THE GE MEAD COMPUTER-AIDED CONTROL ENGINEERING ENVIRONMENT

James H. Taylor[†], Dean K. Frederick[*], C. Magnus Rimvall[†], and Hunt A. Sutherland[†]

† Control Systems Lab
GE Corporate R & D
PO Box 8
Schenectady, NY 12301

* Dept. of Electrical, Computer, and
Systems Engineering
Rensselaer Polytechnic Institute
Troy NY 12180

ABSTRACT

The GE MEAD Project (Multi-disciplinary Expert-aided Analysis and Design‡) involves the integration of several computer-aided control engineering (CACE) packages under a supervisor which coordinates the execution of these packages with a data-base manager, an expert system, and an advanced user interface. The principal components are, in functional terms:

- a supervisor, which integrates the underlying CACE packages and coordinates all activity within the GE MEAD Computer Program (GMCP),
- an expert system shell and rule bases for "expert aiding" specific procedures to relieve the user from unnecessary low-level detail,
- a data-base manager to track system models that evolve over time along with associated results, and
- a user interface, to facilitate access to the CACE package capabilities by permitting the user to work in several modalities, i.e., menu/forms style, using GE MEAD commands, using the core packages' native commands, or using the GE MEAD Macro Facility.

The variety of interaction modes supports both inexperienced and expert users most flexibly and effectively. The data-base manager and expert system relieve the user of much manual "overhead" that is required using CACE packages that are unsupported in these areas. The operation of the GMCP, including the user interface, expert system and data-base manager, will be illustrated by examples.

1. INTRODUCTION

There is a growing need for improved performance from embedded control systems in many technology areas - e.g., aerospace and transportation systems, manufacturing processes, and consumer appliances. These increasingly stringent demands require the use of advanced control technology. Two major trends in this regard are integrating the control of subsystems, thereby implementing the overall "systems approach" to design, and the development and application of approaches to accommodating dynamic variability, uncertainty, component failures, and other effects and phenomena that may degrade control system performance in some sense.

Requirements for advanced integrated control of technological systems necessitates improvements in CAD software, so better designs can be obtained at less cost in terms of engineering time and effort. The needed improvements include a better user interface, so that users of varying degrees of expertise can be supported, rigorous data-base management, so better engineering practices can be facilitated, and expert aiding, to relieve the user of unnecessary tedious or low-level tasks. Improved CACE software environments has been the thrust of research and development at GE Corporate R & D since 1981 [1 - 9]; the specific improvements just listed are the primary goals of the GE MEAD Project.

Our approach to creating the GMCP has been to take maximum advantage of existing software modules. Implemention thus entails the integration of CACE packages under a *Supervisor* which coordinates the execution of these packages with a data-base manager *(DBM)*, an *Expert System Shell (ESS)*, and an advanced *User Interface (UI)*. The resulting architecture is depicted in Fig. 1. The underlying CACE tools ("core packages") include the PRO-MATLAB® package for linear analysis and design and a choice of the ACSL® or SIMNON® programs for nonlinear simulation, equilibrium determination, and linearization.

The GE MEAD Project and GMCP development are discussed in [9]. Here, we will present the operation of three of the central components of the GMCP, namely the data-base manager (DBM), the expert system (ES), and (secondarily) the GMCP user interface (UI). The illustrations will be presented by showing how the user takes advantage of these capabilities using the UI; for more detail concerning the user interface, see [8].

‡ The acronym MEAD originated in the Air Force MEAD Project, which is a parallel / synergistic effort to that described here. The USAF MEAD effort was sponsored in part by the Flight Dynamics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), United States Air Force, Wright-Patterson AFB, Ohio 45433-6523, under contract F33615-85-C-3611. The USAF MEAD Project is overviewed in [7].

® PRO-MATLAB is a registered trademark of The MathWorks, South Natick MA 01760; ACSL is a registered trademark of Mitchell and Gauthier Associates, Concord MA 01742; and SIMNON is a registered trademark of Lund University, Lund, Sweden.

2. GMCP FUNCTIONALITY

The following list captures the *basic* CACE functions that are directly performed by the present GMCP:

1. Modeling: nonlinear and linear systems, continuous- and discrete-time systems; building arbitrarily structured models from components
2. Simulation: initializing system state variables, setting system parameters, defining input signals, designating simulation variables for storage, running a simulation
3. Steady-state (equilibrium) determination
4. Linearization
5. Linear analysis: eigenvalues/eigenvectors, zeros, controllability and observability, model reduction, model transformations, root locus, frequency response
6. Linear control system design: frequency-domain methods, pole placement, time-domain methods (LQG, LQR)
7. Control system validation: frequency-domain analysis of linear models, time-domain analysis (simulation of linear and nonlinear models)

This list is *not* all-inclusive from a control theoretic point of view. The objective of the first phase of the GE MEAD Project is to define and create a state-of-the-art environment that directly supports basic CACE functionality. Furthermore, the nature of controls is such that creating an exhaustive catalog of functionality would be difficult or impossible, and new approaches and theories are being added on a continuing basis. The GMCP environment has thus been designed to be "open", in the sense of being extensible either by adding built-in functionality or by use of GE MEAD macros or the "Package Mode" access to any CACE functionality of the core software (see Section 5).

3. GMCP DATA-BASE MANAGER (DBM)

Requirements for a data-base manager for CACE are presented in [5]. The basic data elements are *models* which are comprised of *components* and a *description* (containing model type, connection definition, key variable names, etc.). Associated with each model there are *results* (e.g., files containing frequency-response or time-history data). Models and results are often organized in terms of *projects* (e.g., project = *GE654* for the analysis and design of a new turbine control system to be designated the GE654). These observations led to the basic data-base (DB) organization portrayed in Fig. 2. This hierarchy supports control engineers who naturally think of projects and models as being of paramount importance; all data elements produced during CACE activity are "children" of these entities.

While these CACE DB categories are few in number and simple, there are several factors that complicate the data-base management. Models tend to change over the life-time of the project, some results are also models (e.g., linearizations of nonlinear models or transformed linear models), and components are often used in several models yet they should be stored in one location to simplify their maintenance. The GMCP DBM incorporates mechanisms to handle all of these situations:

- The DBM keeps track of system models that evolve over time using a standard "version control" approach in the software engineering sense. Generally the model changes at the component level, e.g., a component is updated whenever better modeling information becomes available or as preliminary modeling errors are corrected. This motivated the use of a tool that tracks each *version* of a *component* (e.g., rotor dynamic model) so that version = 1, 2, 3, ... refers to the original and subsequent refinements of this component, and each *class* of a *model* (e.g., a complete control system) that incorporates the component. Thus the user is always working with a specific class of any given model, and each analysis or design result is associated with the correct model instance. This is illustrated in Fig. 3, where we observe that model Turbctrl has three classes in the data base (class = 2 has been purged), and each class has its own set of results.
- Maintaining data-base integrity at the component level is difficult if several copies of the same component must be separately stored and maintained for use in various models. The GMCP DBM supports *non-redundant* model management via *links*. These allow the engineer to maintain each component in one model (the "home" model) and use it elsewhere by bringing it out of the home DB and incorporating it in any other model ("linking" to the component). Note that different classes of the home model may correspond to different versions of the linked component; if so, creating a new class of other models using links will also use the latest version of the linked component. This feature is also depicted in Fig. 3.
- Relations between results and models are tracked in the GMCP DBM using an association called the *reference*. This is required, for example, when a linearization is saved both as a *result* obtained using a nonlinear model and as a *component* that may be used in other models for analysis and design. The same need exists with regard to linear model transforms. For example, one may create a reduced-order version of a linear model, and save this as both a result and a model for further study; this is illustrated in Fig. 3 where 'Ltlrotor' is maintained both as a result and a component. The engineer may access the reference (*'Ref'*) to determine where the reduced-order model result exists as a component in a model; from the other side, the reference of component 'Ltlrotor' may be checked to determine that it was obtained as a result using model 'Turbine:7'. Thus, any derived linear component in a GMCP DB can be traced back to determine how it was obtained (e.g., with what model using what method); this protects the value of such models.

Finally, there is a secondary data element stored in the data base called the *condition specification*. This element contains information regarding operations performed on a model before a result is obtained. These operations include changing a parameter value, specifying an initial condition and/or input signal before performing a simulation, defining a frequency list before obtaining Bode plot data, etc. The condition specification also records numerical conditions, such as setting a tolerance for determining controllability or observability, selecting an integration algorithm for simulation, etc. Capturing this data is critical, since it is the combination of model instance *and* condition specification that determines the result and thereby allows the engineer to document or repeat the result. Condition specifications are stored in the GMCP data base and may be recovered for any result that has been saved.

In summary, the GMCP supports a simple hierarchical organization of Projects, Models, Components, and Results/Condition-Specification pairs in the database. Models are tracked over time via class number and version control. In addition, Link and Reference relations completely maintain the integrity of the database. All of these lower-level data-base relations are portrayed in Fig. 3.

The DBM functionality outlined above is provided by the GMCP with little or no extra work by the user, as the example below illustrates. The supervisor and DBM manage DB organization and version control with no effort from the user beyond supplying personally meaningful names for new elements. In fact, accessing the database via the GMCP Browsing Facility and having the ability to make direct use of data elements from that facility makes the DBM an asset rather than a liability in terms of overhead, as will be demonstrated in Section 5.

4. GMCP EXPERT SYSTEM

The expert system shell (ESS) provides the basis for "expert aiding" clear-cut but complicated procedures that would otherwise involve the user in unnecessary low-level detail. A concept for expert-aided CACE was originally defined in [2]; the primary difference in GE MEAD involves adopting a less ambitious model for expert aiding that makes the expert system the *user's assistant* [6] rather than putting it in charge of the CACE effort being performed. This change in perspective was motivated by the specific goal of providing support without forcing the control engineer to do things the Expert's way; this approach is called the "control engineer's assistant" paradigm.

The GMCP expert system shell interfaces with the supervisor in exactly the same fashion as the user working through the user interface. The ESS outputs GE MEAD commands and/or package commands to the supervisor, and gets the same return as the UI, i.e., a result (if little data is involved), a file name (for larger results), or messages (errors or information). This simplifies "knowledge capture", since this is exactly the output and input of the supervisor when the expert user performs the task at hand.

The first CACE rule base to be built in the GMCP was a simple "eigenvalue assessor", created to define and test the ESS-to-Supervisor interface. A more substantial rule base presently under test is a reimplementation of the lead/lag compensator design expert system described in [4]. The function of the latter rule-based system is to accept specifications for steady-state error coefficient, bandwidth, and gain margin, and design a lead / lag compensator to meet these specifications if possible within constraints (e.g., order of the compensator). At the end of this task, the expert system performs a simulation of the step response of the designed control system; the user may inspect the result and accept the design if the response is satisfactory or iterate on the input specifications for another design trial. The invocation of a rule base to perform an expert-aided task is identical to executing a conventional task.

5. GMCP USER INTERFACE

The GMCP user interface (UI) is designed to facilitate access to the CACE package capabilities by users with widely different levels of familiarity with the environment, to unify access to the core packages despite very different package interfaces, and to simplify the use of the data-base manager. A more detailed discussion of the GMCP user interface may be found elsewhere [8]; a brief overview is provided here so the context is clear for the example in Section 6.

The goal of creating an environment that is accessible to users of widely different levels of expertise was achieved by permitting the user to work in a number of modes:

- *IDE Mode* (IDE = Integrated Design Environment), from a menu/forms style UI for basic CACE functionality,
- *M_Command Mode,* i.e., using GE MEAD commands when this expedites CACE work compared with the menu/forms interactive mode,
- *Package Mode,* i.e., using an interactive core packages' native commands when the exact desired functionality is not available via M_Commands, and
- *GE MEAD Macro Mode,* which includes both a macro-execute mode and a flexible macro-edit mode and is based on M_Commands, Package Mode commands, or a combination of these.

The availability of various interaction modes and the extensibility afforded by the GMCP Macro Facility supports the inexperienced user as conveniently as possible (primarily via IDE), while providing the more experienced GMCP user with a most effective environment for CACE. Note that the GMCP DBM functionality is available in all modes, even Package Mode where the user appears to be using a core package without interacting with the GMCP (see the results catalogued at the end of the example).

6. EXAMPLE

The GMCP environment has been used to create an illustration of the capabilities and features described above. The demonstration scenario involves creating the following data-elements:

- a new project named Tampa;

- a model called Twoloop which is comprised of components that are either original or linked from other sources (the user's Library DB and a model in project Orlando); then class 2 is created by editing one component;
- results using class 2 of Twoloop, including a derived model (a discrete-time transformed version); and
- a monolithic (one-block) version of Twoloop (which can be used as a component for hierarchical model building).

Figures 4 through 10 illustrate this activity; these are discussed in detail in the paragraphs below.

Figure 4 depicts the model Twoloop, which is the first model to be created in project Tampa. The user begins by browsing projects and creating project Tampa as shown in Fig. 5. The button 'Create Proj' was clicked previously, the user was prompted for a new project name, and the data base for Tampa was set up and now appears on the Browse Projects Screen, ready for model building.

Creating a model using the GMCP involves defining the model in terms of name, type, and configuration, then defining the components, and finally connecting the components. Figure 6 shows the first step: the user names the new model Twoloop, specifies type = ABCD (state-space continuous-time), and declares it to be a general configuration ('General Config') because the two-loop structure in Fig. 4 does not fit any of the standard forms. Then the user is prompted to define components as portrayed in Fig. 7 for the first component ('Block1'). This may be accomplished in one of four ways: the user may -

- create a new component, as shown in this case, by clicking 'NEW', then declaring the format to be ABCD (i.e., the state-space form indicated in Fig. 4 for the component 'process'), and then clicking 'New'; this will bring a template in ABCD form into the editor for the user to insert the desired arrays A,B,C,D; the other alternatives under 'NEW' ('Edit file', 'Use file' in the bottom row of buttons) allow the user to install a component file that is not presently in the data base;
- link to a component of a different model by clicking 'LINK' (the second option of Fig. 7);
- copy a component from a different model by clicking 'COPY'; or
- link to a component from the user's Library (a special-purpose project) by clicking 'LIBRARY' (this is a streamlined way to access components from one central location).

Finally, the user clicks on the button 'CONNECT' (also shown in Fig. 7) which causes a connection template to be written and brought up in the editor for completion. The completed template is as follows (user inputs in CAPITAL LETTERS):

```
system twoloop(process,first,mylag,innerfbk) is
  %% Please enter the connection for the component
  %% blocks just entered, in terms of **block inputs**
  %% and **system outputs**.
  %%
  %% Component outputs available for connection:
  %% ------------------------------------------
  %%  process.out(1)
  %%  first.out(1)
  %%  mylag.out(1)
  %%  innerfbk.out(1)
  aliases (optional)
    system.in(1)  : REF;
    system.in(2)  : DIST;
    system.out(1)  : OUTP;
  end aliases;
  connections
    process.in(1)  = FIRST.OUT(1)-INNERFBK.OUT(1)+DIST;
    first.in(1)  = REF-MYLAG.OUT(1);
    mylag.in(1)  = PROCESS.OUT(1);
    innerfbk.in(1)  = PROCESS.OUT(1);
    system.out(1)  = PROCESS.OUT(1);
  end connections;
end twoloop;
```

The details of building the system model from these component and connection definitions using PRO-MATLAB is completely managed by the supervisor.

Once the model Twoloop is in the DB, the user can click on 'Description' to see its composition. As shown in Fig. 8, the four blocks are listed along with the GMCP-generated pseudo-component 'vardef' that contains all of the information required to use the model through the point-and-click UI (e.g., the list of input names in vardef is used to prompt the user when it becomes necessary to define the signals to be applied to the model for a simulation). The existence of links is indicated under 'Association'; a linked component can be selected and 'Disp Link' clicked to learn the name of the "home" model of that component, as shown for the component 'mylag' which is linked from the Library.

The user proceeds to use class 2 of model Twoloop to create results which may be saved in the DB if desired. This model is made active by 'configuring' it from the data base. The outcome of a sequence of analyses is shown in Fig. 9. The list of results includes one frequency-response analysis ('bodeplt' = BODE_PLOT), an eigen analysis ('cloopeig' = EIGEN_RESULTS), a discretized version of Twoloop ('discr2lp' = DABCD_MODEL), two time-histories or simulation results ('ramp2at' and 'rampdblt' = TIME_HIST), and one "Package Result" ('svd' = PKGE_RESULT, a singular-value decomposition). Observe that the result 'discr2lp' has a reference; clicking on the 'Disp Ref' button reveals that the result is also instantiated as the component Twoloopd in the DABCD model Twoloopd in project Tampa. Finally, the condition specification for the result 'rampdblt' may be revealed by clicking 'Disp Cspec'; the outcome is:

```
input Ref ramp 2.00E+00 5.00E-01 2.50E+00 1.00E+00
input Dist doublet 5.00E-01 2.00E+00 2.00E+00 0.00
simu  8.00E+00 5.00E-02
```

which shows the input definitions Ref = ramp input, Dist = doublet, followed by the simulate command.

The status of models in project Tampa at the end of the scenario is depicted in Fig. 10. In addition to Twoloop, there are its discrete-time version just mentioned, as well as the one-block version of Twoloop called Mono2lp. Observe that there is a Note attached to Mono2lp; selecting Mono2lp and clicking on 'D/A/E Notes' (display/add/edit notes) reveals the contents:

```
-------------------------------------
-- Model Notes
-------------------------------------
-- Project:      tampa
-- Model:        mono21p
-------------------------------------
-- Date:         Sat Sep 16 12:12:04 1989
This is a one-block version of twoloop:2
```

All lines in the note file were generated automatically by the GMCP except for the last ("This is a one-block version of Twoloop:2"); these were inserted in the note file as an aid to documentation.

7. CONCLUSION

The first phase of the GE MEAD Project is nearly completed. Version 1 of the GMCP (GMCP-1.0) is based on the user interface, data-base manager, and expert system shell from the USAF MEAD Project [7], a new data-driven supervisor, and PRO-MATLAB and ACSL or SIMNON. This software is now in final integration, test, and evaluation.

GMCP-1.0 represents a new, more supportive environment for computer-aided control engineering (CACE). The most important novel features are an *integrated engineering data-base manager,* a *built-in expert system shell,* and a *flexible user-friendly user interface* including a "point-and-click' interactive mode, two command modes (GE MEAD and Package), and a Macro Facility. Another notable attribute of the GMCP is the ability to use the core packages directly without sacrificing the benefits of data-base management.

In terms of CACE functionality, GMCP-1.0 is a basic CACE package for control system analysis and design, and much functionality is not particularly "fancy". The higher-level functionality of PRO-MATLAB is available through the most user-friendly access mode; all lower-level primitives may be used via Package Mode or by writing GMCP Macros in package command mode.

In terms of "value added", we believe that the GMCP data-base manager with version control and integrity maintenance will provide the largest benefits in comparison with CACE packages lacking such support. The user interface with its widely different interaction modes should extend the user group from the set of expert users that now enjoy the power of modern linear analysis and design packages and nonlinear simulation environments to the many control engineers that find the learning curve to be too steep to be worth the effort. We have tried to satisfy expert users as well, by the inclusion of Command Modes and the Macro Facility to allow the flexibility and extensibility demanded by the more advanced user. It is our hope that every GMCP user will graduate to that exalted status! Finally, the use of expert aiding is still in the experimental stage of development. We anticipate that the ability to deliver meaningful rule bases under the GMCP will move this technology from research to real use in practical applications.

A number of extensions and refinements are planned, including the incorporation of a more flexible general-purpose model-building environment, improved user interface features, and additional expert aiding. In addition, we are presently porting the GMCP to a window-based workstation, to further improve the flexibility of the user interface. In the longer term, we plan to broaden the scope of the GMCP by adding packages that support other aspects of CACE, such as automatic code generation for the controller, and modeling, simulation, and analysis of structures.

Acknowledgements: The work described above has taken advantage of many of the results of the USAF MEAD Project [7]. Invaluable programming support is being provided by Aule-Tek Inc. personnel; the contributions of the following individuals are most gratefully recognized: David Kassover, Jim Trojan, Mike Charbonneau, and Al Antoniotti.

REFERENCES

[1] Spang, H. A., III, "The Federated Computer-Aided Control Design System", *Proc. Second IFAC Symp. CAD of Multivariable Technological Systems,* West Lafayette, Indiana, September 1982; also *IEEE Proceedings, Vol. 72,* 1724-1731, December 1984.

[2] Taylor, J. H. and Frederick, D. K., "An Expert System Architecture for Computer-Aided Control Engineering", *IEEE Proceedings, Vol. 72,* 1795-1805, December 1984.

[3] Taylor, J. H., "Computer-Aided Control Engineering Environment for Nonlinear Systems", *Proc. 3rd IFAC Symposium on CAD in Control and Engineering Systems,* Lyngby, Denmark, August 1985.

[4] James, J. R., Frederick, D. K., and Taylor, J. H., "On the Application of Expert Systems Programming Techniques to the Design of Lead/Lag Precompensators", *Proc. Control '85,* Cambridge, UK; also in *IEE Proceedings D: Control Theory and Applications,* May, 1987.

[5] Taylor, J. H., Nieh, K-H, and Mroz, P. A., "A Data-Base Management Scheme for Computer-Aided Control Engineering", *Proc. American Control Conference,* Atlanta, GA, June 1988.

[6] Taylor, J. H. "Expert-Aided Environments for CAE of Control Systems", *Proc. 4th IFAC Symposium on CAD in Control Systems (CADCS '88),* Beijing, PR China, August 1988.

[7] Taylor, J. H., and McKeehen, P. D., "A Computer-Aided Control Engineering Environment for Multi-Disciplinary Expert-Aided Analysis and Design (MEAD)", *Proceedings of the National Aerospace and Electronics Conference (NAECON),* Dayton, OH, May 1989.

[8] Rimvall, M., Sutherland, H., Taylor, J. H., and Lohr, P. J., "GE's MEAD User Interface - a Flexible Menu- and Forms-Driven Interface for Engineering Applications", *Proc. CACSD '89,* Tampa FL, December 1989.

[9] Taylor, J. H., Frederick, D. K., Rimvall, C. M., and Sutherland, H. A., "A Computer-Aided Control Engineering Environment with Expert Aiding and Data-Base Management", submitted to *Proc. 10th IFAC World Congress,* Tallinn, USSR, August 1990.

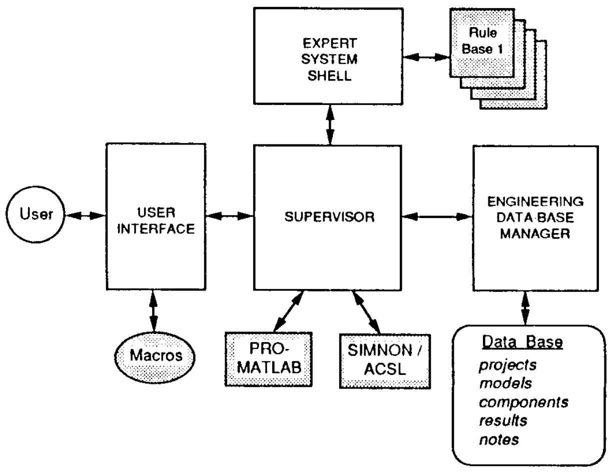


Figure 1. GMCP Architecture

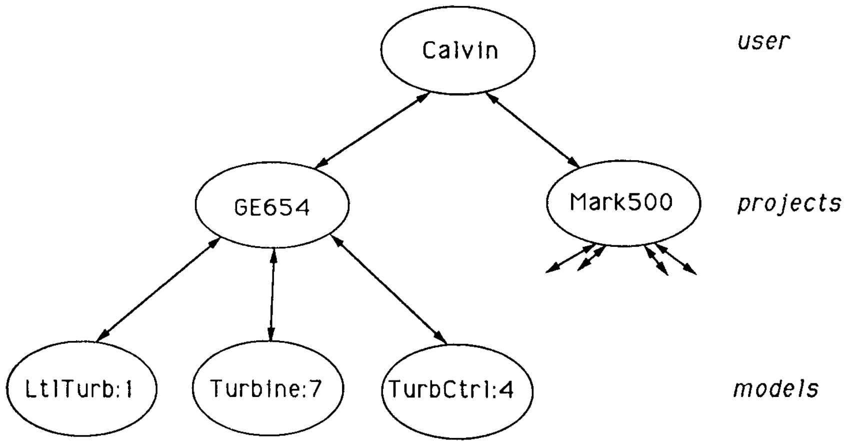


Figure 2. Basic GMCP Data-Base Hierarchy

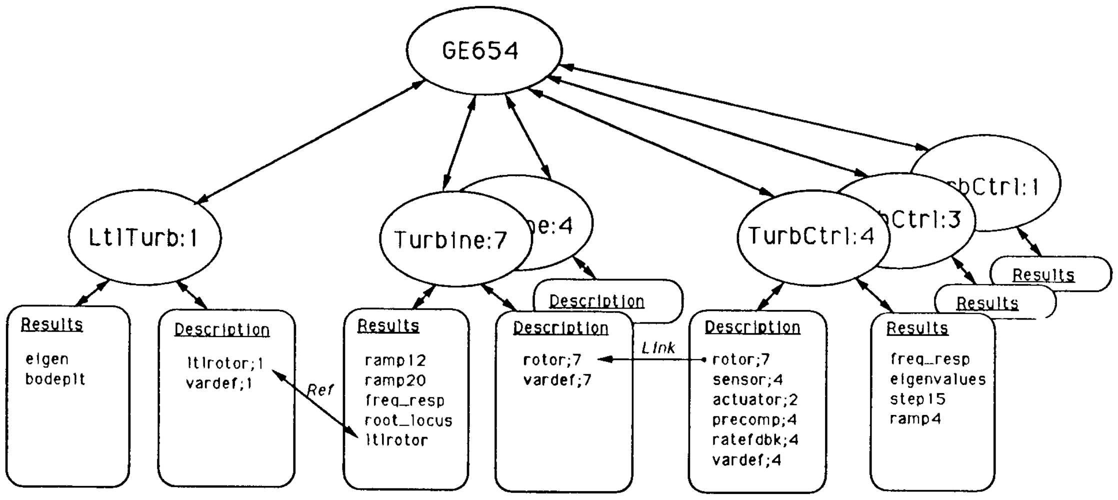


Figure 3. Detailed Data-Base Element Relations

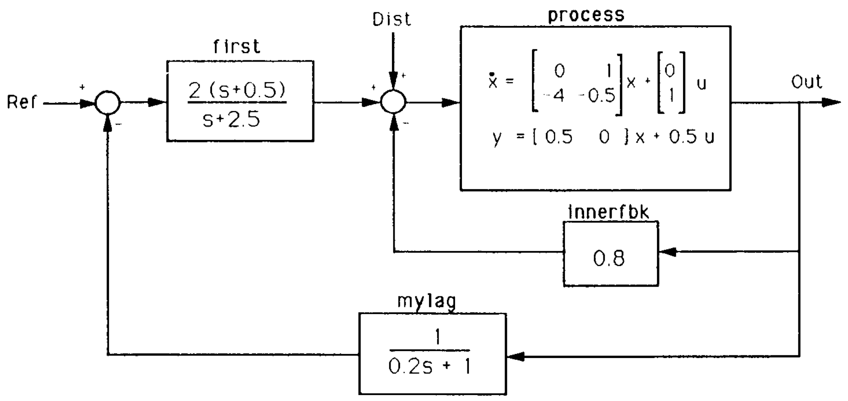
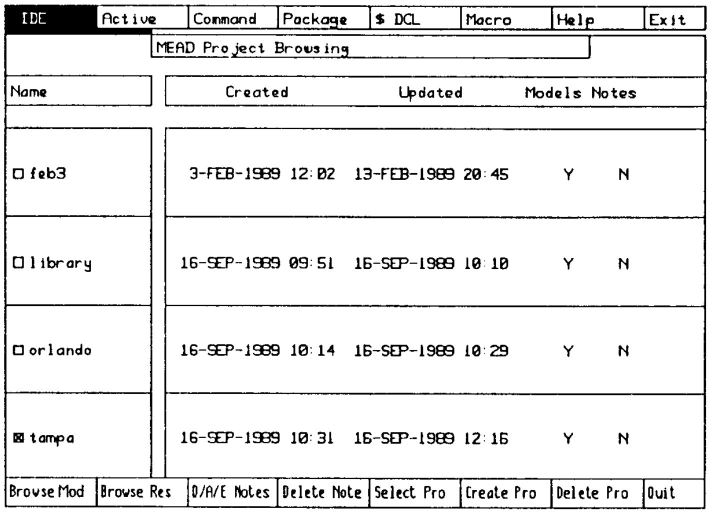
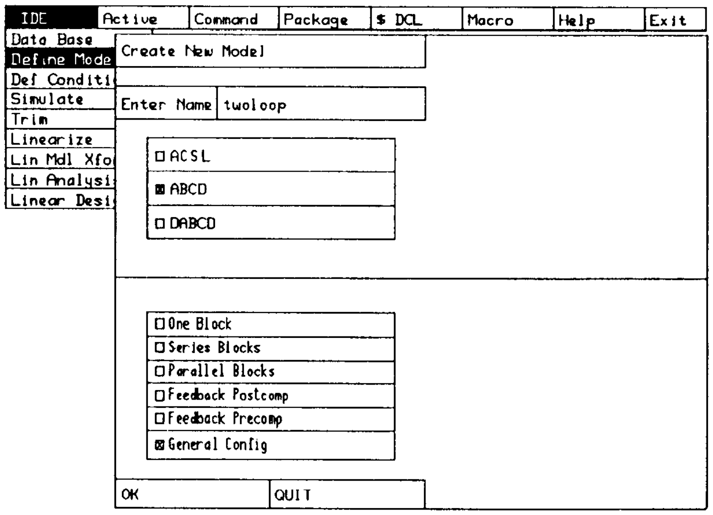

Ref
first
$\frac{2(s+0.5)}{s+2.5}$
Dist
process
$\dot{x} = \begin{bmatrix} 0 & 1 \\ -4 & -0.5 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$
$y = [\,0.5 \quad 0\,]\, x + 0.5\, u$
Out
innerfbk
0.8
mylag
$\frac{1}{0.2s+1}$

Figure 4. Example 'Twoloop' Block Diagram



Name	Created	Updated	Models	Notes
☐ feb3	3-FEB-1989 12:02	13-FEB-1989 20:45	Y	N
☐ library	16-SEP-1989 09:51	16-SEP-1989 10:10	Y	N
☐ orlando	16-SEP-1989 10:14	16-SEP-1989 10:29	Y	N
☒ tampa	16-SEP-1989 10:31	16-SEP-1989 12:16	Y	N

Figure 5. GMCP Project Browsing Screen



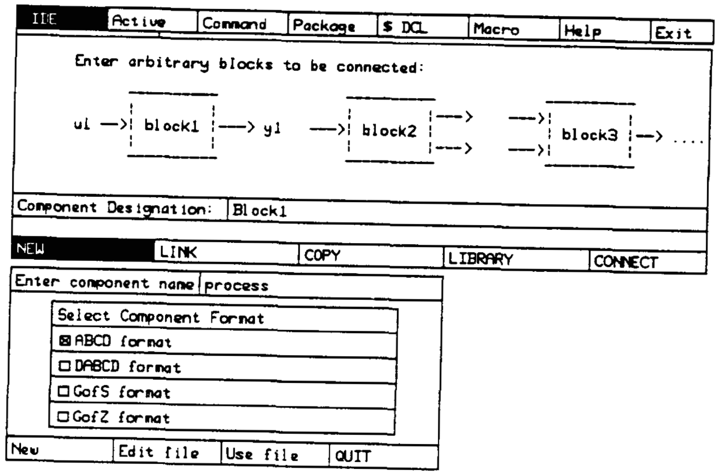
Create New Model

Enter Name twoloop

☐ ACSL
☒ ABCD
☐ DABCD

☐ One Block
☐ Series Blocks
☐ Parallel Blocks
☐ Feedback Postcomp
☐ Feedback Precomp
☒ General Config

Figure 6. GMCP Screen for Defining Models



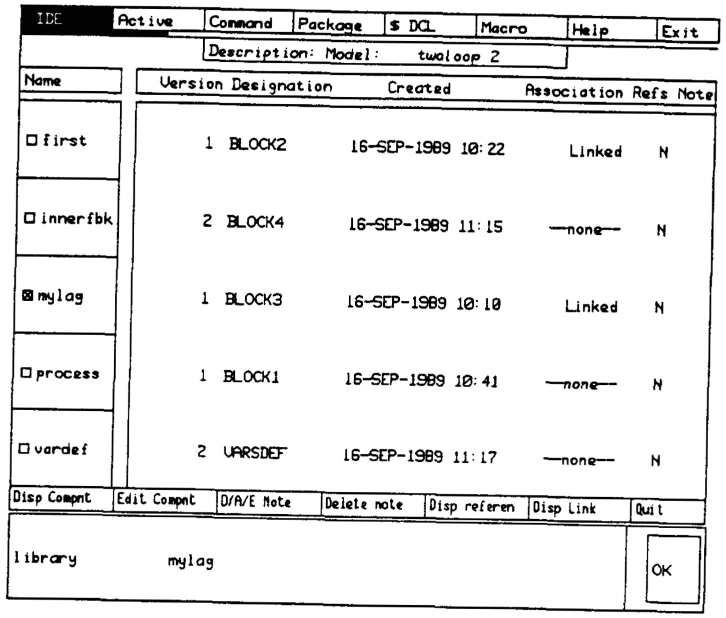
Enter arbitrary blocks to be connected:

Component Designation: Block1

Enter component name process

Select Component Format
- ☒ ABCD format
- ☐ DABCD format
- ☐ GofS format
- ☐ GofZ format

Figure 7. GMCP Screen for Defining Components

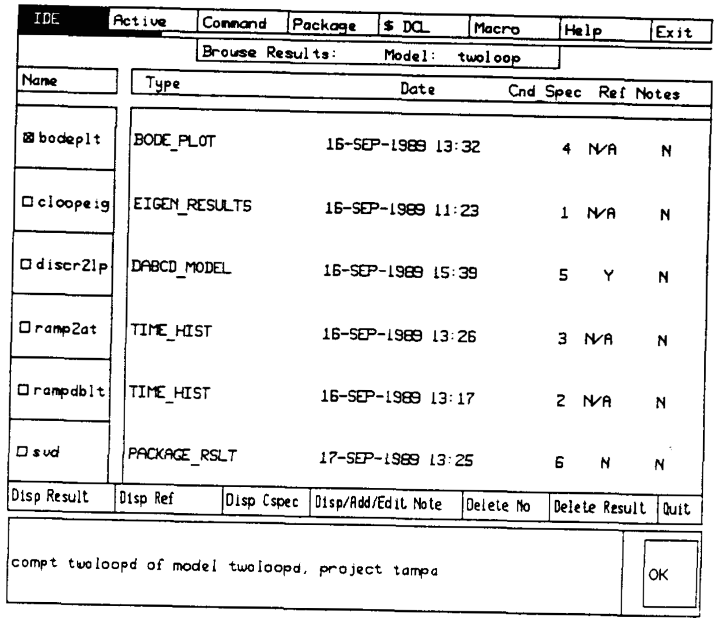


Description: Model: twoloop 2

Name	Version	Designation	Created	Association	Refs	Note
☐ first	1	BLOCK2	16-SEP-1989 10:22	Linked		N
☐ innerfbk	2	BLOCK4	16-SEP-1989 11:15	—none—		N
☒ mylag	1	BLOCK3	16-SEP-1989 10:10	Linked		N
☐ process	1	BLOCK1	16-SEP-1989 10:41	—none—		N
☐ vardef	2	VARSDEF	16-SEP-1989 11:17	—none—		N

library mylag

Figure 8. GMCP Screen for Description Browsing

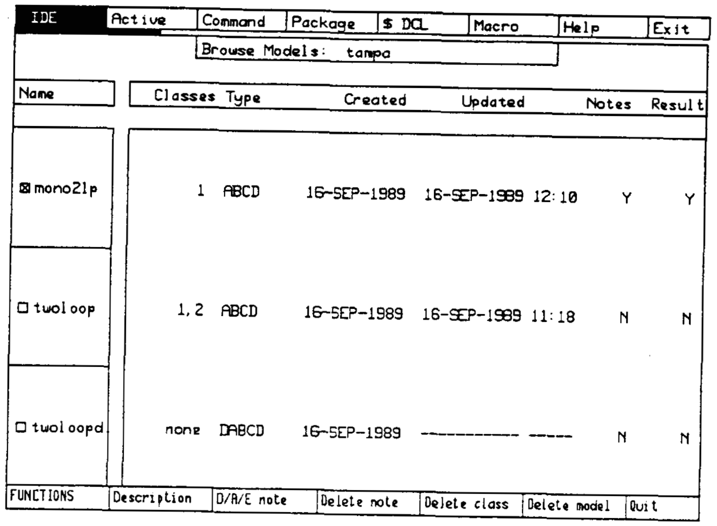


Browse Results: Model: twoloop

Name	Type	Date	Cnd Spec	Ref	Notes
☒ bodeplt	BODE_PLOT	16-SEP-1989 13:32	4	N/A	N
☐ cloopeig	EIGEN_RESULTS	16-SEP-1989 11:23	1	N/A	N
☐ discr2lp	DABCD_MODEL	16-SEP-1989 15:39	5	Y	N
☐ ramp2at	TIME_HIST	16-SEP-1989 13:26	3	N/A	N
☐ rampdblt	TIME_HIST	16-SEP-1989 13:17	2	N/A	N
☐ svd	PACKAGE_RSLT	17-SEP-1989 13:25	6	N	N

compt twoloopd of model twoloopd, project tampa

Figure 9. GMCP Screen for Results Browsing



Browse Models: tampa

Name	Classes	Type	Created	Updated	Notes	Result
☒ mono2lp	1	ABCD	16-SEP-1989	16-SEP-1989 12:10	Y	Y
☐ twoloop	1,2	ABCD	16-SEP-1989	16-SEP-1989 11:18	N	N
☐ twoloopd	none	DABCD	16-SEP-1989	———————— ————	N	N

Figure 10. GMCP Screen for Model Browsing