Real-time CAD of Control Systems
Achieving Cooperation of Modeling and Design of Controllers

Masanobu Koga, Hiroaki Toriumi and Mitsuji Sampei

Department of mechanical and Environmental Informatics
Tokyo Institute of Technology
2-12-1, Oh-okayama, Meguro-ku, Tokyo 152, JAPAN
koga@mei.titech.ac.jp, toriumi@mei.titech.ac.jp and sampei@mei.titech.ac.jp

Abstract

In this paper development of a CAD of control systems which realizes coordination of modeling of plant and design of controller is introduced. It enables us to do not only analysis of control systems and design of controllers but also on-line modeling of plants in the same environment. By utilizing this software, the control engineer is able to repeat the procedure of design of control systems efficiently to archive the best performance, which depends on the controller as well as the model of the plant. It is also shown how to improve the speed of computation by introducing the technique of memory management for real-time control and how to change the parameters of the controller without stopping real-time control.

1. Introduction

Many computer aided control system design (CACSD) softwares have been developed in universities and company [1, 2, 3], while the control system theory has made big progress, and it is going to be able to achieve the good performance if only the good model is given. The procedure of design of control systems by these softwares is shown in Fig. 1. Those softwares treat the control system design as the procedure in which the design specification is satisfied by repeating the analysis (stability and performance) of plant and control system (=plant + controller), whose mathematical model has been already determined by input-output data or physical formula, and synthesis of controller. The program for the experiment is supposed to be implemented by using another software in a different environment after its performance is verified in the simulation. It is apparent that modeling of plant, synthesis of controller, and real-time implementation of the control system are separated.

When we use a controller which is designed with the nominal model, the simulation yields good result, however, if we apply it to the real plant, the performance of the closed-loop system may be worse than estimated. This is because the estimated error of the response of the closed-loop system depends on the controller as well as the precision of the model of real plant. Therefore the modeling and control problems are not separable and are necessarily iterative in order to achieve the best performance [4].

In this paper we treat the design of control system as the procedure in which the design specification is satisfied by iterating not only analysis of plant and synthesis of controller but also modeling of plant. We develop a new software which supports comprehensively the proposed design procedure, illustrated in Fig. 2, which achieves cooperation of modeling and design of controller.

There are many methods for modeling (identification) of plant [5]. The modeling based on input-output data is indispensable in order to consider the robustness to parameter variation and to deal with uncertainty of plant. And the real-time implementation facility, which is considered in Chapter 2, is necessary to obtain the input-output data of the real plant. Therefore archiving cooperation of modeling of plant and design of controller requires the integration of real-time implementation facility to CACSD software. This paper deals with some problems for integration of real-time implementation facility to CACSD software and gives a solution to them. The reason why the conventional CACSD software excluded the modeling of plant from the iteration of procedure of control-system design may be that it was difficult to support real-time implementation facility by themselves.

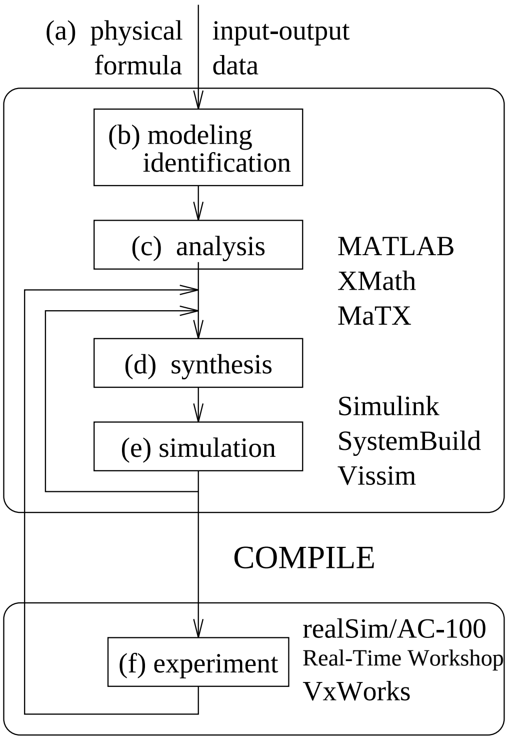

Figure 1: Procedure of design of control systems

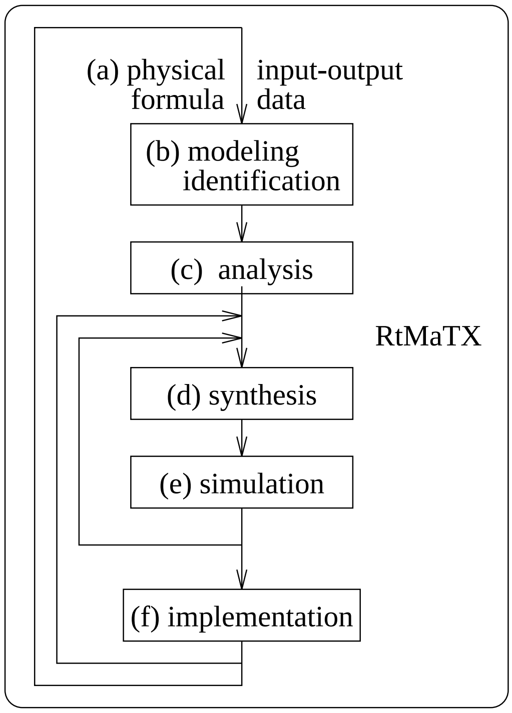

Figure 2: Procedure of design of control systems (new)

2. Real-time Implementation

2.1. Software for real-time implementation

There are a lot of softwares for real-time implementation:

- Real-Time Workshop (Mathworks Inc.)
- Realsim/AC-100 (Integrated Systems)
- Vissim (Visual Solutions)
- VxWorks (Wind River Systems)
- RCOS, MRCOS (Tokyo Institute of Tech.)
- ...

Since RCOS (Real-time Control Operating System) is very simple, it can be executed on many personal computers, even on the computer with a 16-bit CPU 8086. Vxworks supports network facility and provides very useful environment for users. Since RCOS and Vxworks don't provides any tools for matrix calculation, it takes much time to write control programs for complex control algorithm. Vissim enables us to write the controller by using block diagram interface, however, the speed of computation is slow since it is an interpreter. Real-time Workshop and Realsim/AC-100 generate real-time control programs directly from block diagram of the control system, and provide matrix calculation library. They are excellent softwares for real-time implementation. Since we have to recompile the generated code for real-time control if we change the parameters of the controller or at least the structure of the controller, the iteration of control system design proposed in this paper can not be done efficiently.

2.2. CAD for real-time implementation

In this work we developed a CAD (RTMATX) for real-time implementation, which provides real-time implementation facility as well as facility of analysis and synthesis of control system. We integrated real-time implementation facility to CACSD software MATX [6, 7, 8], which allows the definition and manipulation of matrices in a mathematical algebraic fashion. Since RTMATX programs can be compiled, the speed of computation is as fast as C programs.

When we use the software above for real-time implementation, we have to stop real-time control in order to change parameters of controller. If the real-time control is not stopped, the incorrect control output may be calculated since some parameters are updated and others are not. RTMATX allows changing parameters of controller without stopping real-time control. This facility makes it possible to iterate the procedure of control system design efficiently.

3. Memory Management for real-time implementation

3.1. Memory management of MATX

If we write a program with matrix notation, the program is plain to read, however, the size of memory for matrix calculation is not determined until the execution time. This is why the function like `malloc()`, which allocates the memory dynamically, and the function like `free()`, which releases the allocated memory, are necessary. These functions are slow since they are general-purpose memory management functions. MATX has its own memory management mechanism to reduce the number of calls of `malloc()` and `free()`. It is equipped with some memory management list for memory allocation. When a block of memory is required, the memory management list is referred to obtain a suitable size of memory. If there exists such a memory block in the list, it is used. Otherwise `malloc()` is used to get the memory from OS.

When a block of memory becomes unnecessary, it is registered into the list instead of being returned to OS by `free()`. In consequence a necessary amount of memory which is required for a series of calculation is registered into the memory management list. When the same series of calculation is executed, no more `malloc()` is necessary. This strategy of memory allocation is significantly efficient for the program like simulation which repeats the same calculations, because the function `malloc()` is necessary only at the first loop.

State transition of a memory block: The memory management of MATX mentioned above is carried out based on the idea of state transition of memory shown in Fig. 3. The memory takes the following four states.

1. `SYS` state (Managed by OS)
2. `TMP` state (Obtained from OS by `malloc()`)
3. `VAR` state (Assigned to the variable)
4. `REV` state (Registered into the memory management list)

Example: We illustrate the state transition of memory by using the following example program shown in Fig. 4. The state of memory before the program is started is shown in Fig. 5(a), where `a`, `b`, ..., and `k` denote the memory block of a 2×2 matrix.

When the function `afo()` is called and the right-hand side of Eq. (1) is evaluated, the state of memory becomes as shown in Fig. 5(b), and after the assignment of Eq. (1), as show in Fig. 5(c). Similarly after assignment of Eq. (2) and Eq. (3), the state of memory becomes as shown in Fig. 5(d). When the right-hand side of Eq. (4) is evaluated, as shown in Fig. 5(e), where `d`, `e`, and `f` have the result of $(A + B)$, $(A - B)$, and $(A + B) * (A - B)$, respectively. After assignment of Eq. (4), as shown in Fig. 5(f). The memory of variable `f`, which has the result of $(A+B)*(A-B)$, moves to state `VAR` and the memory of `d` and `e` move to state `REV`. When the right-hand side of Eq. (5) is evaluated, the state of memory is shown in Fig. 5(g), where `d`, `e`, and `g` have the result of $(C + D)$, $(C-D)$, and $(C+D)*(C-D)$, respectively. The memory of `d` and `e` move from state `REV` to state `TMP`, while the memory of `g` moves from state `SYS` to state `TMP`. After assignment of Eq. (5), the state of memory becomes as shown in Fig. 5(h). The memory of variable `g`, which has the result of $(C + D) * (C - D)$, moves to state `VAR`, and the memory of `d` and `e` move to state `REV`.

When the function finishes, the state of memory becomes as shown in Fig. 5(i). All memory in state `VAR` move to state `TMP`. After the value of function is assigned to a variable, all memory in state `TMP` move to state `REV`, and the state of memory is shown in Fig. 5(j). When the function `afo()` is called again, all memory is obtained from the memory management list, and no more `malloc()` is necessary.

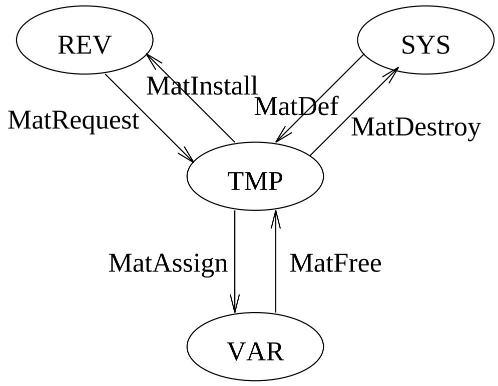

Figure 3: State transition of memory block

```
Func Matrix afo()
{
  Matrix A, B, C, D, E;

  A = [[1 2][3 4]];        // (1)
  B = [[5 6][7 8]];        // (2)
  C = A + B;               // (3)
  D = (A + B) * (A - B); // (4)
  E = (C + D) * (C - D); // (5)
  return E;                // (6)
}
```

Figure 4: Program of matrix calculation

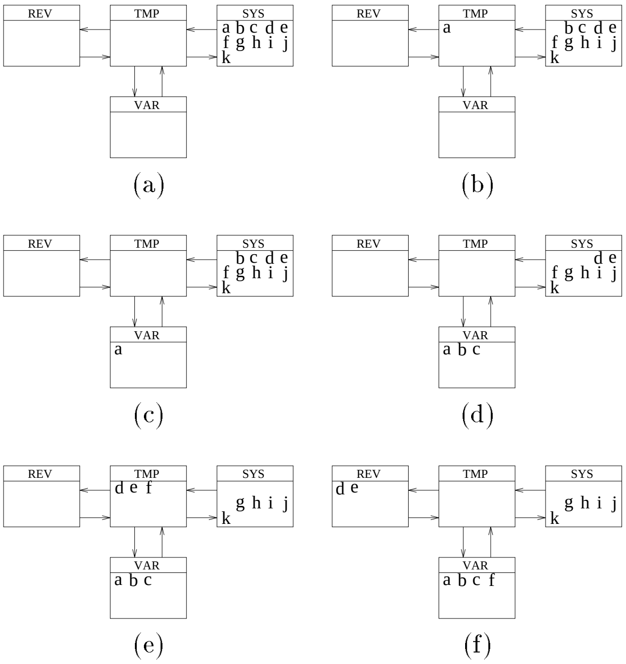

Figure 5: An example of state transition of memory

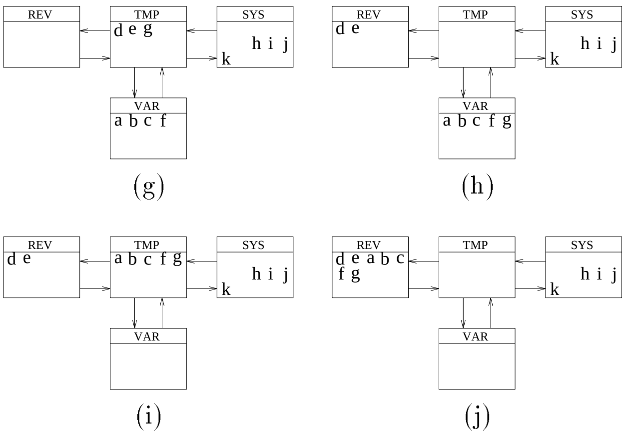

Figure 5: An example of state transition of memory

3.2. Memory management of RTMATX

As mentioned above, the function `malloc()` is unsuitable for the real-time control program. Accordingly RTMATX calls the function for real-time control once in advance before starting the real-time control. Then the necessary amount of memory which is required for the calculation of real-time control is registered in the memory management list as a result of the memory management of MATX. Therefore no more `malloc()` is called when the function for real-time control is executed every sampling period. It is possible to restrain the unnecessary control output such as D/A output by checking whether the program is running at the rehearsal mode or real-time control mode.

4. How to update parameter in real-time

We can utilize two tables of parameters in order to share parameters of control systems between non-real-time functions and real-time functions [9]. As shown

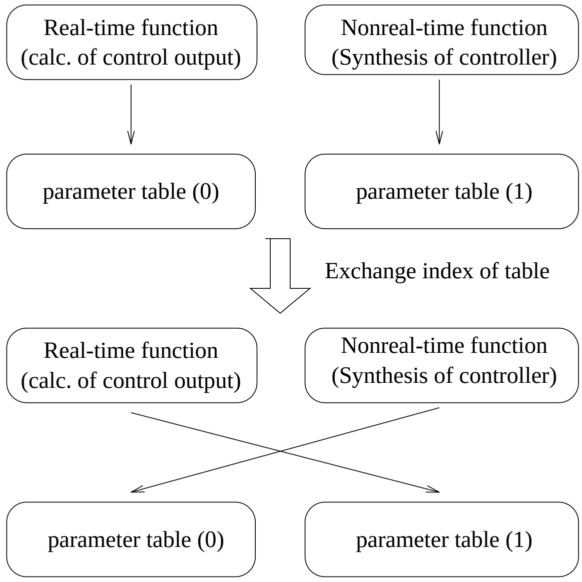

Figure 6: Exchange of parameter table

in Fig. 6 a real-time function refers to parameters of one table for calculation of control output, while a non-real-time function changes parameters of another table. After changing all parameters of table, two parameter tables are exchanged. Accordingly we can change parameters, which is shared between real-time functions and non-real-time functions, with consistency of all parameters.

4.1. Real-time variable

If we use the method proposed in [9], the table of parameters must be maintained by programmers. RTMATX introduces idea of **real-time variable** in order that the shared parameters can be changed without taking care of parameter table. If we declare the variables, which should be shared between real-time and non-real-time functions, as **real-time variables**, two parameter tables including these variables are prepared automatically, and the consistency of all parameters in the tables is guaranteed. Therefore we don't have to take care of parameter table to update the shared parameters.

We consider two implementations for real-time variable, i.e., double-array scheme and double-function scheme.

How RTMATX treats real-time variables and generates C language code is described below.

4.2. Double-array scheme

The double-array scheme prepares two arrays for parameter table and exchanges the tables by changing the index of arrays. A reference to real-time variable in the program corresponds to a reference to an element of an array in C program. It takes a little extra time for the double-array scheme to calculate the index of an array to access a real-time variable.

Non-real-time function:

- An index of array is determined for each real-time variable.
- Real-time variables can be accessed and printed as the original name without its index.
- When the function to exchange the index is called, the updated variables become valid in real-time functions.

Real-time function:

- An index of array is determined for each real-time variable.
- Assignment to real-time variables are inhibited.

4.3. Double-function scheme

The double-function scheme prepares two functions for real-time control and exchanges the parameter tables by exchanging functions. The speed of computation is a little bit faster than double-array scheme since the calculation of the index of an array to access a real-time variable is not necessary, however, the size of program is bigger than that.

Non-real-time function

- Real-time variables are distinguished by the index of real-time function.
- Real-time variables can be accessed and printed as the original name without its index.
- When the function to exchange the index is called, the updated variables become valid in real-time functions.

Real-time function

- Two real-time functions with different names are generated and independent real-time variables are declared in each function.
- The speed of computation is a little bit faster than double-array scheme since the calculation of the index of an array to access a real-time variable is not necessary.
- Assignment to real-time variables are inhibited.

5. Real-time control of inverted pendulum

We deal with the real-time implementation of the inverted pendulum shown in Fig. 7 to illustrate the facility of RTMATX.

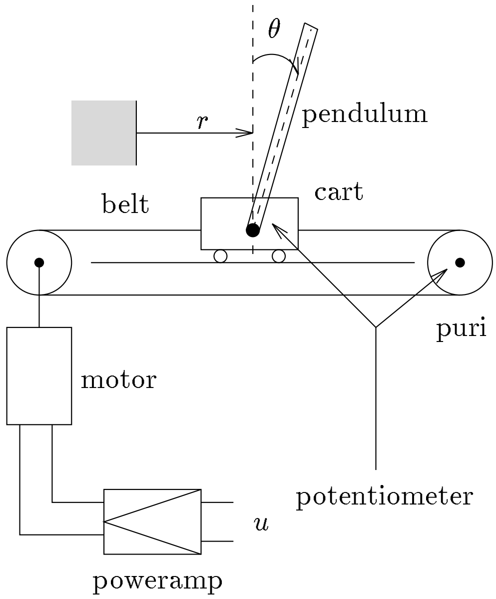

Figure 7: Inverted pendulum

Consider the linearized model of the plant:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}$$

We design an LQ state feedback control law

$$u = -Fx$$

and an observer

$$\begin{aligned} \dot{z} &= A_h z + B_h y + J_h u \\ \hat{x} &= C_h z + D_h y \end{aligned}$$

5.1. Real-time control program

Figure 8 shows the main function of the program.

- Sampling period is set to `stime`.
- `break_task()` is invoked when the interrupt key is hit, in which we describe the statement for stopping real-time control.
- Once `rtStart()` is called, `on_task()` is invoked every sampling period.
- In the function `off_task_loop()` there are statements to display variables and to get the command from the user. When the user changes the weight

matrices of LQ or the poles of the observer, the state feedback control law and observer gain matrices are calculated, and real-time variables `F`, `Ah`, `Bh`, `Ch`, `Dh`, and `Jh` are updated.

- After `off_task_loop()` the function `rtStop()` is called and real-time control is stopped.

```
Func void main()
{
  para_init();
  var_init();
  machine_ready();

  rtSetClock(stime);
  rtSetTask(on_task);
  rtSetBreak(break_task);

  rtStart();
  off_task_loop();
  rtStop();

  machine_stop();
}
```

Figure 8: Main function of real-time program

Figure 9 shows the function for real-time control of the program.

- Since the variables `Ah`, `Bh`, `Ch`, `Dh`, `Jh`, and `F` are declared as real-time variable, they can be updated while the real-time control is running.
- The function `sensor()` returns the output of plant measured by sensors.
- `actuator()` operates the actuator.

```
Realtime Matrix Ah,Bh,Ch,Dh,Jh,F;

Func void on_task()
{
  Matrix y, xh;

  y  = sensor();

  xh = Ch*z + Dh*Y;
  u  = - F*xh;
  z = Ah*z + Bh*y + Jh*u;

  actuator(u);
}
```

Figure 9: Real-time function of real-time program

6. Summary

In this paper we considered the design of control system as the procedure in which the design specification is satisfied by iterating not only analysis of plant and synthesis of controller but also modeling of plant. We developed a software which supports comprehensively the proposed design procedure, which achieves cooperation of modeling and design of controller.

Achieving cooperation of modeling of plant and synthesis of controller requires the integration of real-time control facility to CACSD software, since it is necessary to obtain the input-output data of the real plant. This paper dealt with some problems for integration of real-time facility to CACSD software and gives a solution to them.

There is a control technique, called **auto-tuning**, in which the transfer-function of a plant is identified while keeping on controlling and the controller is synthesized with the new model of plant and the control continues with the new controller. The CACSD software with real-time facility developed in this work will be very useful for auto-tuning, since the technique iterates modeling and synthesis of controller while controlling.

References

[1] Cleve Moler. *MATLAB – User's Guide*. Alberquerque, USA, 1980.

[2] Robert Walker, Charles Gregory Jr., and Sunil Shah. MATRIXx: A data analysis, system identification, control design and simulation package. *Control Systems Magazine*, pp. 30–37, 1982.

[3] C. Magnus Rimvall. Computer-aided control system design. *IEEE Control Systems*, Vol. 13, No. 2, pp. 14–16, 1993.

[4] Robert E. Skelton. *Dynamic Systems Control*. John Wiley & Sons, 1988.

[5] Lennart Ljung. *System Identification Theory for the User*. Prentice Hall, 1987.

[6] Masanobu Koga and Katsuhisa Furuta. MaTX : A high-performance programming language (interpreter and compiler) for scientific and engineering computation. *Proc. of CACSD'92, Napa, U.S.A.*, pp. 15–22, 1992.

[7] Masanobu Koga and Katsuhisa Furuta. Programming language MaTX for scientific and engineering computation. In Derek A. Linkens, editor, *CAD for Control Systems*, chapter 12, pp. 287–317. Marcel Dekker, Inc., July 1993.

[8] Masanobu Koga, Mitsuji Sampei, and Katsuhisa Furuta. A compiler of matlab to MaTX: Compiling and linking of m-files to an executable program. *Proc. of CACSD'94, Tucson, Arizona, U.S.A.*, pp. 137–142, 1994.

[9] Suzumu Sekino. Integration of real-time kernel to cad. Bachelor thesis, Tokyo Institute of Technology, Department of Control Engineering, 1995.