.

2.3 Model analysis

Model analysis is necessary to obtain its algorithmic properties so it can reproduce high-fidelity behaviour. After a given formulation was defined, a model is coded and analyzed.

Our scheme for model-analysis is based on the schemes used by Grienwank (2000) for algorithmic differentiation using operator overloading. The basic principle relays on the *ensemble of the terms of an expression using the same computational graph to form the final result* (Molina *et al., 1999*), similar to the chain rule. When this model-analysis is executed, all possible branches, cycles and procedures of the code are traced. Tolsma and Barton (1998) based in their computational experiments, concluded that algorithmic differentiation has significant advantages over finite differences, symbolic differentiation and hand coding differentiation in terms of speed and memory usage.

After the formulation was defined, models were coded and analysed to obtain:
- *Assignment & reference*. This analysis detects uninitialized global variables, unassigned intermediate variables, and un-referenced local variables that can be used to evaluate the balances. The correct assignation and reference is required to avoid difficulties with delay of information caused by improper equation sequence.
- *Structure & states*. This analysis obtains the incidence matrix of the model. Incidence matrix has a row for every equation and a column for every variable. If a variable j appears in equation i, the position i,j of this matrix is marked (Mah, 1990). With this matrix it is possible to detect block of equations that could cause structural or numerical difficulties.

Costa, Griño and Basañez (1998) generated models with Differential-Algebraic equation where its equations are differentiated symbolically using Maple to produced a Jacobian, and is solved numerically using DASPK, Which is a solver with variable order, variable step-size with the backward differentiation formula (Brenan *et al,*, 1996).

However, models cannot be differentiated symbolically when they contain conditional clauses that bound the working regions; or cycles, which evaluate several compounds in a mixture.

Example 1

Starting for a particular tank model equations:

```
/* Mass balance */
TnR[0] = Tn00MaLD - Tn01WmLA +
Tn02WmLA - Tn03WmLA;
TnR[1] = Tn02PaLA - Am20PaLA -
(9.81 * Tn00MaLE / Tn00Ar_P) +
(Tn02Pd_P * fabs(Tn02WmLA) *
Tn02WmLA) - Tn02Pa_P;
/* Momentum balances */
TnR[2] = Tn03PaLA - Am20PaLA +
(Tn03Pd_P * fabs(Tn03WmLA) *
Tn03WmLA) - Tn03Pa_P;
```

The incidence matrix obtained for this model is:

Residual / Variable	Tn00 MaE or Tn00 MaD	Tn01 WmLA	Tn02 WmLA	Tn02 PaLA	Tn03 PaLA	Tn03 WmLA
TnR[0]	X	X	X	0	0	X
TnR[1]	X	0	X	X	0	0
TnR[2]	0	0	0	0	X	X

- *Efficiency & robustness.* This analysis detects vulnerable arithmetic expressions. Since models represented as a set of DAEs offer more flexibility than models with only algebraic (resistance) or with only differential (accumulation) equations, but at the same time they are more fallible. It is also required to evaluate the computing effort requirements.

2.4 Model packing

Once a model is successfully analyzed, (i.e. it does not have any type of severe diagnoses), it is packed. Colhun and Lewandowski (1994) use classes to store the state variables, and ports in the developments of dynamic models.

Model packing binds all model variables and procedures in a C++ class. It increases cohesion; it also allows incorporating several units of the same type in a process network. A preprocessor builds the following data structures and functions automatically, Tabla 2.

Table 2. Data structures and procedures generated for model packing.

Data structure	Purpose
Class Equipment	Contains the procedures and the ports required to describe the mathematical model.
Struct Port	Contains the variables, which are transported from one unit to another. One unit has at least one *port.* Ports correspond to physical connection through which exchange takes place.
Struct State	Contains the variables that define the state of the process unit, as well as its maximum and minimum value allowed.
Procedure	Purpose
Incide()	Assigns the number of states, and number of balances of every process unit
Couple(Port01, Port02)	Assigns the relationships between the list of ports, which can be connected in a unit, and model's state variables.

3. Connectors

3.1 Streams

Stream is a set of variables that communicate a process unit with another without any accumulation. The number and type of each variable characterize a stream, Table 3.

Stream variable convey state variables. Intermediate variables required in a model are reevaluated internally.

Table 3. Type of streams.

Stream	Air - Gas	Hidraulic	Thermic
Component	Wm, Pa, Ht	Wm, Pa, Ht	Ql, Tm

3.2 Connection characteristics

In this section we also present the characteristics of connections as phenomenological, mathematical and computational view.

3.2.1 Phenomenological view

Physical connection is established between ports, Fig 2.

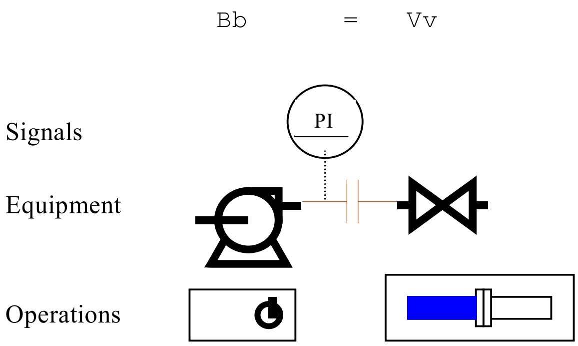


Fig. 2. Coupling two units by a common stream.

3.2.2. Mathematical view

Equality is established for the elements of a stream. When stream S_i is connected with stream S_j, then $S_{i,k} = S_{j,k}$ $\forall$ elements in the connecting stream, k.

3.2.3 Computational view

A pair of connected streams shares the same space in memory. Fig. 2 shows the mechanism used to establish model connections. When Bb is connected to Vv the residuals of the network are formed concatenating both sets of residuals.

Matrix of partial derivatives for composed Pump-valve (Jacobian):

	Bb01 WmLA	Bb01 PaLA	Bb02 WmLA	Bb02 PaLA	Vv01 PaLA	Vv02 PaLA
R[0]	1	0	-1	01	0	0
R[1]	0	-1	-dBb1	1	0	0
R[2]	0	0	1	0	-1	0
R[3]	0	0	0	- Vv00 Ar_P	dVv1	Vv00 Ar_P

where

dBb1 = Bb00Pd2P + Bb00Pd3P*(fabs(Bb02WmLA) + Bb02WmLA*sign(Bb02WmLA))

dVv1 = Vv00Pd_P*(fabs(Vv02WmLA) + Vv02WmLA)*sign(Vv02WmLA))

3.3 Connection Analysis

A connection between two ports can be established if:

1. Connecting ports do not belong to the same process unit.
2. Both ports have the same type.
3. They have complementary direction of connection (input with output) or one of them is an in/output port. Figs. 3 to 5 show different types of connections:

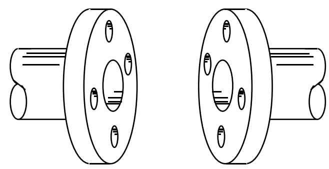


Fig. 3. Indistinct Ports.

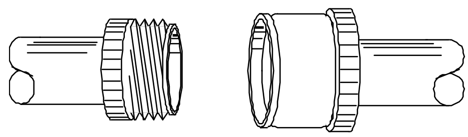


Fig. 4. Complementary ports. Feasible.

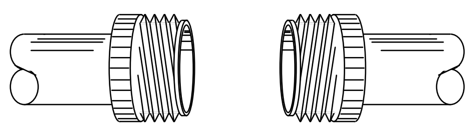


Fig. 5. Unfeasible connection.

Figs.3 and 4 present feasible connections. In Fig. 5 an infeasible connection is presented, since both have the same connecting direction.

4. Network

4.1 Network specification

A process model based on physical principles implies the specification of balance equations. When the residual form of balance equations (as in Eqns 7, 8) is used the following advantages are obtained:

· They fully describe the state of a system in relation to its material properties and its geometry (Gerstlauer, *et al*., 1994).
· They represent accurately the global behavior without assuming any causality or strength of interaction between neighboring units (Cellier, Hilding, Elquist, 1993).
· They are additive. The total residual vector can be formed by sequential aggregation of the residuals of every model, according to the specific network configuration.

Hence, the user composes the process network by a set of process units that are connected by ports transporting streams. A process network is built specifying the connection among units analogous to an electric wiring or to a process piping.

The network is defined in a unique form by:
1. Specifying the components
2. Specifying the connections
3. Specifying the initial and operating conditions.

We shall explain every stage in the following subsections.

4.1.1 Specifying components

The user selects the elements used to compose a network from a process unit "menu" answering Yes/No. This "menu" is located in the file **`RedEqp.eqp`**. The first column contains the "Unit key".

The sizes and capacities of every unit is specified in file **`EeEquipo.dsg`**

4.1.2 Specifying connections

Once the user has selected the units that conform the network, he specifies the connection among units. The connection is specified by indicating origin and destination ports. This information is stored in file redeqp.cnx.

4.1.3 Specifying conditions

4.1.3.1 Initial conditions

The user specifies initial conditions of differential variables, and starting value of algebraic variables in file RedEqp.ini.

The numerical method produces new approximation of the state variables within the feasible interval: ValMin ≤ Val ≤ ValMax.

Variable	Valor	Derivada	Conocida	ValMin	ValMax
To00MaLE	0.0	0.0	s	0.	100.0

Table 4. Specification of initial conditions.

Variable	Variable name
Valor	Specified value for known variable; starting value for algebraic variable; or initial value for differential variable.
Derivada	Derivative value for differential variable
Conocida	*s* if variable is known, *n* if variable is unknown (differential or algebraic)
ValMin	Minimum allowed value
ValMax	Maximum allowed value

4.1.3.2 Operating conditions

The user specifies time dependent changes of operating variables. These changes are defined in file **`RedEqp.opr`**

Table 5. Specification of operating conditions.

Variable:	Variable name
ValDif:	Variable increment
TiemRef:	Reference time
TiemDif:	Time increment

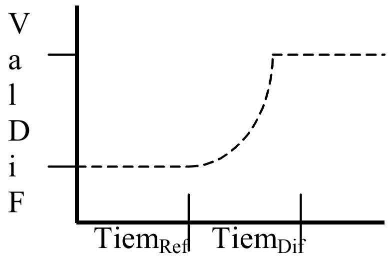
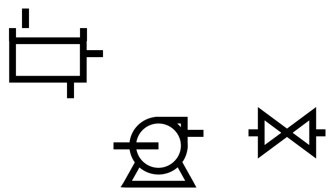


Fig. 6. Specification of operation.

Modo: Form of increment: (L) linear, (C) quadratic, (S) sinoidal, (E) exponential.

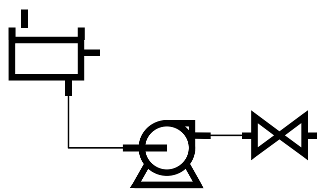
Example 2

1. The following units were selected:

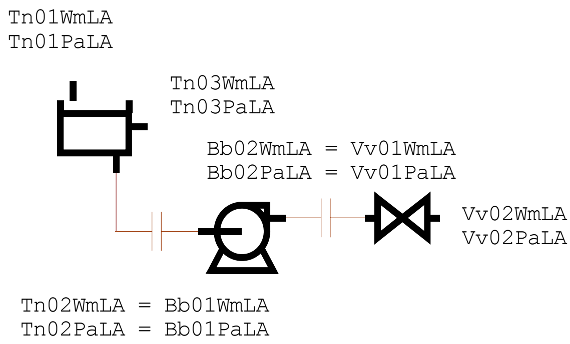


With their capacities:
Tank 1000 lt, Pump ½ hp, Valve ¼ in

2. The following connections between units are specified:



3. The following conditions are defined: Tank Mass, Tank Input flow, Pump angular speed, Valve position.



Network built

Number of units = 3
Number of connections = 2

Once the network is specified and operation condition of pump and valve is specified, the network is analysed.

4.2 Network analysis

4.2.1 Degrees of freedom

A table of global variables in the network is formed according to their mathematical type. The degrees of freedom are the number of variables that can be specified in the network. From these variables we can calculate the others by solving the global system of equations. The number of degrees of freedom is evaluated then as:

$$ {}^{o}F = \sum_{i=1}^{n} {}^{o}f_i - \sum_{j=1}^{m} c_j \qquad (12) $$

where
c_j = Number of variables in connection *j*
^{o}F = Number of degrees of Freedom for the network
^{o}f = Number of degrees of Freedom for unit *i*
n = Number of units
m = Number of connections

We compare the number of degrees of freedom with the number of specified variables. If they agree, the system is properly posed; otherwise, it is necessary to review the number of specified variables.

4.3 Network initialization

Once a network is configured, to start a simulation the user provides: a) initial condition y_0 and b) starting values of algebraic variables x_*, (Table 4). Initial conditions can also be given as a system of non-linear equations, for instance the mass contained in a two phase drum must satisfy $V_T = M_l/\rho l + M_v/\rho_v$. Initial conditions have a general form I(x, u, y, dy/dt) = 0. The DASPK solver provides facilities to handle these types of conditions.

4.4 Network solution

The unknown state variables are solved globally, whether they contain recycles or not. Solution is achieved varying the value of state variables to achieve that $||r|| \rightarrow 0$ (Eqns 5, 6)
Once the total balances are solved (their value is close to zero), the signal equations are evaluated for every unit.

4.5 Network singularities

Due to the linearity on the derivatives of the balance equations, a model is well posed if the matrix A(y) is not singular (Lefkopoulos and Stadtherr, 1993).

4.6 Implementation tools

For model development and analysis C++ Compilers from Borland and Microsoft were used. To improve the code and to test the model-analysis tools, Codewizard and C++Test from Parasoft were used.

5. Results

In this section we employed all the concepts previously described in the sections 2 to 4.

A hydraulic network and steam generation networks are built and simulated to analyze the dynamic behavior of the variables during the coupling of two or more equipment with different operating conditions, and to observe the characteristics that the numerical method presents to solve the equipment's network.

5.1 Hydraulic network

The adequate operation is necessary to avoid reverse flow in this process.
The configuration of this network is shown in Fig. 7.

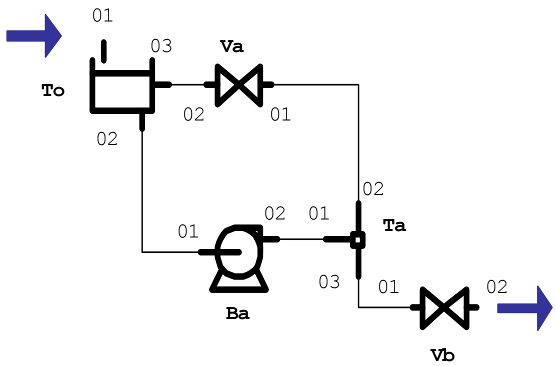


Fig. 7. Schematic diagram of water supply.

Operating conditions: After the pump has started up, the recirculation valve Va is closed while the feeding valve Vb is opened.

Fig. 8 presents the changes in input variables, Fig. 9 and 10 shows the results. When valve Vb is open from time 20 to 40 we observe that mass in tank **To** diminishes slowly, and mass flow in pump **Ba** increases. Mass flow in valve **Vb** increases, while mass flow in valve **Va** decreases slowly.

When the recirculation valve **Va** is reduced from time 60 to 80 we observe that mass in tank continues its downward trend while mass flow of valve **Va** diminishes to zero, mass flow in pump **Bb** diminishes.

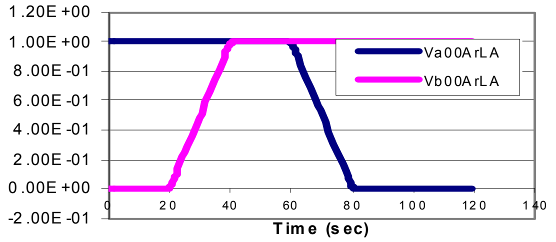


Fig. 8. Operating conditions. Valve areas A/Atot)..

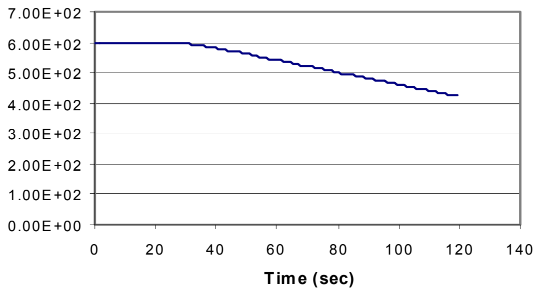


Fig. 9. Mass n tank To (Kg)

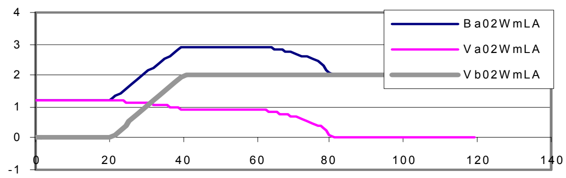


Fig. 10. Mass flow (Kg/s)

5.2 Steam generation network

The cycle drum-superheater of a steam generator unit is integrated by the models, Fig.11: drum (Do), downcomer (Tb), pump (Bb), waterwall (Ww) and superheater (SP).

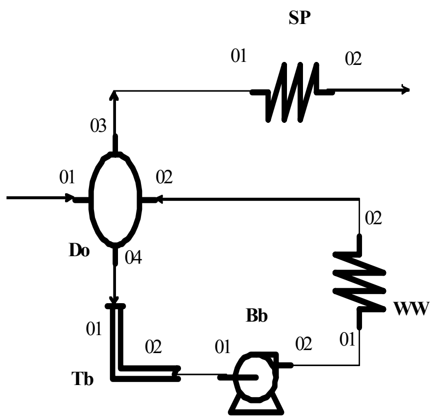


Fig. 11. Cycle drum-superheater.

5.2.1 Steady state test

Operating conditions: The steady state of the network variables during a time interval of 100 seconds from 75% of load.

Result of test. Table 6, describes the steady state values.

Table 6.-Variables in steady stable at 75% of load.

Downcomer	Pump	WaterWall	Drum	Superheater
Pressure 2,611.94	Output Pressure 2,649.00	Pressure 2,585.02	Pressure 2,585.02	Pressure 2,540.59
Liquid Enthalpy 705.57	Liquid Enthalpy 704.60	Liquid Enthalpy 8,115.03	Steam Enthalpy 1,082.1	Steam Density 3.98
Liquid Flow 1,775.49	Liquid Flow 1,775.49	Liquid Flow 1,775.49	Steam Flow 377.84	Gas Pressure 11.54
	Angular Velocity 233.97	Metal Temperature 1,140.10	Liquid Flow 1,775.49	Gas Flow 486.87
			Liquid Volume 340.43	Gas Temperature 1,442.36

From steady state the following variables are modified:

- The values of the pressure and enthalpy of the waterwall are the operating conditions; therefore these variables are fixed at their desired value.
- The waterwall pressure (Ww02PaLA) and drum pressure (Do02PaLA) have the same value.
- The mass flow and enthalpy variables of the drum have initial conditions, their values are modified during the simulation, to achieve the steady state value

5.2.2 Steady state behavior of the network

Time = [s]
Density = [lb/ft^3]
Volume = [ft^3]
Pressure = [lb/in^2]
Enthalpy = [Btu/lb]
Flow = [lb/s]
Angular speed = [rad/s]
Temperature = [R]

5.2.3 Dynamic test of the network

Operating conditions. This test simulates the steam flow at output of superheater to environment. This simulation is carried out by reducing the pressure of the superheater. The steps for this test are the following:

1.- The simulation is done at 75% load in steady state during 100 seconds.
2.- Decrease 50% the values of the superheater pressure at time t=100 during 2500 seconds.
3.- The simulation ran during 3000 seconds.

Results of tests. The decrease of the superheater pressure (Sp02PaVA) reduces steam pressure (Do00PaVS), steam density (Do00RoVE), and liquid enthalpy (Do00HtLS) of drum (Fig. 12), and increases liquid density (Do00RoLS), and liquid volume (Do00VVLE) of same equipment. The enthalpy of waterwalls (Ww02HtLA) and recirculating pumps (Bb02HtLA) is also increased since the recirculating flow is reduced.

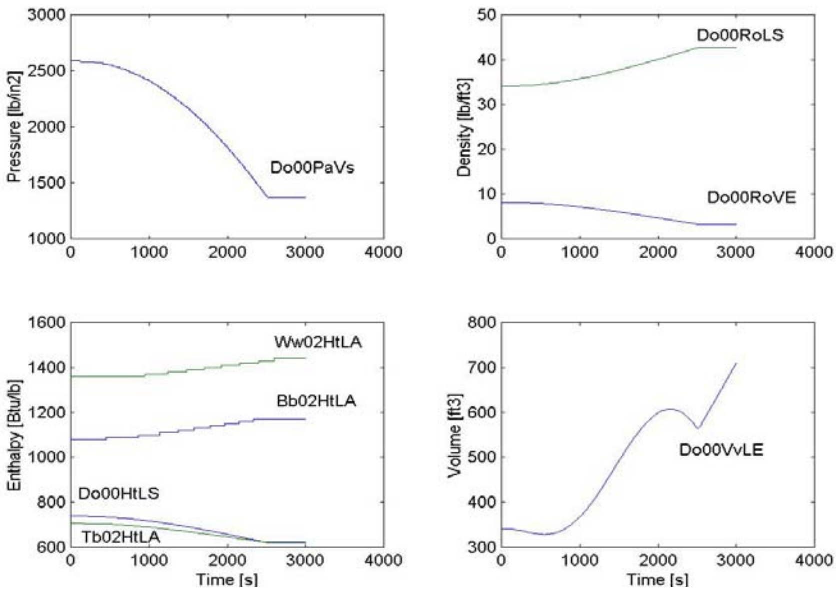


Fig. 12. Dynamic behaviour of pressure, enthalpy, density and volume for cycle drum-superheater.

The decrease in output pressure of superheater increases output steam flow from drum, until the output pressure is readjusted, Fig. 13.

Statistics of execution. The number of function residual evaluations per time step is shown in Fig. 14. In this figure we observe that DAE solver takes between 10 and 19 residual evaluations, which includes the evaluation of the numerical Jacobian per time step. In this figure also appears the approximation order of the integrator, which is between first and second order.

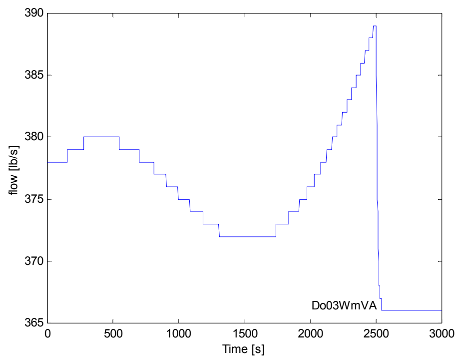


Fig. 13. Dynamic behaviour of flow for cycle drum-superheater.

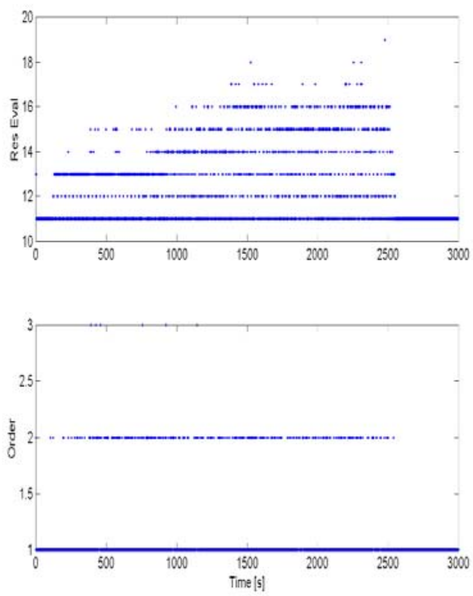


Fig. 14. Statistics of execution of dynamic transient.

Conclusions and related work

This work pursued the production of computer models with characteristics of operability, observability, configurability, and traceability as discussed in section 2.

Having specified some standard procedures of every model, the development relies strongly in the advance of compilers for the organization of these procedures, and in operator overloading for the automatic analysis of their expressions.

This scheme allow the agility of team development, but reduces inconsistencies emerged during model composing. Additional work is required in the analysis of process network, and in the conditions that guarantee a well posed problem.

Remaining work

The process network is formed by coupling a set of models, which were tested individually. Therefore the main issues are:

Analysis of specified conditions

To analyse the specification of free variables in a system of *algebraic* equations, the main criteria is to assign output variables to equations. Incidence matrix can be used to detect the proper specification in equation systems and free variables. (Morton, and Collingwood, 1998).

To analyse the specification of free variables in a system of *Algebraic - Differential* equations, the model equations are linearized.

$$A\frac{dy}{dt} - By = 0$$

Then, after Laplace transformation: $A(s)sY - B(s)Y = 0$. A proper assignment of free variables can be specified if and only if *the degree of the polynomial det (sA – B) is equal to the rank of B.* (Soetjahjo, Go, Bosgra, 1998).

Model discontinuities

Two possible types of discontinuities can be present during the execution of a model. *Explicit discontinuities* are expressed in terms of the input values, u (t). In this case the discontinuities can be predicted; *implicit discontinues* are expressed in terms of the state variables, x(t), y(t). Discontinuities can also be *reversible* or *irreversible*. Reversible discontinuities allow the model to return to the previous state when the variable(s), which caused the discontinuity, is moved back. Irreversible discontinuities (for instance a pipe rupture), do not allow that the model returns to the previous state. Irreversible discontinuities are therefore difficult to treat. Barton and Pantelides (1994) recommend to lock the same time step, $t + \Delta t$, and condition to cross the discontinuity. After the time step is successfully taken, it is necessary to locate the time of the discontinuity, $t \leq t_D \leq t + \Delta t$, then to jump through the discontinuity with the new condition and to restart.

Related work

Here we discuss some developments of available environments suitable for dynamic simulators, the general overview appears in Table 7:

gProms has a language description is based on the concept:

Process = Process unit + Model + Tasks.

This facility allows a excellent model operatility.

Ascend. Allows detection of dimensional consistency in the models. The solver offer a detail information about sparse characteristics during the solution.

Modelica was designed to model, simulate and optimize or control physical systems.

ICAS includes a model generator through DAEs, ODEs, Aes or a combination of them, (which are solved as residuals), functions constraints, a simulator for dynamic and steady state and toolboxes for physical properties, sinthesys, optimization, and control.

Table 7. Comparison of dynamic simulators.

Computing Environment	Semantics	Discrete	Spatial Profiles	Operating	DA Solver	Links to other environments
gProms www.ps.ac.ic.uk/gPROMS	Declarative	X	X	Parallel, conditional	Numeric-Symbolic	Fluent
Ascend www-2.cs.cmu.edu/~ascend/	imperative	X		Conditional	Numeric-Symbolic	
Modelica www.modelica.org,	Imperative	X		Conditional	BDF	Simulink
ICAS http://www.capec.kt.dtu.dk	Imperative	X		Conditional	BDF	Open

Acknowledgements

Several of the computer tools used here were acquired through the Red Nacional de Investigación en Informática. Authors received a scholarship from CONACYT, and from REDII to complete this work; www.mor.itesm.mx/~emorales/

The observations made by reviewers helped to present a neater work.

Authors would like to acknowledge the support of the demonstration project "Tecnicas Avanzadas para Simulación en Tiempo Real", of CFE, and the support of the PROMEP grant of SEP, both promoted by Dr D. Resendiz.

References

Alarcón, F., Mendoza, Y., Molina, J. M., Suárez, J. M., Rodriguez, G., Ojeda, R., Treviño, J., Morales, J. L. and Juárez, D. (2001). *Programas para la Construcción Sistemática de Redes de Equipos.* Technical Report CIICAp-UAEM.

Brenan, K. E., Cambell, S. L. and Petzold, L. R. (1996). Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. *SIAM*. Philadelphia.

Bär M., Schaffner, J., Selg, W. and Zeitz, M. (1993). Functionality and Implementation of a Knowledge-Based Flowsheet-Oriented user Inerfase for the Dynamic Process Simulator DIVA. *Simulation 61*, 117-123.

Barton, P.I. and Pantelides, C.C. (1994). Modeling of combined discrete/continuous processes. *AIChE Journal 40*, 966-979.

Calhoun, D. and Lewandowski, A. (1994). Design C++ Classes for Structured Modeling y Sensitivity Analysis of Dynamical Systems. In *Proceeding of the 2nd Annual OON-SKI'94.* Sunriver, Oregon Apr 24-27, 1-15.

Cellier F. E. and Elqvist, H. (1993). Automated Formula Manipulation Supports Objet-Oriented Continuous-System Modeling. *IEEE Control Systems*, 28-38.

Costa, R., Griño, R. and Basañez, L. (1998). DAE Methods in Constrained Robotics System Simulation. *Computación y Sistemas.* Jan-Mar,145-160

Griewank, A. (2000). *Evaluating Derivatives Principles and Techniques of Algorithmic Differentiation,* SIAM Philadelphia.

Holibaugh R. (1991). Object Oriented Modelling" Workshop Addendum to the *OOPSLA Proceedings.* Arizona 73-77.

Law A. M. and Kelton, W. D. (1992). *Simulation Modeling and Analysis*. 2nd Ed. Mc Graw Hill.

Lefkopoulos, A. and Stadtherr, M. A. (1993). Index Analysis for the Unsteady-State Chemical Process Systems-I. An algorithm for Problem Formulation. *Computers in Chemical Engineering. 17*, 399-413.

Mah, R. (1990). *Chemical Process Structures and Information Flows*. Butterworths, Oxford.

Molina, J. M., Zamora, J. R., Mendoza, Y. and Juárez, D. (1999). Analysis of computational models by operator overloading. *Memorias Congreso Internacional de Computación*, IPN, Nov, 202-206.

Morton, W. and Collingwood, C. (1998). An equation analizer for process models. *Chemical Engineering 22*, 571-585.

Ponton, J. M. and Gawthrop, P. J. (1991). Systematic Construction of Dynamic Models for Phase Equilibrium Processes. *Computers in Chemical Engineering, 15,* 803-808.

Resendiz, D. and Nagore, G. (1994). Proyectos de Demostración, Esenciales para el Sector Eléctrico. Bol IIE, Ene/Feb, 9-10.

Soetjahjo, J., Go, Y. G. and Bosgra, O. O. (1998). Diag -a structural diagnose tool for interconnection assignment in model building and re-use. *Computers in Chemical Engineering 22* Suppl., 933-936.

Tolsma, J. E. and Barton, P. I. (1998). On Computational Differentiation, *Computers in Chemical Engineering.. 22,* 475-490.

Zamora, J. R. *Desarrollo e Implantación de Técnicas Simbólicas y Numéricas para Evaluación de Matrices de Iteración".* BSc Thesis ENEP ACATLAN, UNAM.

Zanobetti D. (1989). The Simulators in Mexico *Power Station Simulators*. Commission of the European Communities 173-174.