AN INTEGRATED SOFTWARE ENVIRONMENT FOR DESIGN AND REAL-TIME IMPLEMENTATION OF CONTROL SYSTEMS

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
Tel: +81-3-5734-2328; Fax: +81-3-5734-2328
E-mail: koga@mei.titech.ac.jp

Abstract: In this paper development of a CAD of control systems is introduced which enables us to do not only analysis of control systems, design of controllers but also real-time implementation of controllers. By utilizing this software, the control engineer is able to repeat the procedure of modification of controllers and experiments without recompile to attain better performance. The software also offers the facility to update the parameters of controllers without stopping real-time control, which helps on-line tuning of controllers. If some parameters of the controller is changed on-line, the control input may change discontinuously. It has serious effect on the control systems. A method for on-line tuning of state feedback controller with state observer is proposed and verified through the experiment with an inverted pendulum.

Key Words: Computer-aided control system design, Real-time systems, Simulation

1. INTRODUCTION

In this paper, an integrated software environment for design and real-time implementation of control systems is introduced. It enables us to do not only off-line design but also on-line design. In other words, we can do analysis of plant, synthesis of controller, modeling of plant, and real-time implementation for the experiment, all in one software environment. By utilizing this software, it is possible to repeat the cycle of design of control systems efficiently to achieve the good performance of the closed-loop system. We consider the procedure of design of control systems from software environment point of view in Section 2.

The software also offers the facility to update the parameters of controllers without stopping real-time control, which helps on-line tuning of controllers. If some parameters of the controller are changed on-line, the control input may change discontinuously. It may have serious effect on the control systems. A method for on-line tuning of controller is considered in Section 3.

The system has been used in research and education, and has reduced the implementation time considerably, e.g. when developing new laboratory exercises, or when a control algorithm is tested in a laboratory experiment. A few examples are described in Section 4.

2. PROCEDURE OF DESIGN OF CONTROL SYSTEMS

Many computer aided control system design (CACSD) softwares have been developed in universities and companies. The typical procedure of design of control systems supported by these softwares is shown in Fig 1.

First, analysis of plant, whose mathematical model has been determined by input-output data or physical formula, is done. Then, synthesis of controller, simulation, and experiment are repeated until the design specification is satisfied. After the performance of the closed-loop system is verified in simulation, the program for the experiment is implemented in another software environment. It is apparent that design of control system and real-time implementation are separated. In

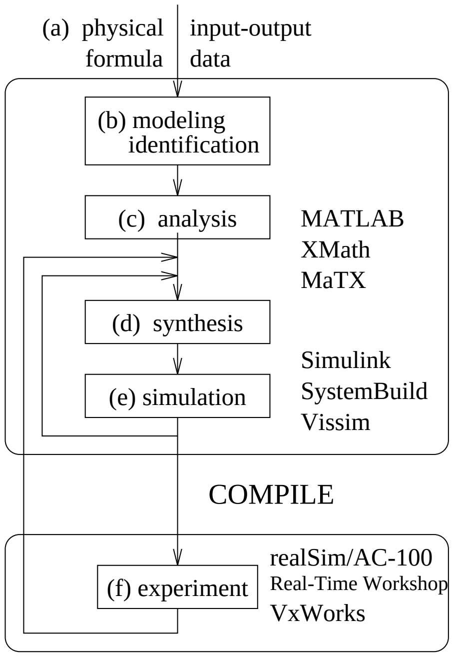

Fig. 1. Conventional design cycle of control systems

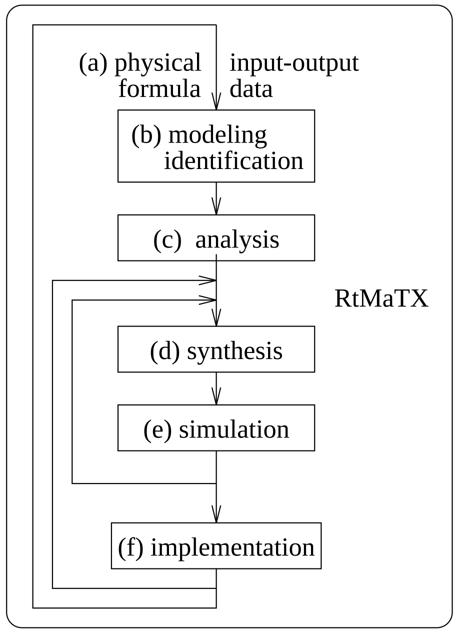

Fig. 2. New design cycle of control systems

this paper, we consider the procedure of design of control systems, in which the design specification is satisfied by iterating analysis of plant, synthesis of controller, simulation, experiment, and modeling of plant. For this purpose, we developed a new software environment which supports comprehensively the proposed design cycle. Figure 2 shows the proposed design cycle of control systems.

There are many methods for modeling (identification) of plant. Among them, the modeling based on the input-output data is indispensable to treat the robustness to parameter variation and uncertainty of plant. This leads us to requirement of real-time implementation facility. Therefore we integrated the real-time implementation facility to CACSD software. This paper deals with some problems to integration of real-time implementation facility to CACSD software and gives a solution to them.

3. REAL-TIME IMPLEMENTATION ISSUE

We consider some issues to integration of real-time implementation facility to CACSD software and give a solution to them by RtMaTX.

3.1 *Speed of Computation*

Speed of computation is a crucial point for real-time implementation. Therefore the program should rather be compiled than be executed by the interpreter.

RtMaTX programs can be compiled to be as fast as C programs (Koga *et al.*, 1996).

3.2 *Tuning of Controller without recompile*

There some commercially available CACSD softwares which support real-time implementation in some way (Integrated Systems, 1991; Inc., 1992). When we use those softwares, we have to recompile the generated code for the experiment if we change the parameters of the controller or at least the structure of the controller. This makes the iteration of the procedure of control system design inefficient.

Since RtMaTX supports all steps of the design cycle, the iteration of the procedure can be done without recompile.

3.3 *Memory management for real-time implementation*

If we write a program with matrix notation like Matlab, the program is easy to read, however, the size of memory for calculation is not determined until the execution time. This is why the function like `malloc()`, are necessary. These functions are slow since they are general-purpose memory management functions. Therefore we have to consider the method to improve the speed of computation.

MaTX has its own memory management mechanism to reduce the number of calls of memory allocation function (Koga *et al.*, 1996). In particular, no memory allocation function is called when real-time control is executed.

Memory management of MATX

When a block of memory is required, the memory management list in MATX is referred. If there exists a suitable size of memory block in the list, it is used. Otherwise `malloc()` is used to get the memory from the operation system. When a block of memory becomes unnecessary, it is registered into the memory management list instead of being returned to OS. Therefore the exact size of memory for a series of calculation is registered into the memory management list. When the same calculation is executed again, no more `malloc()` is necessary. This strategy of memory management is significantly efficient for the program like simulation which repeats the same calculation, because the memory allocation function is called only once.

Memory management of RTMATX

The memory management of RTMATX is almost same as MATX. RTMATX calls the function for real-time control once in advance before starting the real-time control. In this stage, the necessary amount of memory for the calculation of real-time control is registered in the list. No memory allocation function is called when the function of real-time control is executed every sampling period.

3.4 *On-line parameter change*

In most cases, we have to stop real-time control in order to change parameters of the controller. If the real-time control is not stopped, the incorrect control input may be calculated since some parameters are updated and others are not at the moment.

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.

How to update parameter in real-time

We can utilize two tables of parameters in order to share parameters of control systems between a non-real-time function and a real-time function. As shown in Fig. 3, a real-time function refers to parameters of one table for the calculation, while a non-real-time function changes parameters of th other table. After changing all parameters of the table, two parameter tables are exchanged. Accordingly we can change parameters, which are shared between a real-time function and a non-real-time function, with consistency of all parameters.

Real-time variable

If we declare the variables, which should be shared, as **real-time variables**, two parameter

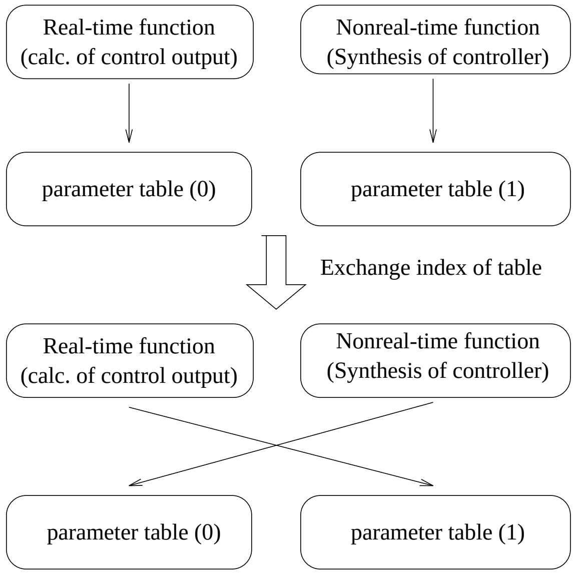

Fig. 3. Exchange of parameter tables

tables including these variables are prepared automatically. Therefore we don't have to take care of the table of the shared parameters. An example program for real-time implementation is given in Section 4.

3.5 *On-line tuning of controllers*

If some parameters of the controller are changed on-line, the control input may change discontinuously. Since it has serious effect on the control systems, the proper method for on-line changing parameters should be considered.

On-line tuning of state feedback and observer

We consider the effect of changing the parameter of the controller without stopping real-time control. The state equation and output equation of the plant are described as

$$\dot{x} = Ax + Bu \quad (1)$$

$$y = Cx \quad (2)$$

where $x \in R^n$, $u \in R^m$ and $y \in R^p$. And the system is assumed to be controllable and observable. The controller consists of state feedback

$$u = -Fx \quad (3)$$

and minimum order state observer

$$\dot{z} = \hat{A}z + \hat{B}y + \hat{J}u \quad (4)$$

$$\hat{x} = \hat{C}z + \hat{D}y. \quad (5)$$

We assume that the parameters are changed around the equilibrium point, i.e., $(x = 0, u = 0)$, at $t = t_0$.

State Feedback
We found that the effect of changing state feedback on-line is small through some simulations and experiments, since the state is changed around the equilibrium point, $x \sim 0$.

State Observer
When the parameters of the observer are changed, the estimated state $\hat{x}$ may change discontinuously and control input u may change dramatically. We consider a method to determine $z_{new}(t_0)$, the initial state of the updated observer, so that $\hat{x}_{new}(t_0)$, the estimated state of the new observer

$$\begin{aligned} \dot{z}_{new}(t) &= \hat{A}_{new} z_{new}(t) + \hat{B}_{new} y(t) + \hat{J}_{new} u(t) \quad (6) \\ \hat{x}_{new}(t) &= \hat{C}_{new} z_{new}(t) + \hat{D}_{new} y(t), \quad (7) \end{aligned}$$

is consistent with $\hat{x}_{old}(t_0)$, the estimated state of the old observer

$$\begin{aligned} \dot{z}_{old}(t) &= \hat{A}_{old} z_{old}(t) + \hat{B}_{old} y(t) + \hat{J}_{old} u(t) \quad (8) \\ \hat{x}_{old}(t) &= \hat{C}_{old} z_{old}(t) + \hat{D}_{old} y(t). \quad (9) \end{aligned}$$

We transform the system (1)-(2) with state transformation $x = T\bar{x}$ to the equivalent system

$$\begin{aligned} \dot{\bar{x}}(t) &= \bar{A}\bar{x}(t) + \bar{B}u(t) \\ y(t) &= \bar{C}\bar{x}(t), \quad C = [O \ I_p] \end{aligned}$$

Let the minimum order state observer for the equivalent system be described as

$$\begin{aligned} \dot{\bar{z}}(t) &= \hat{\bar{A}}\bar{z}(t) + \hat{\bar{B}}y(t) + \hat{\bar{J}}u(t) \quad (10) \\ \hat{\bar{x}}(t) &= \hat{\bar{C}}\bar{z}(t) + \hat{\bar{D}}y(t) \quad (11) \end{aligned}$$

where

$$\begin{bmatrix} \hat{\bar{C}} & \hat{\bar{D}} \end{bmatrix} = \begin{bmatrix} I_{n-p} & L \\ O & I_p \end{bmatrix}, \quad (12)$$

and L is a constant matrix which determines the poles of the observer. From (11) and (12), we obtain the state of the observer $\bar{z}$:

$$\bar{z}(t) = \begin{bmatrix} I_{n-p} & O \end{bmatrix} \hat{\bar{x}}(t) - Ly(t)$$

If we choose the initial state of the new observer as

$$\bar{z}_{new}(t_0) = [I_{n-p} \ \ O]\hat{\bar{x}}_{old}(t_0) - L_{new} y(t_0), \quad (13)$$

then the estimated state of the new observer is consistent with one of the old observer as follows.

$$\begin{aligned} \hat{\bar{x}}_{new}(t_0) &= \hat{\bar{C}}_{new}\bar{z}_{new}(t_0) + \hat{\bar{D}}_{new} y(t_0) \\ &= \begin{bmatrix} I_{n-p} \\ O \end{bmatrix} \bar{z}_{new}(t_0) + \begin{bmatrix} L_{new} \\ I_p \end{bmatrix} y(t_0) \\ &= \hat{\bar{x}}_{old}(t_0) \end{aligned}$$

4. EXAMPLE

We deal with some examples to illustrate the facility of RTMATX for simulation and real-time implementation. The plant is the well-known inverted pendulum shown in Fig. 4. If we choose the state

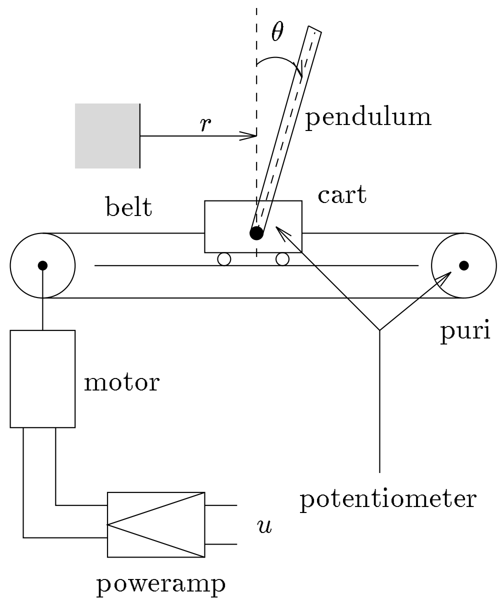

Fig. 4. Inverted pendulum

of the system as

$$x^T = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} = \begin{bmatrix} r & \theta & \dot{r} & \dot{\theta} \end{bmatrix},$$

then the state equation is described as

$$\frac{dx}{dt} = \begin{bmatrix} x_3 \\ x_4 \\ (1+\alpha \sin^2 x_2)^{-1}\{a_{32}\sin x_2 \cos x_2 + a_{33}x_3 + a_{34}\cos x_2 x_4 + a_{35}\sin x_2 x_4^2 + b_3 u\} \\ (1+\alpha \sin^2 x_2)^{-1}\{a_{42}\sin x_2 + a_{43}\cos x_2 x_3 + a_{44}x_4 + a_{45}\sin x_2 \cos x_2 x_4^2 + b_4 \cos x_2 u\} \end{bmatrix} \quad (14)$$

where a_{ij}, b_i, and α are parameters of the plant. We consider the linearized model of the plant (1)-(2) and design an LQ state feedback control law (3) and an observer (4)-(5) The observer is discretized for digital control such that

$$\begin{aligned} z[k+1] &= \hat{A}_d z[k] + \hat{B}_d y[k] + \hat{J}_d u[k] \quad (15) \\ u[k] &= -F\hat{x}[k] \quad (16) \\ \hat{x}[k] &= \hat{C}z[k] + \hat{D}y[k]. \quad (17) \end{aligned}$$

4.1 *Simulation*

The simulation program consists of three functions, `main()`, `diff_eqs()`, and `link_eqs()`. The function `diff_eqs()` is a function which calculates the derivative of state vector $\dot{x}$ and `link_eqs()` is the function which calculates control inputs

u. In main() the differential equation is integrated according to RKF45 algorithm automatically changing the step size to guarantee the specified computation error. The lists of diff_eqs() and link_eqs() are shown in List 1 and List 2. Note that the program can be written almost same as the mathematical notation. See (Koga, 1995) for the detail description.

```
Func void diff_eqs(dx, t, x, u)
  Matrix dx, x, u;
  Real t;
{
  Real c2, s2, dm;

  c2 = cos(x(2));
  s2 = sin(x(2));
  dm = 1 + alpha*s2^2;

  dx =
  [[                 x(3)              ]
   [                 x(4)              ]
   [dm~*(a32*s2*c2 + a33*x(3)
    +a34*c2*x(4)+a35*s2*x(4)^2+b3*u(1))]
   [dm~*(a42*s2 + a43*c2*x(3)+a44*x(4)
    +a45*s2*c2*x(4)^2+b4*c2*u(1)    ]];
}
```

List : Function for state equation

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

Func void link_eqs(u, t, x)
 Matrix u, x;
 Real t;
{
 Matrix y, xh;

 y  = C*x;         // output equation

 xh = Ch*xo+Dh*y; // state estimation
 u  = - F*xh;      // state feedback
 z  = Ah*z + Bh*y + Jh*u; // observer
}
```

List 2: Function for control input

4.2 *Real-Time Implementation*

The real-time implementation program consists of three functions, main function main(), on-line function on_task(), and off-line function off_task(). The function on_task() is a function for calculation of control inputs and off_task() is the user-interface function for showing and changing parameters. Once the real-time control starts, on_task() is called every sampling period. The list of on_task() is shown in List 3. The function sensor() returns the output of the the plant measured by sensors, and actuator() operates the actuator. Since the variables, Ah, Bh, Ch, Dh, Jh, F are declared as real-time variables, they can be changed while the real-time control is running. Note that the difference between function on_task() and link_eqs(), which was shown in the simulation program, are two lines. Therefore we can obtain the on-line function for the real-time experiment from the simulation program with two-line change. This is one of the most useful features of RTMATX. See (Koga, 1995) for the detail description.

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

Func void on_task()
{
  Matrix y, xh;

  y  = sensor();
  xh = Ch*z+Dh*Y;  // state estimation
  u  = - F*xh;     // state feedback
  z = Ah*z+Bh*y+Jh*u; // observer
  actuator(u);
}
```

List 3: Function for real-time control

4.3 *On-Line Tuning*

Figure 5 and Fig. 6 show the experimental results of the inverted pendulum. We changed the poles of the observer from multiple double root -70 to multiple double root -20 at 3 second. In the first case, we didn't change the initial state of the new observer, while in the second case, we set the initial state of the new observer according to the proposed method. It is found that the proposed method works very well.

5. SUMMARY

In this paper, an integrated software environment for design and real-time implementation for control systems was introduce. The software supports comprehensively the design procedure, including analysis of plant, synthesis of controller, simulation, and experiment. The paper dealt with some problems to integration of real-time implementation facility to CACSD software and gave a solution to them. The software offers the facility to update the parameters of controllers without stopping real-time control, which helps on-line tuning of controllers. A method for on-line tuning of state observer was proposed and verified through the experiment.

REFERENCES

Inc., The Math Works (1992). *SIMULINK – User's Guide*. first ed.. 24 Prime Park Way, Natick, Mass. 01760-1500, USA.

Integrated Systems, Inc. (1991). *MATRIXx Core*. 8 ed.. 3260 Jay Street Santa Clara, California 95054, USA.

Koga, Masanobu (1995). An interactive environment for simulation and real-time implementation of control systems. *Proc. of KACC'95, Seoul, Korea* pp. 336–339.

Koga, Masanobu, Hiroaki Toriumi and Mitsuji Sampei (1996). Real-time cad of control systems achieving cooperation of modeling and design of controllers. *Proc. of CACSD'96, Dearborn, Michigan, U.S.A.* pp. 457–462.

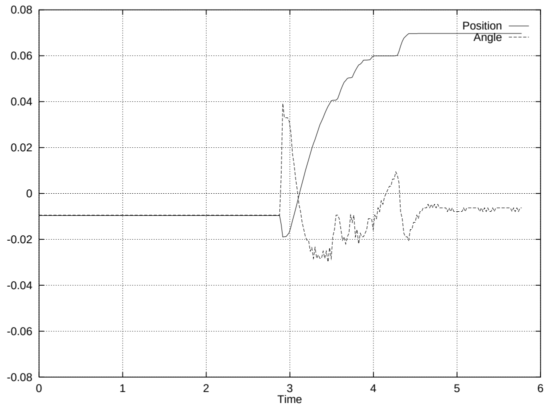

Fig. 5. The initial state of observer is unchanged

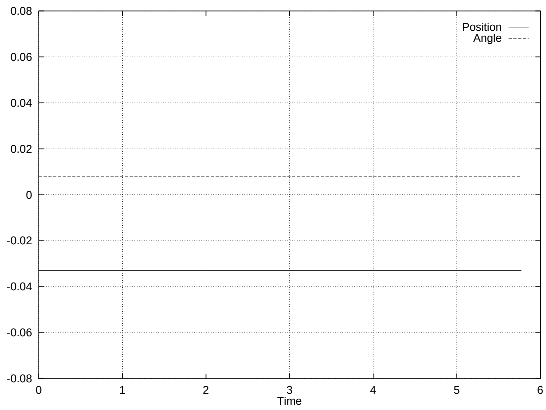

Fig. 6. The initial state of observer is changed according to the proposed method