Proceedings of the 5th WSEAS Int. Conf. on APPLIED INFORMATICS and COMMUNICATIONS, Malta, September 15-17, 2005 (pp163-170)

ControlAvH Tune: Real-time software environment for PID, H2 and H∞ controller design and implementation

JOSE LORENZO, MANUEL J. LOPEZ, LUIS GARCIA
Dpto. Ingeniería de Sistemas y Automática
Escuela Superior de Ingeniería. Universidad de Cádiz
11002 Cádiz
SPAIN

Abstract: - In this paper the architecture and functionalities of ControlAvH Tune software application is described. ControlAvH Tune has been developed for data acquisition, system identification, controller design and evaluation, applied to processes control in real time. It implements identification methods for obtaining mathematical models of the process to control, which can be used for PID, H2 and H_∞ controller design and real time controller implementation. ControlAvH Tune has communication interface for process control through National Instruments data acquisition devices, RS232, NetDDE and OPC technologies. SISO (single-input and single-output) and MIMO (multiple-inputs and multiple-outputs) processes can be identified, controlled and evaluated using performance and robustness indicators. A friendly user interface enormously facilities monitoring, design and analysis tasks. The application is based on an optimized software using object oriented programming (OOP) technique.

Key-Words: real time, optimal control, PID control, H2 control, H_∞ control, user interface, controller tuning.

1 Introduction

One of the main desired characteristics of a controller tuning method is that operator tasks are few and easy to carry out. This property is satisfied by the well-known PID industrial controller, where auto-tuning function implements the controller pre-tuning following a automated procedure, and controller fine-tuning work is made by operator. In process control context, this property must be taken into account for any controller design technology evaluation, such as GPC (Generalised Predictive Control) [1], H2 or H_∞ robust control [2].

Control engineers community generally uses different software packages and tools in order to accomplish the design cycle of a control system. Examples of commercial applications for PID controller design and process control are Expertune [3], Protuner [4] and Intune [5]. As examples of more general tools are Simulink and Matlab toolboxes for identification and control [6].

In this paper, we present the software architecture of ControlAvH Tune, which has been developed by GAPSIS group at Cádiz University. ControlAvH Tune is an integrated software environment for data acquisition, process identification, design, analysis and real time implementation of control systems. As innovative element ControlAvH Tune has the possibility of designing H2 and H_∞ controllers as well as classical PID control, for SISO (single-input single-output) and MIMO (multiple-input and multiple-output) industrial processes. Data acquisition serves as I/O interface and identification techniques are implemented to obtain useful process models for controller design and validation.

The rest of the paper is organized as follows: In section 2 design application to be implemented is described. Section 3, 4, 5 and 6 show the following components: The internal image of the plant, data acquisition system, system identification module, tuning methods of PID, H2 and H_∞ optimization algorithms, communication interface and control system evaluation module. In section 7 examples of different visual screens and displays presented to the user are shown. Finally in section 8 conclusions are resumed.

2 Design Aplication

In order to carry out the application design we have used methods based on real time computer control methods [7, 8] and on automatic control theory [2,9] in order to implement the design cycle of a controller. This can be seen in Fig. 1, where the application architecture is shown, and in Fig. 2 where a flow diagram of the design cycle for a control system is sketched.

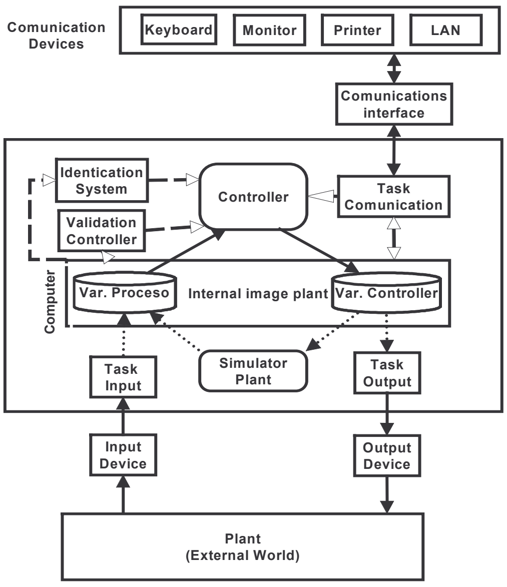


Fig. 1. Application Architecture

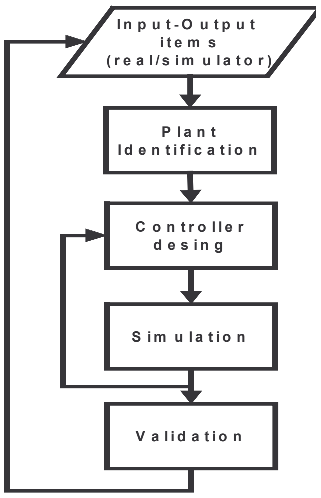


Fig. 2. Design cycle for control system

Each application component can be represented with classes, objects and tasks within application. By means of object oriented programming (OOP) high flexibility is obtained, which permits an easy joint among different components. Besides that facilities to develop structured software in order to divide complex algorithms in smaller modules of easy resolution. Each one of components can be assembled within a generic part that can be implemented by a object oriented framework [10, 11].

The application must work in real time, which supposes an element of additional complexity. Data acquisition and control algorithms execution for industrial processes require strict time restrictions and reliability, due to what the following functionality characteristics must be incorporated:

1. Coordination among real-time tasks.
2. Processing of interruptions and messages within the system.
3. Input-output device drivers to insure that they do not lose data.
4. Inlet and outlet time restriction specifications of the system.
5. To insure databases precision.

To develop a software application with such restrictions, an adequate knowledge of software engineering is needed to avoid possible bottlenecks and to be able to get the maximum performance from hardware and software.

3 Internal image of the plant data

Data from real time signals that the control application must register are the following:

1. Process variable (PV) or controlled variable.
2. Controller output (CO).
3. Setpoint (SP).

Algorithms related with controller design, controller implementation, identification methods, control system analysis, system simulation, as well as system temporary evolution visualization will demand data of iterative form to the internal image of the plant module

One of control system requirements is that the software must work in real time (at least soft real time). Due to this requirement the storage system can not reside in an established database allocated in the hard disk; although a possibility is to use a virtual disk in memory RAM. But this technique is little standardised, and is very dependent upon the Operating System. The adopted solution implements the storage system through a class that reserves main memory dynamically.

The implementation options considered has been the following:

a) A doubly linked list.
b) Round lists doubly linked.
c) To develop a special data structure to reserve buffers on demand.

Most demanded operations by the different algorithms of control that will be implemented related to the memory system are:

a) Adding a new element in head lists.
b) Sequential search of stored values of a signal in the list.

c) To pick up an item of the list. We will use efficient algorithms as the binary search.

Many possibilities are available and it depends on specifications defined in our application to choose among them. Then, independently of the selected method, the software to develop must be flexible enough to avoid be affected by changes of the storage model selected.

The application to develop is based on a monolithic architecture under Windows O.S. The amount of memory available to reserve in this system is 2 Gb maximum, if available as virtual memory (main memory plus swap area on disk).

An innovative storage system has been implemented which is characterized by:

a) As basic element of information to store in each item is utilized a structure formed by the value of the signal and the time instant when data is acquired or generated..

b) It is used 64 Kbytes's dynamic array of rebooted pointers like void except the first. Each pointer reserves a buffer of 65,536 records.

The system will begin storing in the first buffer reserved by the first pointer. When the storage system has stored 65,536 records, and a new record storage is requested, a new dynamic memory reserve will be produced by the second pointer. Thus the system will go reserving memory of fragmented form under demand . With this technique the operating system has bigger facility to shedule the memory. The quantity of records that can be addressed with this structure is greater than 4 thousand million records for each storage signal. In Fig. 3 the storage system structure is shown.

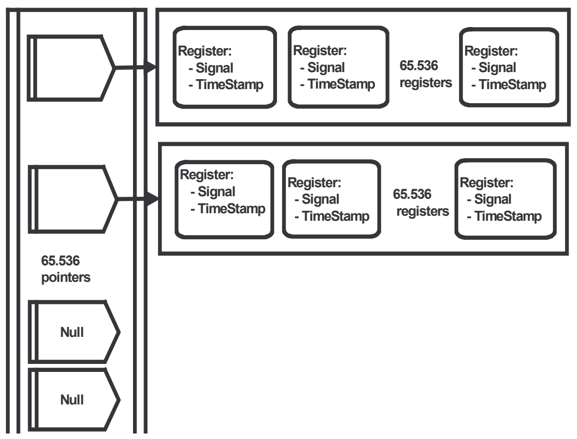


Fig. 3. Data storage structure

4 Data Communication and Interface System

The data communication and interface system implements the following methods used in industrial applications:

1. Data acquisition devices.
2. Interface RS232.
3. Socket.
4. Standard of communication NetDDE.
5. Standard of communication OPC.

Methods 1 and 2 are directly related to hardware components and to the operating system of the computer, for what the developed software is far away from open standards implemented by means of software middleware.

The data acquisition devices that we have worked with are from National Instruments [12]. By means of the NIDAQ library [13] we can implement functions of higher level that enable its communication.

The software technologies that we have considered for distributed communication among computers have been:

1. Technology based in messages through datasocket, NetDDE.
2. Calls to remote intervening procedures RPC.
3. Distributed objects: DCOM, CORBA and RMI [12].

NetDDE and OPC methods permit to establish communication among computers in a network, which permits to share data of distributed equipments with client-server structure.

For each one of the communication methods given before, a library for Builder C++ has been implemented with the following functions:

1. Instantiation of the communication elements for two situations:
   a. Obtaining the respective interfaces in case of RS232 and data acquisition devices; and
   b. Creating the objects necessary for NetDDE and OPC.
2. Obtaining of the configuration parameters for a determined connection.
3. Establishment of the configuration of a determined communication channel. In case of error a code will be returned.
4. Timeout definition for a channel or link.
5. Data read/write in asynchronous mode. Error code and timeout information.
6. Connection and disconnection of a communication channel.

7. Release of resources and restoration of the communication systems. This function will be executed when the main application has finished.
8. Visualization of error codes that has been generated in anyone of previous functions.

5 Simulator System

The functionality of simulation systems is fundamental for following points [14]:

1. Hardware in the loop simulation (HIL) [15].
2. System identification and controller validation.

The following specifications has been defined for the simulator:

1. Continuous time plant model is considered.
2. SISO (Single Input Single Output) and MIMO (Multiple Input Multiple Output, 2x2, 2x3, 3x2, and 3x3) processes can be selected.
3. Simulation algorithms: Euler, Runge Kutta and Runge-Kutta-Butcher methods.
4. Plant model structures:
   a. Second order transfer function.
   b. High order transfer function.
   c. State space model (four matrices).
   d. Data file in Matlab format.
5. Physical restrictions of process and controller variables:
   a. Magnitude saturation.
   b. Magnitude Ratio of change velocity of amplitude signal.
   c. Delays.
6. Effects due to external factors:
   a. Noise.
   b. Disturbances.
7. Simulation in real time mode and in computation time mode. The simulation in computation time mode is utilised by the identification system and for controller pretuning.

In the Fig. 4 a diagram represents simulation system relationship with other parts of the application.

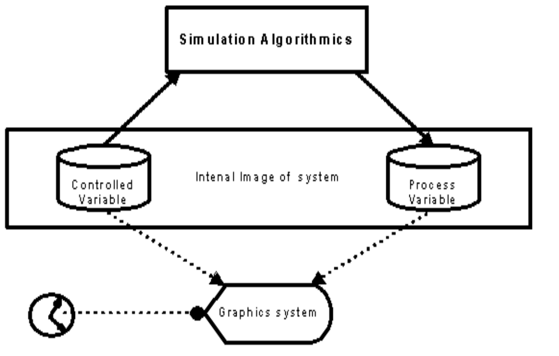


Fig. 4. Relationships among simulation system and other parts of the application.

The flow of data within simulation algorithm is shown in Fig. 5, from controller output (CO) to process variable (PV).

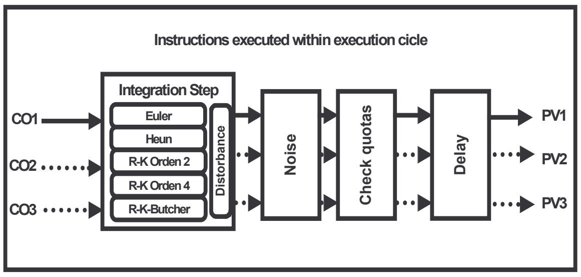


Fig. 5. The flow of data within simulation algorithmic.

In order to implement the system simulator the following functionalities has been developed:

1. Structure of the plant, magnitudes of variables and physical restrictions.
2. Storage system of signals.
3. Library for treatment of variables of temporary type.
4. Library for matrix simple operations: creation and matrix release, add and multiplication.
5. Read function for files with Matlab format.
6. Library of transfer functions: Creation of transfer functions and transformation in matrix space state.
7. Library of auxiliary signals for noise and disturbance simulation.

In Fig. 6 it is shown how different global entities share information.

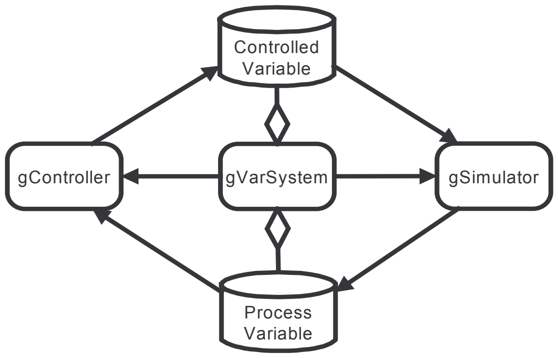


Fig. 6. Relationship among global classes.

6 Controller Structure

The following functionalities are incorporated for the controller:

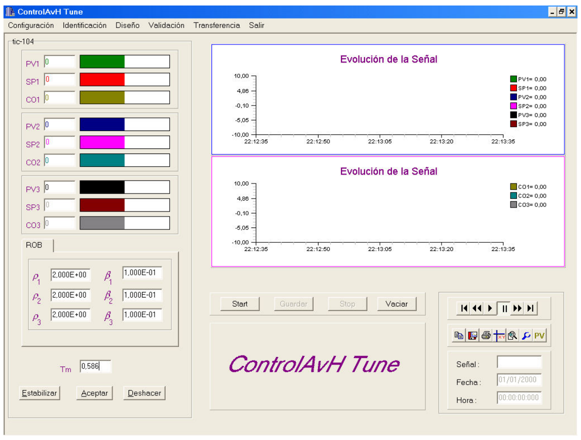
1. Controller initialization.
2. Controller internal parameters setup.
3. Regulator tuning parameters setup.
4. Control action calculation and execution.
5. Release of consumed resources by objects.

The following types of control are considered:

1. Open loop control. This control is utilised for the process identification phase.
2. PID controller (proportional integral and derived).
3. H_∞ & H2 control. A robust control technique which is applied to MIMO systems.
4. Generic controller. It consists of a controller given as four matrices for state space implementation of the controller.

7 User Interface

ControlAvH Tune application has been developed with Builder C++, which is a last generation RAD (Rapid Aplication Development) tool that incorporates a great quantity of standards with a very fast and efficient compiler [16]. The base language is C++ which permits rapidity, flexibility and portability of the developed software [17]. Builder C++ permits us to design the different interfaces that the user has to execute [18]. In Fig. 7 the main screen of the application is shown.



tic-104

PV1	0
SP1	0
CO1	0
PV2	0
SP2	0
CO2	0
PV3	0
SP3	0
CO3	0

ROB

ρ_1	2,000E+00	β_1	1,000E-01
ρ_2	2,000E+00	β_2	1,000E-01
ρ_3	2,000E+00	β_3	1,000E-01

Tm 0,586

PV1= 0,00 SP1= 0,00 PV2= 0,00 SP2= 0,00 PV3= 0,00 SP3= 0,00

CO1= 0,00 CO2= 0,00 CO3= 0,00

Señal:
Fecha: 01/01/2000
Hora: 00:00:00:000

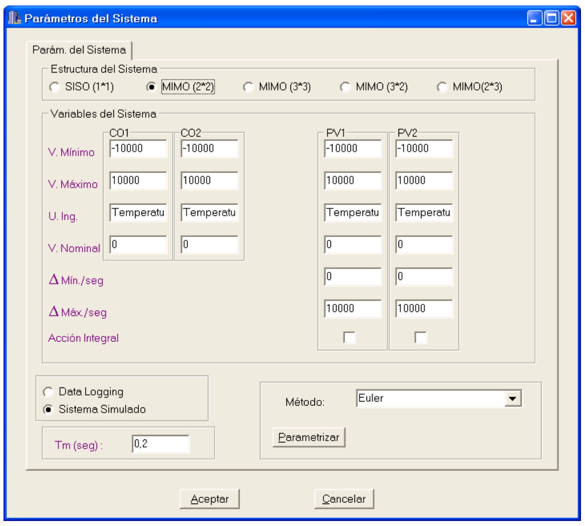
Fig. 7. Main screen of *ControlAv.*

In this screen the following elements can be observed:

1. Graphic system for signals evolution.
2. Start and stop buttons of the control system.
3. Online information of the run times of different tasks.
4. Set of buttons to use the graphic system.
5. Controller parameters for PID and sample time.

The first step is to setup the parameters configuration of the system. By means of the screen shown in Fig. 8, the user assign defines the following:

1. System structures SISO/MIMO.
2. Selection among real process control or simulated system control.
3. Establishment of bounds on input and output signals.



Estructura del Sistema: SISO (1*1) [✓ MIMO (2*2)] MIMO (3*3) MIMO (3*2) MIMO(2*3)

Variables del Sistema

	CO1	CO2	PV1	PV2
V. Mínimo	-10000	-10000	-10000	-10000
V. Máximo	10000	10000	10000	10000
U. Ing.	Temperatu	Temperatu	Temperatu	Temperatu
V. Nominal	0	0	0	0
Δ Mín./seg			0	0
Δ Máx./seg			10000	10000
Acción Integral			☐	☐

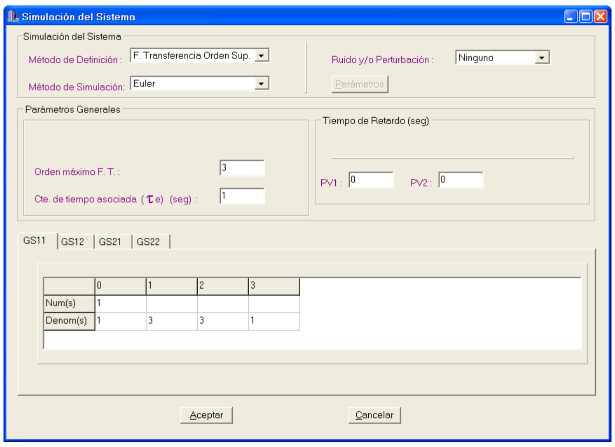
Data Logging [✓ Sistema Simulado]

Tm (seg): 0,2

Método: Euler

Fig. 8. Screen for definition of data on the plant.

If the user wishes to work with a simulated system, the application will show him the screen given in Fig. 9.



Simulación del Sistema

Método de Definición: F. Transferencia Orden Sup.
Método de Simulación: Euler
Ruido y/o Perturbación: Ninguno

Parámetros Generales

Orden máximo F. T.: 3
Cte. de tiempo asociada (τe) (seg): 1

Tiempo de Retardo (seg)

PV1: 0 PV2: 0

	0	1	2	3
Num(s)	1			
Denom(s)	1	3	3	1

Figure 9. Screen for definition of the simulated system.

The following characteristics are selected by the user:

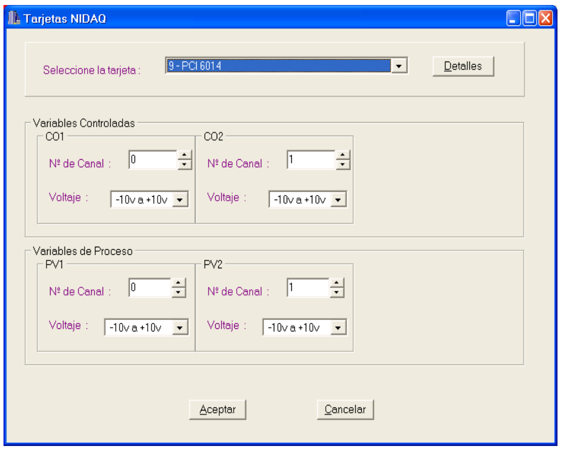
a) System model definition method: Matrix of states space, transfer functions.

b) Simulation algorithm: Runge Kutta order two, Runge Kutta order fourth, Runge Kutta Butcher, Euler, modified Euler.

c) Associate time constant.

d) Noise, disturbances within the plant and delays.

If the user wishes to connect with a external or real process, he will be able to choose among the following methods: data acquisition device of National Instruments [12], RS232 interface, NetDDE and OPC [19]. In Fig. 10 we can see the screen to establish the channels of I/O for data acquisition device.



Seleccione la tarjeta : 9 - PCI 6014

Variables Controladas

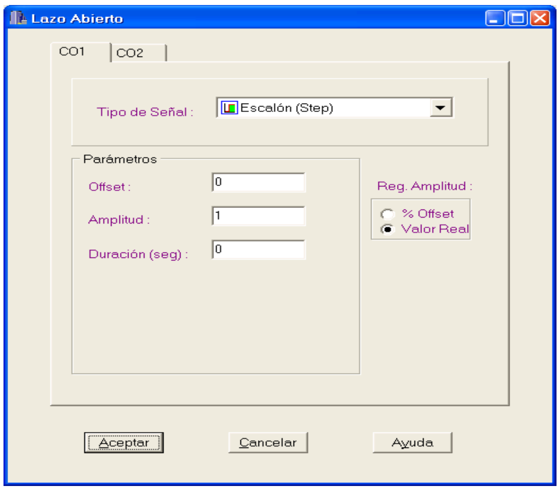
	CO1	CO2
Nº de Canal :	0	1
Voltaje :	-10v a +10v	-10v a +10v

Variables de Proceso

	PV1	PV2
Nº de Canal :	0	1
Voltaje :	-10v a +10v	-10v a +10v

Figure 10. Data acquisition device configuration screen.

In order to carry out the system identification (SISO and MIMO) in open loop, several test signals can be choosen. A sample screen is shown in Fig. 11, where step signal is selected.



Tipo de Señal : Escalón (Step)

Parámetros

Offset : 0

Amplitud : 1

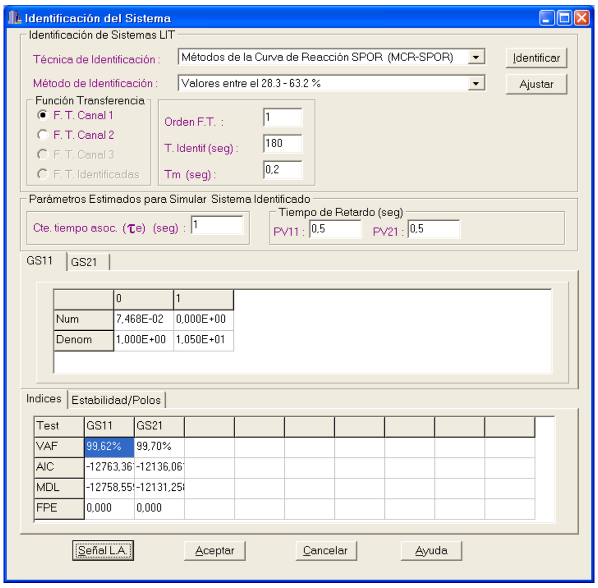
Duración (seg) : 0

Reg. Amplitud : [ ] % Offset [✓ Valor Real]

Fig. 11. User screen for process identification test.

The user will be able to elect among the following list of trial signals: Step, Pulse, Impulse, Sine, Exponential, White Noise, Stocastic Noise.

Finally, the operator will agree to the principal menu item, that he permits the system identification (see Fig. 12).



Identificación de Sistemas LIT

Técnica de Identificación : Métodos de la Curva de Reacción SPOR (MCR-SPOR)

Método de Identificación : Valores entre el 28.3 - 63.2 %

Función Transferencia: [✓ F. T. Canal 1] F. T. Canal 2 F. T. Canal 3 F. T. Identificadas

Orden F.T. : 1

T. Identif (seg) : 180

Tm (seg) : 0,2

Parámetros Estimados para Simular Sistema Identificado

Cte. tiempo asoc. (τe) (seg) : 1

Tiempo de Retardo (seg) PV11 : 0,5 PV21 : 0,5

	0	1
Num	7,468E-02	0,000E+00
Denom	1,000E+00	1,050E+01

Test	GS11	GS21						
VAF	99,62%	99,70%						
AIC	-12763,36	-12136,06						
MDL	-12758,55	-12131,25						
FPE	0,000	0,000						

Fig. 12. User screen for identification process and evaluation.

As it can be observed in Fig. 12, the application permits the user:

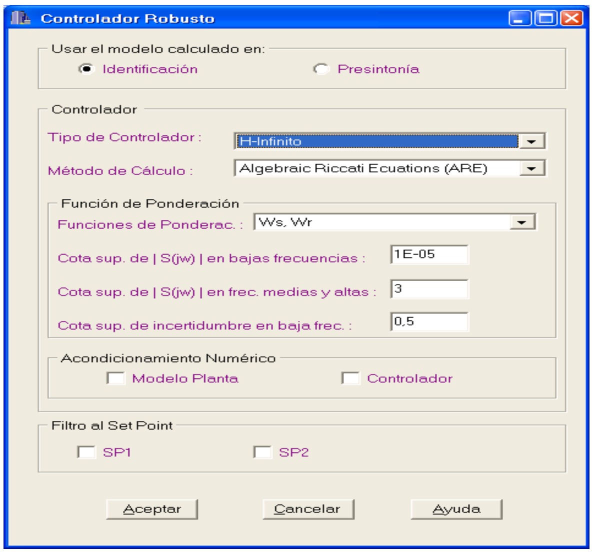
1. To select among a variety of identification methods: Based on the reaction curve of the process, finite impulse response (FIR), finite step response (FSR), recursive least square (RLS), stocastics methods (ARMAX), and domain frequency method based on pulse test and Fourier transform.
2. To establish the parameters of the process of identification.
3. To identify a system in automatic mode, and optionally posterior manual adjustment.
4. To visualize the identified mathematical model.
5. To get information of identification quality indicators (VAF, AIC, MDL, FPE, poles, stability).
6. To compare identified model and real system responses.

Once the sistema has been identified, operator passes to pretuning phase. ControlAv enables to design PID, H2 and H_∞ controllers, where finally, controller is implemented as a generis discrete time controller in the state space form.

For designing H2 and H_∞ controllers ControlAv is charactrized bye the following main properties:

1. Complex mathematical calculus are transparent for the user.
2. Optimal or suboptimal controllers are obtained.
3. Controller order reduction may be made
4. Easy parameter tuning procedure for SISO and MIMO systems is required .
5. Valid for unstable plant control.

Screen for H2, H_∞ controllers design is given in Figure 13.



Usar el modelo calculado en: [✓ Identificación] Presintonía

Controlador

Tipo de Controlador : H-Infinito

Método de Cálculo : Algebraic Riccati Ecuations (ARE)

Función de Ponderación

Funciones de Ponderac. : Ws, Wr

Cota sup. de | S(jw) | en bajas frecuencias : 1E-05

Cota sup. de | S(jw) | en frec. medias y altas : 3

Cota sup. de incertidumbre en baja frec. : 0,5

Acondicionamiento Numérico: Modelo Planta, Controlador

Filtro al Set Point: SP1, SP2

Fig. 13. Disegn robust controller

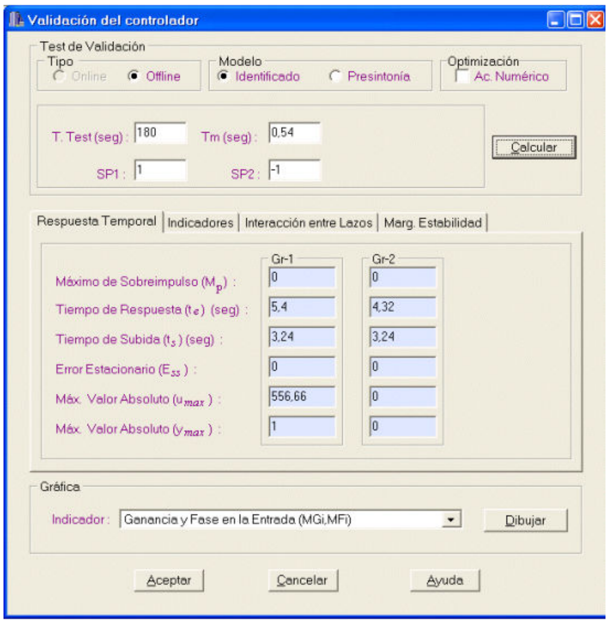
In this screen operator has the following options:

1. Controller is designed using a complete model or a simplified model of the plant, obtained in the identificación phase
2. H2 or H_∞ controller is selected.
3. In case of H_∞ control, controller is obtained using ARE (Algebraic Riccati Equation) approach or LMI (Linear Matrix Inequalities) approach.
4. Weighting transfer functions are selected and their parameters are assigned pretuning values.
5. Controller matrices are balanced to improve numerical conditioning.

With the designed controller (PID, H2 or H_∞), operator may carry out operations for controller analysis and validation:

1. Simulation tests in order to analyze performance by means of indicators such as rise time, overshoot, etc. As it can be seen in the top of Fig. 14, operator may modify the test parameters such as:
   a. Simulation with complete model of the plant or with simplified model of the plant.
   b. Simulation time.
   c. Sample period for controller.
   d. Setpoints values.

2. Robustness indicators calculation based on frequency response (singular values, Bode diagrams). In Fig. 14 a screen with temporary performance indicators is shown, a similar screen for robustness indicators based on frequency responses may be presented if it is selected by the operator.



Test de Validación

Tipo: Online [✓ Offline]

Modelo: [✓ Identificado] Presintonía

Optimización: Ac. Numérico

T. Test (seg) : 180 Tm (seg) : 0,54

SP1 : 1 SP2 : -1

	Gr-1	Gr-2
Máximo de Sobreimpulso (M_p) :	0	0
Tiempo de Respuesta (t_e) (seg) :	5,4	4,32
Tiempo de Subida (t_s) (seg) :	3,24	3,24
Error Estacionario (E_{ss}) :	0	0
Máx. Valor Absoluto (u_{max}) :	556,66	0
Máx. Valor Absoluto (y_{max}) :	1	0

Gráfica

Indicador : Ganancia y Fase en la Entrada (MGi,MFi)

Fig 14. Validation Controller

7 Conclusions

ControlAvH Tune has been developed for implementing the complete design cycle of a control system: data acquisition and monitoring, plant modeling by identification techniques, controller pretuning (H2, H_∞ and PID control), real time control, controller evaluation, controller fine tuning and user interface. Software architecture has been specially designed for obtaining improve performance on user interface, data acquisition, data processing and system identification module, as such as controller tuning and evaluation.

The identification techniques implemented in ControlAvH Tune are specially suitable to PID controller tuning methods, and for using by an innovative methods for tuning H2 and H_∞ controllers. These novel methods are specially valid for process control, for which the user only has to adjust two parameters for each process variable, due to that it is applicable to SISO and MIMO processes.

Acknowledgment:
This work was supported in part by the C.I.C.Y.T project DPI2005-03668.

References:

[1] Camacho, E.F. and C. Bordons, *Model Predictive Control,* Springer-Verlag 2004.

[2] Zhou K., J. C. Doyle and K. Glover (1995), *Robust and Optimal Control,* Addison Wesley Publishing Company 1995.

[3] Expertune, Inc., http://www.expertune.com.

[4] Techmation Inc. *ProtunerTM 32 User Guide*, http://www.protuner.com

[5] Intune: http://www.controlsoftinc.com.

[6] The MathWoks Inc., *Matlab, Simulinik, System Identification Toolbox, Control System Toolbox, Robust Control Toolbox,* http://www.mathworks.com.

[7] Bennet, S., *Real Computer Control. A Introduction, 2nd Edition,* Prentice Hall 1998.

[8] Gazi V., M.L. Moore, K.M. Passino, W.P. Shackleford, F.M. Proctor, J.S. Albus, *The RCS Handbook. Tools for Real-Time Control Systems Software Development,* John Wiley & Sons, INC 2001.

[9] K. J. Åstrom, and B. Wittenmark, *Computer Controlled Systems. Theory and Design 3rd Edition,* Prentice Hall 1997.

[10] G. T. Heineman, and W.T. Council, *Component Based Software Engineering: Putting the Pieces Together*, MA: Addison Wesley, 2001.

[11] B. S. Heck et al. *Software Technology for Implementing Reusable, Distributed Control System,* IEEE Control System Magazine pages 21-35. February 2003.

[12] National Instruments, *Measurement and automation catalog,* http://www.ni.com

[13] National Instruments Corporation, *NIDAQ Function Reference Manual for PC Compatibles,* http://www.natinst.com.

[14] M. Koga, H. Toriumi, M. Sampei, *An integrated software environment for the design, the real and device implementation of control systems,* Control Engineering Practice 6 (1998) 1287-1293.

[15] R. Isermann, et al., *Hardware in the loop simulation for the design and testing of engine control systems,* Control Engineering Practice 7 (1999) 643-653.

[16] K. Reisdorph, et al., *Borland C + + Builder 4 Unleashed*. Sams Publishing 1999.

[17] Schildt, H., *Turbo C++, The Complete Reference*. McGraw Hill, Inc. 1992.

[18] Inprise Corporation, *Borland Builder C + + 5,0, Microsoft Win32 reference,* California U.S.A 2000.

[19] OPC Foundation, *OPC Data Access Custom Interface Specification 2,04,*. September 2000. http://www.opc.com.