Proceedings of the 1996 IEEE International Symposium
on Computer-Aided Control System Design
Dearborn, MI • September 15-18, 1996

SA01 11:30

Embedding Objects into Matlab/Simulink for Process Supervision

J. Meléndez quimmel@ei.udg.es
J. Colomer colomer@ei.udg.es
J.Ll. de la Rosa peplluis@ei.udg.es
J.Aguilar-Martin aguilar@laas.fr
J. Vehí vehi@ei.udg.es

Systems Engineering and Automatic Control Group
EIA department - University of Girona
E-17071, Girona (Catalonia)

KeyWords

Matlab/Simulink, Supervisory Systems, Object-Oriented Approach, Control Architectures, Dynamic Linked Libraries

Abstract

This paper introduces how Artificial Intelligence technologies can be integrated into a known CACSD framework, Matlab/Simulink, using an object oriented approach. The aim is to build a framework to aid supervisory systems analysis, design and implementation. The idea is to take advantage of an existing CACSD framework, Matlab/Simulink, so that engineers can proceed first, to design a control system, and second, to design a straightforward supervisory system of the control system in the same framework. Thus, expert systems and qualitative reasoning tools are incorporated into this popular CACSD framework to develop a Computer Aided Supervisory System Design (CASSD) framework. Object-variables are introduced into Matlab/Simulink for sharing information between tools.

1. Introduction

A goal of this paper is to show how a set of supervisory tools could take benefit of an existent CACSD framework for dealing with expert knowledge when designing supervisory systems (See "Fig. 1"). In this new framework supervisory strategies could be designed based on expert knowledge and implemented as sets of rules in expert systems (ES). Also Qualitative Reasoning (QR) techniques are embedded to assist supervisory systems, specifically by adding qualitative simulation facilities (QS) by means of a representation language, ALCMEN [1] (**A**utomaticians **L**anguage for **C**ausal **M**odelling for **E**xpert k**N**owledge). Moreover, this framework is provided with some abstraction tools that perform tasks of analysis and qualification for obtaining qualitative description of process data and/or events generation.

These are object oriented tools in the sense that information is encapsulated with the access methods needed to set and get this information. Process variables are encapsulated as object-variables.

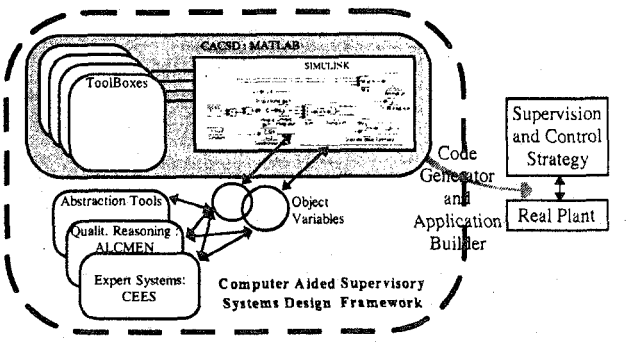

Fig. 1 A Computer Aided Supervisory Design framework using Matlab/Simulink

Matlab/Simulink is used as a CACSD framework and these object-variables are introduced to share information between reasoning tools. The integration of utilities into Matlab/Simulink using an object oriented approach is not implemented as a typical ToolBox because Matlab/Simulink is not conceived to work from the object oriented point of view. Then, some difficulties must be solved. The solution given in this paper uses as MEX files called by Simulink blocks. Object-variables are built into these blocks and information and access methods are available at its outputs.

2. Supervisory Tools into Matlab/Simulink.

To achieve the kind of facilities needed for supervisory systems development an open framework is required. Matlab/Simulink has been chosen as a platform to develop and to integrate all of those tools, because it offers openness and many useful services that can help the user to

develop (ToolBoxes), and to implant (Real-Time Workshop) supervisory systems (See "Fig. 1"). The aim is to dispose of a framework where control and supervisory systems could be developed without the necessity of using external applications [10].

In the bibliography many CACSD applications and specialised packages are developed by using this framework (for example [6] and [7]). Others use Matlab as a support tool (as [5] and [9]). Some examples of linking Matlab or Simulink with an external KB can be found (for instance, [12] or [15]), but there are not examples of embedding these KB into Matlab/Simulink. This is because these are object oriented tools and integration is more difficult. The tools required in supervisory tasks are:

a) A shell for developing ES, is needed to represent expert knowledge when an analytical description is not possible. CEES was chosen because of its capabilities and flexibility.
b) A representation language, ALCMEN, is used to represent qualitative relations. This is a useful way to complete models when plants are partially known.

They are added into Matlab/Simulink in order to get a framework for Computer Aided Supervisory System Design (CASSD). Lets have a look at these tools for better understanding why objects are needed as an integration technology of reasoning tools into Matlab:

Fig. 2 Simulink CEES block

CEES (C++ Embedded Expert System, [8]) is a shell developed under object oriented languages that permits definitions of qualitative models and co-operative ESs as independent objects with agent capabilities. This utility is embedded into Matlab/Simulink as Simulink blocks ("Fig. 2") to perform knowledge based expert reasoning.

For example, a simple expert controller ("Fig. 3"), could be designed using expert rules reasoning about error to deduce if the control signal must be increased or decreased. ES can take advantage of information embedded in the object-variable associated to the error signal. ES will use methods to access that information from object-variables.

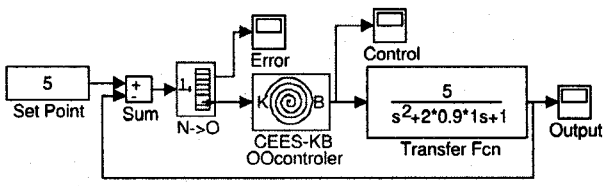

Fig. 3 Object Oriented Expert Controller

In the example only two fields of the object-variable are accessed by the ES (this is the signal value and its derivative). Control law is given as follows:

$u(k) = u(k-1) \pm \Delta u$

$\pm\Delta u$, is deduced by ES when reasoning about error signal (e(k) and Δe (k)).

In "Fig. 4", evolution of process signals is represented when a change in the setpoint is introduced. Control signal is obtained directly from the expert rules :

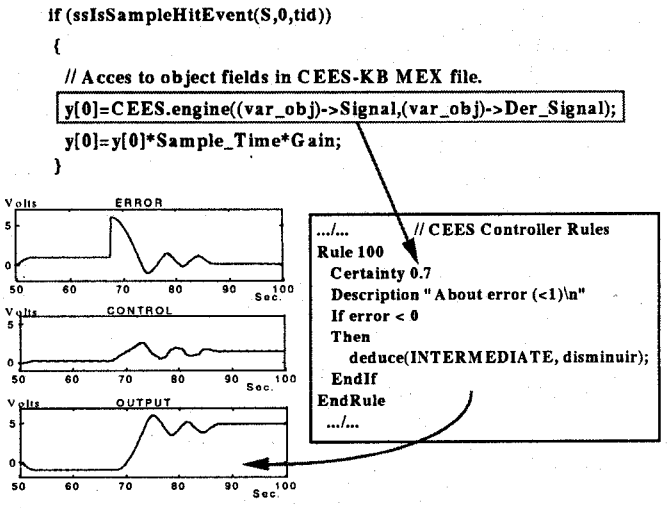

Fig. 4 Object fields are accessed in the MEX files to supply Expert Rules with data to deduce the control law of the example in "Fig. 3"

ALCMEN (Automaticians Language for Causal Modelling for Expert kNowledge, [1][2]) was conceived as a set of blocks with capabilities of representing imprecision with the aim of facilitating the communication between process engineers and control engineers. It was originally conceived as follows:

Blocks: It is a causal graph with three variables associated to, *input* or cause, *output* or effect, and *parameters* or conditions.
Variables: They are objects with attributes as *type* (numerical, qualitative, or mixed), *range* (set of possible values), *subsets of values* (desired or usual values), and N/S conversion.
Relationships, between variables can be chosen among a predefined set although user can build new ones when are needed. The defined relationships are:

a) Static relationships:
a.1) Correspondence tables (Single input, single output) : Linear tables, Extreme tables.
a.2) Combination relationships (Multiple inputs, single output).

b) Dynamic relationships (time depending): Qualitative delay, Qualitative derivator/integrator, Trends.

In ALCMEN, algebraic and differential equations have a representation for qualitative values. Then, a N/S conversion is needed. It is performed as a qualification of amplitude space of signals in zones and associating indices to these zones. ALCMEN has been developed as Simulink ToolBox as a set of blocks that perform this relationships (static and dynamic) between qualitative variables ("Fig. 5").

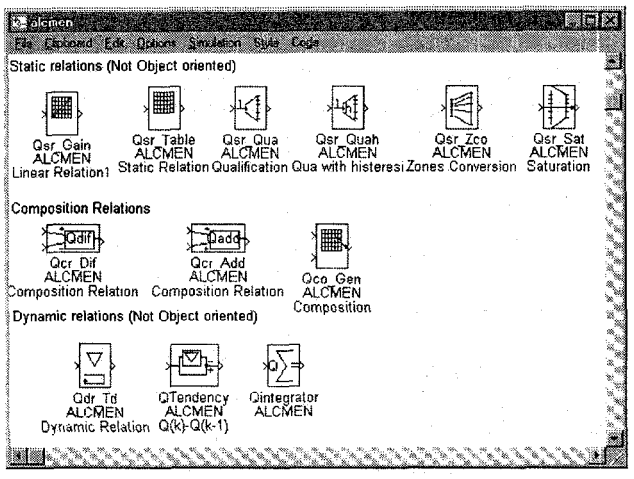

Fig. 5 ALCMEN as a Simulink 'ToolBox'

The main goal of ALCMEN in supervisory systems design is to deal with qualitative relationships in order to deduce inaccessible dynamics, as a qualitative observer, or to supply ES with more significant information, for instance. In this case the object oriented approach is used to add methods to obtain qualitative information from variables and to set it in the same object. Then other tools could access the needed information. ALCMEN blocks can use qualitative information to perform qualitative simulation and ES can use this in reasoning tasks, as well.

According to these tools, an object oriented approach for embedding significant qualitative and numerical information in object-variables could be useful to simplify the design of supervisory systems.

3. Object-variables

Expert reasoning is not associated to isolated process variables but takes into account the whole process dynamics and its influence to the individual process variables behaviour. Reasoning, therefore, becomes easier if all information from the process is clearly encapsulated and locatable [10] [11]. This information could be used for various tools to perform the supervisory strategy. Then access methods must be embedded with data in the same object.

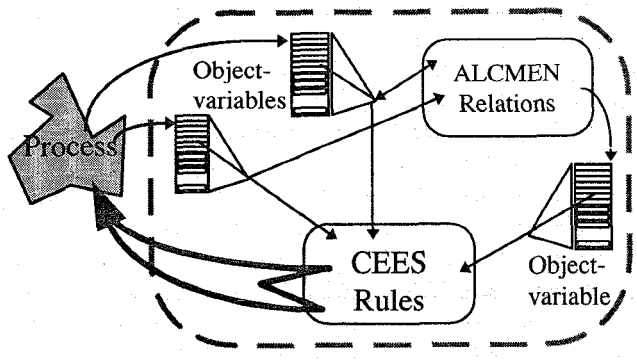

Fig. 6 Object-variables allow information to be shared.

This means that, for reasoning, each process variable must be fully analysed to obtain information of process behaviour. Then, object-variables could be used to encapsulate this information. A major strength of objects is that they represent a concept with its associated characteristics as a single entity [3]. Additional information must be supplied with each variable to help reasoning to discriminate between variables of processes as for instance, a control variable of a level regulation process is qualitatively different from a DC motor speed control variable. Thus, to encapsulate this supplementary information, data structures or objects are an elegant form of supplying this further qualitative information. "Fig. 7" shows how object-variables are declareted using C++. This is introduced into Simulink as S-functions blocks that calls a MEX-file. In this case, the class declaration is obtained as an extension of the ALCMEN definition of variables and the history of signals is added in a temporal sliding window.

```
class VAR_OBJ
{
 public:
  // Signal
   double Signal ;
   double Der_Signal;
   double Der2_Signal;
   int Q_Signal;
   Window WSignal[MAX_NUMPOINTS];
   ... / ...
  // Parameters
   ... / ...
   //Methods:
   VAR_OBJ();
   SetParam(double, double, int, double *,int, double *, int);
   actualitzaST(double, double);
   actualitzaWSignal(double, double, double, double);
  ... / ...
};
```

Fig. 7 Objects for the Supervisory framework

Information in an object-variable is obtained directly from signals and performing simple operations that sometimes need external parameters or process characteristics. In Matlab/Simulink blocks, these parameters are supplied

externally, using dialogue windows. "Fig. 8" shows these capabilities. In this case external parameters are used to define temporal windows and amplitude zones for qualitative representation of signals according ALCMEN definition.

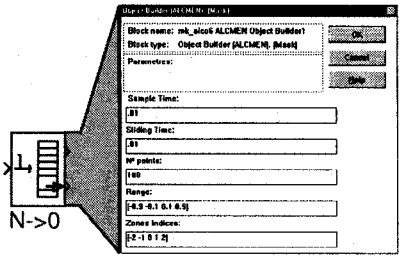

Fig. 8 Simulink object builder block and its dialogue window.

4. Embedding objects into Matlab/Simulink: Technical viability

By working with S-functions in Matlab/Simulink, the user can add new general purpose blocks or incorporate existing 'C' or 'FORTRAN' code into simulations. The only constraint that users must take into account is that the new code must be written into predefined routines. These routines will be called by Simulink at each simulation step.

'SimStruct' [17][18] is the data structure used by Simulink designers to encapsulate the block's information. Each block in a Simulink representation has associated a 'SimStruct' that is accessed by Simulink to perform simulation correctly. Then, the information embedded in this data structure allows Simulink to know the parameters associated to this S-Function as well as to access user defined routines.

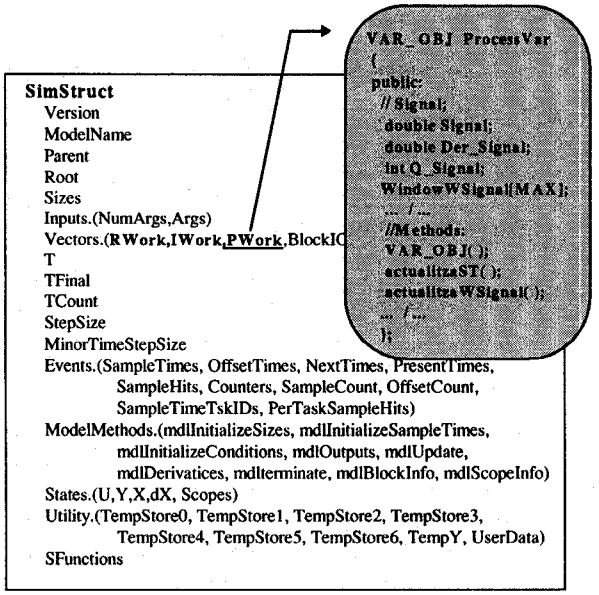

Fig. 9 Internal structure in Simulink. Solution to embed objects.

Among the structure fields there are three fields where the user can save restricted information, usable only by the object owner. These three fields are associated with work vectors that correspond to pointers to integers (int *), to doubles (double *), and pointers to void pointers (void **). In the 'SimStruct', see "Fig. 9", these fields are labelled as *'vectors.RWork'*, *'vectors.IWork'* and *'vectors.PWork'*. Therefore, the only method to embed external information into Simulink 'SimStruct' is using these working vectors.

By taking advantage of this capability, it is possible to associate an object to each Simulink block by saving a pointer (to this object) in the *'vector.Pwork'* field in its 'SimStruct'. Also, the user can have an object associated to each Simulink block.

In order to be successful, some considerations must be taken into account. The source file for getting a .MEX file (as a result of compilation of an S-function) is structured according to a template structure defined by Simulink developers [17]. This structure is composed of routines where the user can insert his own code. In these routines of the future .MEX file some advice must be followed:

- *obj_def. h* → Class definitions and object access method can be defined in an include.h file.
- *mdlInitializeSizes()*→ A pointer vector will be used and must be specified: The command *'ssSetNumPWork(S,1);'* must be used.
- *mdlInitializeConditions()*→ Memory space must be allocated to be used by the object and its pointer.
- *mdlOutputs()*→ A method must be activated to access the desired fields.
- *mdlUpdate()*→ To fill the fields of the object.
- *mdlTerminate()* → Remember to free memory.

The fact of working with objects implies using an object oriented compiler, and at the same time, access to data pointers (C++ compiler) must be allowed. In this case, Watcom 10.5 C++ compiler was chosen.

5. Example

In the next paragraph a supervisory structure is designed for fault detection. The objective is to detect abnormal situations of a controlled plant using an ES. The plant is formed by two coupled tanks and a PID controller and faults are given when connection pipes are blocked (See "Fig. 10"). Rules used in the fault detection are obtained from the process engineer's knowledge. This means that ES must be supplied with 'qualitative observation' of variables behaviour. For instance *qualitative tendency* and *oscillation degree* of error signal is needed. This information could be obtained from measured signals according to abstraction algorithms that use information embedded into object-variables. In "Fig. 10" block labelled as *" Abst(Err)"* performs this task.

At same time expert rules, also, use qualitative information from difference of levels (Level_1-Level_2), but level_1 is not accessible. In this case ALCMEN relationships are used as a qualitative observer to estimate this variable [14].

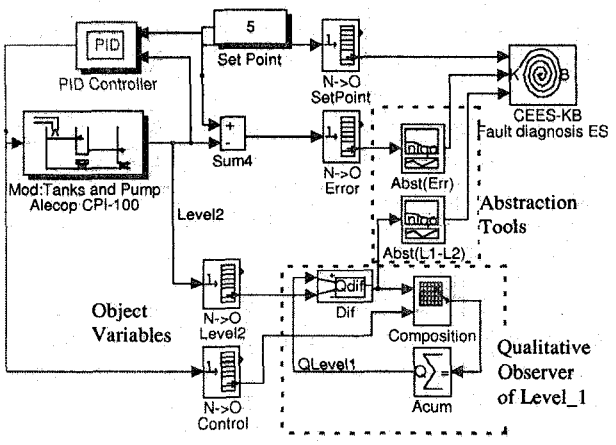

Fig. 10 Example of using Matlab/Simulink as a CASSD

"Fig. 11" shows the compared evolution of qualitative Level_1 obtained from qualitative simulation and the N/S conversion of Level_1 measured signal (in this case obtained directly from the simulation for the comparison). See that qualitative values for Level_1 are coincident with the permanent regime. Dynamics are also preserved although some undesirable transition between indices occurs. This is due to the nature of QR with crisp transitions between zones.

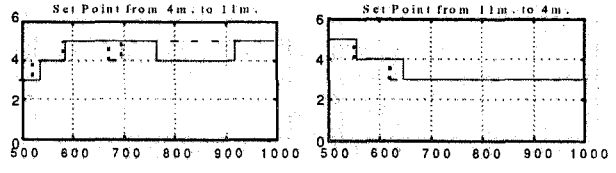

Fig. 11 Comparison of Qlevel 1 (indices) obtained from Qualitative Relations (dotted line) and from N/S of Direct Measure on Process (solid line)

Finally, this significant information is supplied to the ES every sample-time to fire the necessary rules and deduce about the behaviour of the plant. This rules (See "Fig. 12") access information embedded into the object-variables connected to the inputs of block labelled 'CEES-KB' ("Fig. 10").

This example was partially implemented before linking an external ES developed under G2 for reasoning about a Simulink simulation [13]. The problem was to develop tools for abstracting qualitative information used by the ES. Development becomes easier using the same framework because the integration problem is solved automatically.

```
Rule 10
   Certainty 1 Threshold 0.2 TraceHere Yes
   Description "Regim status: Transient "
   IfOr
      tendency_level->fv.v == great
   Or
      tendency_error->fv.v == great
   Then
     deduce (INTERMEDIATE, transient)
   EndIfOr
EndRule
```

```
Rule 210
   Certainty 1. Threshold 0.45 TraceHere Yes
   Description " Valve_1 Blocked . "
   IfAny (PF)
      PF->result == anomalia2
      And
      PF->cf
      And Alternative
        levels_dif->ups_in_interval()
        Or
        levels_dif ->fv.greater(dnivells->fv.low)
      EndAlternative
   Then
      deduce(INTERMEDIATE, v1_blocked)
   Otherwise
      deduce(INTERMEDIATE, v1_open)
   EndIfAny
EndRule
```

Fig. 12 CEES rules used in fault detection

6. Conclusions

Integrating supervisory tools in the same framework is a way to facilitate the transfer of perceptual knowledge and inference of process/control experts to supervisory systems. This paper shows how a Matlab/Simulink can perform this tasks if objects are introduced as integrating technology. Some features of dealing with object-variables into this new CASSD are:

Advantages :

- »Further analysis in the methodology to guide engineers in the development of supervisory systems.
- » Integrated framework of control/supervision.
- »This methodology could be useful for large projects of control/supervision.

Drawbacks :

- » A problem with object-oriented structuring is that is highly application dependent. This means that this variables must be general and not all its capabilities will be used in each design.
- » The integrated tools are not fully developed yet, but they will be in the near future.

Constraints

- » A single class of object must be defined and, of course, all object-oriented tools must know how to access to the fields of this class.
- » When a variable is encapsulated into an object, its primitive value must be accessible transparently from non object-oriented tools.

7. Future Work

Further work on Matlab/Simulink to better support object oriented tools so that our team group could go on to

.research supervisory control architectures, methodologies, and practical applications. A future version Matlab/Simulink is announced to incorporate software structures. Structures will facilitate the application of these ideas to embed objects better. However these structures are not expected to be real objects as required.

8. Acknowledgements

This work continues thanks to several suggestions and comments of Dr. J. Aguilar within the ***TAP96-1114-C03-03*** project *Plataformas Integradas de CAD de Supervisión y Metodolgías* of CICYT program from the Spanish government.

9. References

[1] Aguilar-Martin J., 1991, "Representación simbólico numérica para sistemas expertos de control en tiempo real", Curso de Verano Universidad Internacional Menendez Pelayo, Santander.

[2] Aguilar-Martin J., 1994, "Qualitative control, diagnostic and supervision of compex processes", *Mathematics and Computers in Simulation*, 36, pp 115-127.

[3] Årzen K-E. "Experiences of using G2 for real-time Process Control", in "Supervision de processus à l'aide du système expert G2", Ed. Hermes, Paris, 1995.

[4] Barker H.A., «Open Environments and Object-oriented Methods for Computer-Aided Control System Design», *Control Engineering Practice*, vol 3 nº 3, pp 347-356, 1995.

[5] Blankenship G.L., Ghanadan R., Kwatny H.G., LaVigna C., Polyakov V., «Tools for Integrated Modeling, Design, and Nonlinear Control», *IEEE Control Systems*, vol 15 nº 2, pp 65-77, 1995.

[6] Bohn C. and Atherton D.P., « An Analysis Package Comparing PID Anti-Windup Strategies», *IEEE Control Systems*, vol 15 nº 2,pp 34-40, 1995.

[7] Chipperfield A.J., Fleming P.J.,« PARSIM: A parallel Optimization Tool», *IEEE Control Systems*, vol 15 nº 2, pp 48-53 1995.

[8] De la Rosa J.Ll, «Heuristics for cooperation of expert systems. Application to process control», *Doctoral Thesis*, Universitat de Girona, 1994.

[9] Grübel G, «The ANDECS CACE Framework», *IEEE Control Systems*, vol 15 nº 2,pp 8-13, 1995.

[10] Lynch P.M., De Paso J., «An Object Oriented Intelligent Control Architecture», American Control Conference 1992.

[11] Jobling C.P., Grant P.W., Barker H.A., Townsed P, «Object-oriented Programming in Control System Design: a Survey», *Automatica*, vol 30, nº 8, pp 1221-1261, 1994.

[12] Melendez J., Colomer J. and De la Rosa J.Ll., "Linking G2 and Matlab for Developing an Expert Diagnostic System", in '*Supervision de processus à l'aide du système expert G2™*', pp. 91-110, Ed. Hermès, Paris and also Report de recerca, Universitat de Girona 95/010-EI, Girona (Spain), 1995.

[13] Melendez J., "*Integrating Tools for Computer Aided Expert Supervision Design*, pp. 29-44, "1r Seminari de treball en Automàtica Robòtica i Percepció", Edicions UPC, Barcelona , Feb 1996.

[14] Melendez J., Colomer J., De la Rosa J.Ll., Vehí J., "Dealing with Qualitative Information in Simulation for Supervisory Systems design", in Modelling and Simulation, ESM'96, Budapest, June 1996.

[15] Ong E.K.,"Autonomous Control System Design", ACC/TM3, p.p. 1898-1899, 1992.

[16] Rutz R., Richert J.,« CAMel: An open CACSD environment », *IEEE Control Systems*, vol 15 nº 2,pp 26-33, 1995.

[17] Simulink 1.3 User's Guide, The MathWorks, Inc, April 1993.

[18] Simulink 1.3 Release Notes, The MathWorks, Inc, May 1994.