Proceedings of the 1998 IEEE
International Conference on Robotics & Automation
Leuven, Belgium • May 1998

Parallel Dynamics Computation and H_∞ Acceleration Control of Parallel Manipulators for Acceleration Display

K.Yamane, M.Okada, N.Komine* and Y. Nakamura
(E-mail: katz@ynl.t.u-tokyo.ac.jp)

Dept. of Mechano-Informatics, Univ. of Tokyo
7-3-1 Hongo Bunkyo-ku Tokyo,
113-0033 JAPAN

*Shimadzu Corporation
1 Nishi-no-kyo Kuwahara-chyo
Nakagyo-ku Kyoto, 604-8442 JAPAN

Abstract

In this paper, we propose a control scheme of parallel manipulators focusing on the accuracy of acceleration on endplate, which is an important factor when parallel manipulators are used as acceleration displays. We use two controllers dynamic controller to achieve accuracy of position and to stabilize the system, and H_∞ controller to feedback the acceleration measured on the endplate. The main problem of dynamic control is computational complexity. In order to reduce computation time for inverse dynamics, parallel processing method called multi-thread pogramming is applied. H_∞ controller is added outside the closed loop of dynamic control to remove the vibration of the structure and the influence of modeling errors in dynamic controller.

1 Introduction

Acceleration display is one of many applications of parallel manipulators. Although the Stewart Platforms driven by hydraulic cylinders [1] have been widely used for simulators, it is difficult to control precisly such type of manipulators because of time delay and nonlinearity of the hydraulic actuators. Recently, parallel mechanisms using rotational and/or spherical joints (Figure 1), instead of sliding ones, are coming into use because servo motors are more suitable for precise control.

A problem of parallel mechanisms is that it is prohibitably costly to compute the dynamics and kinematics in real time. Therefore, in most cases, only simple position control is applied and parallel mechanisms are not used unfortunately to the best of their ability in high speed motion. There are two technical reasons for the difficulty: (1) computation of unmeasured passive joint angles, and (2) computation of dynamics due to closed kinematic chains.

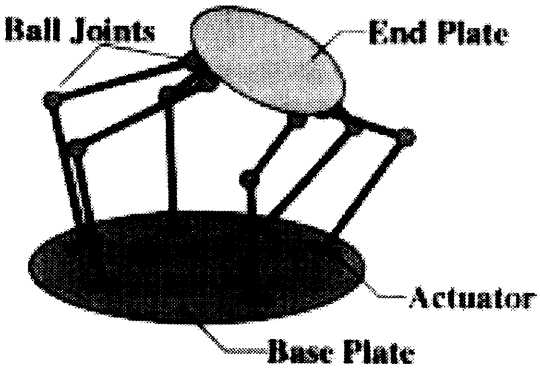

Figure 1: Parallel Mechanism

One of the time-consuming steps of dynamic control is inverse dynamics. There have been basically two approaches to the problem in closed-link structures including parallel mechanisms. They resemble in that some joints of the closed-link structure are virtually cut and the structure is transformed into equivalent open-link tree structure. The earlier method, which uses Lagrange multiplier to compute the force acting at the virtually cut joints, is not computationally effective. The later approach [2], using the Jacobian of the free joints with respect to the actuated ones, is known to require much less computation. It is also known that some of the computation in inverse dynamics of parallel mechanisms can be computed in parallel, because of the parallelism of the structure [3]. Even now real-time dynamic control without approximated dy-

namical models are not yet ready for practical use.

Dynamic control is usually designed for the rigid body model of mechanism, wheras the real mechanism includes structural flexibility that causes unmodeled vibration on the endplate. The supression of such vibration is significant when it is used for an acceleration display. Designing a controller usually requires many trial-and-error steps. An useful method, however, has been developed, in which a controller is designed systematically based on the experimental results for identification [8, 9].

The main goals of this paper are: (1) to reduce the computation time for dynamic control by taking account of the mechanical parallelism, and (2) to design an acceleration feedback controller which realizes smooth acceleration on the endplate.

We first provide with the general computational algorithm of parallel mechanisms and show its parallelism. We then implement the algorithm using parallel processing technique. Experimental results show the advantage of dynamic control over simple position control in the accuracy of position. In high-speed motions, however, a large vibration was observed on the endplate due to the structural flexibility. Also modeling errors such as the frictions of the joints, are difficult to estimate. In order to make the acceleration smooth and accurate, we designed and implemented an H_∞acceleration feedback controller for the dynamic control system. It was observed that the vibration and error of acceleration on the endplate was greatly reduced.

2 Dynamic Control Algorithm for Parallel Processing

The block diagram of dynamic control is shown in Figure 2, where θ_1 is the actuated-joint angles measured by the encoders, and $x_E, \dot{x}_E, \ddot{x}_E$ are the position, velocity, acceleration of the endplate, $\ddot{x}_{Ed}$ is the reference acceleration, $\hat{\ddot{x}}_{Ed}$ is the acceleration which the manipulator should generate, τ_1 is the torque required to generate $\hat{\ddot{x}}_{Ed}$, respectively. G is the whole acceleration-input-output system. For serial link manipulators, this is nothing but the resolved acceleration control[4].

Using the virtual cut algorithm for closed kinematic chain dynamics [3], the dynamic control for parallel manipulators requires the following subcomputations:

1. Angles of the actuated joints θ_1 are measured, and the velocity of those joints $\dot{\theta}_1$ can be computed by numerically differenciating θ_1.

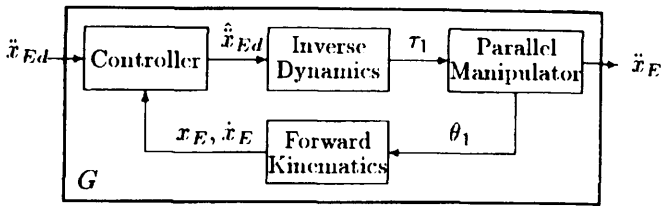

Figure 2: Block diagram of dynamic control

2. Compute the position of the endplate x_E (forward kinematics).

3. Compute the kinematics in parallel at each leg ($i = 1, 2, \ldots, 6$):

   (a) Compute the angles of the passive joints (inverse kinematics):

   $$\theta_i = g_i(x_E) \tag{1}$$

   where $g_i(x_i)$ is the function that gives all joint angles of each leg θ_i.

   (b) Compute the Jacobian matrix of each leg in parallel $J_i = \partial x_E / \partial \theta_i$ and its inverse $G_i = J_i^{-1}$.

4. Form $S = \partial \theta_1 / \partial x_E$ by gathering all the elements of $G_1, G_2, \ldots, G_6$ corresponding to the joints with actuators and compute its inverse S^{-1}.

5. Compute velocity of the endplate:

   $$\dot{x}_E = S^{-1} \dot{\theta}_1 \tag{2}$$

6. Determine the acceleration of the endplate:

   $$\hat{\ddot{x}}_E = \ddot{x}_{Ed} + K_D(\dot{x}_{Ed} - \dot{x}_E) + K_P(x_{Ed} - x_E) \tag{3}$$

   where $\ddot{x}_{Ed}, \dot{x}_{Ed}$ and x_{Ed} are the desired acceleration, velocity and position of the endplate, $\dot{x}_E$ and x_E are the current accleration and position of the endplate, K_D and K_P are the feedback gains, respectively.

7. Here we virtually cut the joints between the endplate and the last link of all legs except for an arbitary leg, thus the closed-link structure is virtually transformed into an open-link structure, assuming all the 31 uncut joints are actuated. Let θ_0 be the angles of the uncut joints and τ_0 the torques.

8. Inverse dynamics is computed in parallel at each leg ($i = 1, 2, \ldots, 6$):

(a) Compute the velocities of all joints:

$$\dot{\boldsymbol{\theta}}_i = \boldsymbol{G}_i \dot{\boldsymbol{x}}_E \tag{4}$$

and the accelerations:

$$\ddot{\boldsymbol{\theta}}_i = \boldsymbol{G}_i(\ddot{\hat{\boldsymbol{x}}}_E - \dot{\boldsymbol{J}}_i \dot{\boldsymbol{\theta}}_i) \tag{5}$$

(b) Compute the joint torques of the virtual open-link structure $\boldsymbol{\tau}_i$ by applying Newton-Euler formulation.

9. Form $\boldsymbol{W} = \boldsymbol{\theta}_0/\boldsymbol{x}_E$ by gathering all the elements of $\boldsymbol{G}_i (i = 1 \ldots 6)$ corresponding to the uncut joints.

10. Transform the torques to those of the actuated joints in the real closed-link structure:

$$\boldsymbol{\tau}_1 = \boldsymbol{S}^{-T} \boldsymbol{W}^T \boldsymbol{\tau}_0 \tag{6}$$

where $\boldsymbol{\tau}_1$ is the actuator torques.

Among these steps, 3. and 8. can be computed in parallel for each leg. The computation time would be reduced by parallel computing. Practically, since the parallel-computation part is divided by serial computation of steps 4. to 7, the overhead due to switching between serial and parallel processing becomes so large as to make parallel processing meaningless. It is desirable that there exists only one segment of parallel computation in the whole sequence of the computation.

For this purpose, we see the six legs of the parallel mechanism as six independent serial manipulators and apply Unified Computation [5] to each leg. This can be realized by taking some simple points into account:

- Set the coordinate of the endlink of each serial leg coincident to that of the endplate of the parallel mechanism. Acceleration and velocity of the endlink coincide with those of the endplate. Position can be computed easily from that of the endplate by subtracting the offset of the leg base.
- Set the mass and inertia matrix of each endlink 0 and $\boldsymbol{O}$ respectively, except for an arbitrary leg, for which their real values are used.

The new algorithm is summarized as follows:

1. Measure $\boldsymbol{\theta}_1$ and compute $\dot{\boldsymbol{\theta}}_1$.
2. Compute the position of the endplate $\boldsymbol{x}_E$.
3. Compute $\ddot{\hat{\boldsymbol{x}}}_E$ by equation (3).
4. Compute the following steps in parallel for each leg $(i = 1, 2, \ldots, 6)$:
   (a) Compute the desired position of the endlink $\boldsymbol{x}_{di}$:
   $$\boldsymbol{x}_{di} = \boldsymbol{B}_i^T \boldsymbol{x}_{Ed} - \boldsymbol{b}_i \tag{7}$$
   where $\boldsymbol{B}_i$ and $\boldsymbol{b}_i$ are the attitude and position of the base coordinate of leg i, respectively.
   (b) Compute all joint angles $\boldsymbol{\theta}_i$ and their velocities $\dot{\boldsymbol{\theta}}_i$.
   (c) Compute the Jacobian matrix $\boldsymbol{J}_i$ and its inverse $\boldsymbol{G}_i$.
   (d) Compute the virtual torques $\boldsymbol{\tau}_i$ via Unified Computation [5], assuming all leg-joints are actuated. Note that inverse dynamics algorithm for serial-link manipulators can be applied without any modification.
5. Form $\boldsymbol{W}$ and $\boldsymbol{S}$ from $\boldsymbol{G}_1, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_6$, and $\boldsymbol{\tau}_0$ from $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_6$.
6. Transform the computed virtual torques to those of actuated joints of real closed-link structure by equation (6).

In this algorithm, some values are redundantly computed at all legs. For example, six velocities of the endlinks are that of the endplate. This computation was done only once in the old algorithm. Although some duplicate computations are included, it is timewise more efficient to unify the parallel-computed part and to eliminate overheads due to synchronization and switching from parallel to serial computations and vise versa. Figure 3 shows the concept of unifying the parallel-computation part.

3 Implementation of Dynamic Control Algorithm

3.1 Multi-thread Programming

There are many architechtures of multi-processor systems for robot control. For example one can connect some PC's, or select a special computer for I/O and interrupt handling to combine with a PC for other calculation. In general, PC's are not good at handling I/O and interrupts which are essencial to the real-time control. Using special computers that have higher real-time ability has a large advantage. Disadvantages of such systems are: (1) the programs will be

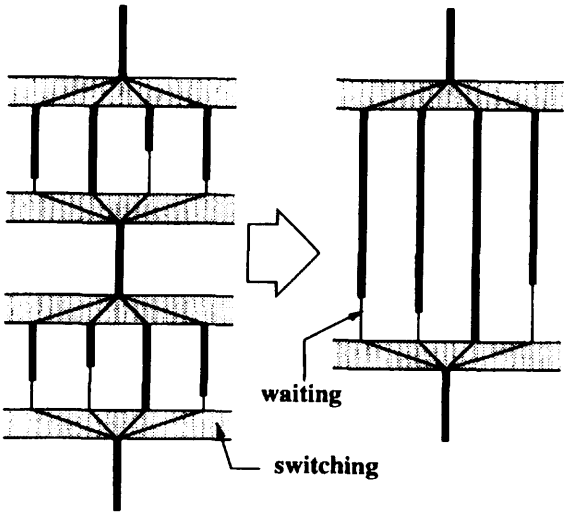

Figure 3: Unifying parallel-computation part

tailored to the system, and (2) programming parallel algorithm is usually difficult.

We chose a PC with four Pentium processors, WindowsNT for the operating system, and multi-thread programming[6] to implement parallel processing all of these technologies are standard items for PC's, though not very suitable for real-time control of manipulators. The advantages of such a system are:

- Various procedures for parallel processing synchronization, locking variables, and so on are well supported by the operating system.
- The programs to be developed for the system will run on any other systems with the same operating system (even if there is only one processor).

A multi-thread program creates multiple *threads* in one process. Threads are small parts of the whole computation and there is no restriction on the number of threads. The operating system will distribute threads among the processors in such a way to equalize computation loads to some extent. Thus, parallel processing is realized.

3.2 The Parallel Manipulator System

Figure 4 shows the parallel manipulator developed and used in the experiments. The kinematics of this parallel manipulator was originally designed by Takeda et al. [7] We scaled up to make a platform for driving simulator. It has the ability of generating 2G with a load of 70kg.

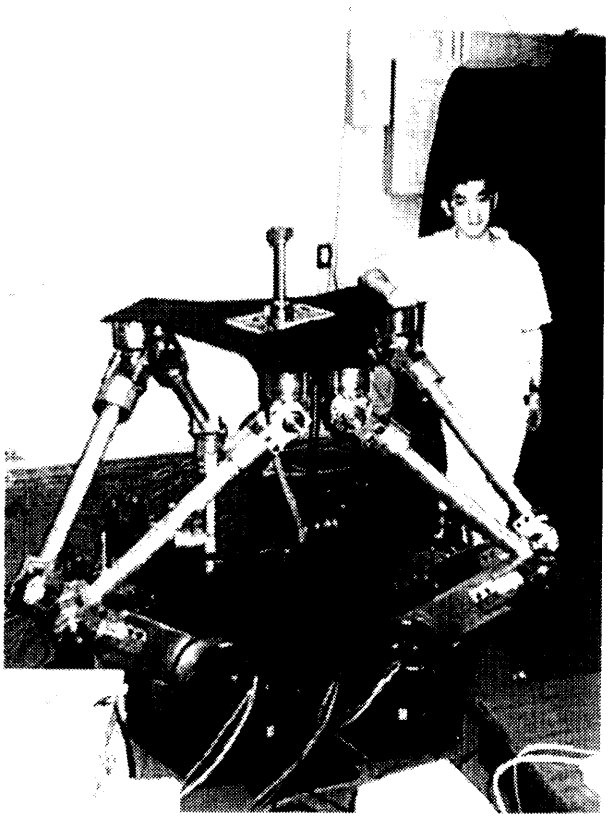

Figure 4: Parallel Manipulator

We use a PC with four Pentium Pro 200MHz processors for parallel processing. The program is written in Microsoft Visual C++. WinRT is also used to drive I/O on WindowsNT. The servo motors are equipped with encoders to measure the input angles. Computed torques are input to the servo motors, which are run in torque control mode, through a D/A board. Acceleration on the endplate is also measured by the force sensor with a mass fixed at the center of the endplate.

3.3 Computation Time (Single Thread)

The computation time of each step in the algorithm when implemented in one thread is shown in table 1. The CPU is Pentium Pro 200MHz.

Although inputs from the counter and outputs to D/A can be handled in parallel, they are handled serially in a single thread. The whole computation time, if computed in a single thread, becomes 4.6msec, the sampling frequency 220Hz.

Table 1: Computation time in one thread

computation	time	remarks
forward kinematics	0.92msec	
read encoder counter	0.60msec	6 channels
inverse dynamics (1 leg)	0.43msec	2.58msec for 6 legs
torque transformation	0.19msec	
output through D/A	0.30msec	6 channels

3.4 Implementation by Multi-thread Programming

In a multi-thread program, a thread called *primary thread* is created at the beginning of the program. The other threads are created by this thread. Unfortunately one cannot predict when and which processor executes the thread. Therefore, we must synchronize the threads by using some methods suspending / resuming threads, or setting / resetting events to let threads run in the order we want. By setting event, for example, a thread can inform others that a sequence of computations has finished.

In our program, primary thread creates seven threads: one for forward kinematics and the other six for inverse dynamics of each leg. The task and procedure of each threads become as follows (see also Figure 5):

1. *Primary Thread* Initialize parameters and I/O devices, and start the control loop. During motion, read the desired position, transform the torques, and output the torques to the D/A board.
   (a) Read desired position and wait for the forward kinematics computation to finish.
   (b) Resume the *Forward Kinematics Thread* and *Inverse Kinematics Threads* and wait for all inverse kinematics computations to finish.
   (c) Transform virtual torques into real torques, and output the result to the D/A board.
2. *Forward Kinematics Thread* Read encoder counter board and compute current position of the endplate.
   (a) When the computation of forward kinematics finishes, set *Event1* and suspend itself.
   (b) Resumed by *Primary Thread*, return to (a).
3. *Inverse Dynamics Threads* Compute the virtual torque of the open-link structure.
   (a) Resumed by *Primary Thread* when all the necessary information is ready.
   (b) Finishing the computation of inverse dynamics, suspend itself. The last one sets *Event2* before suspending.

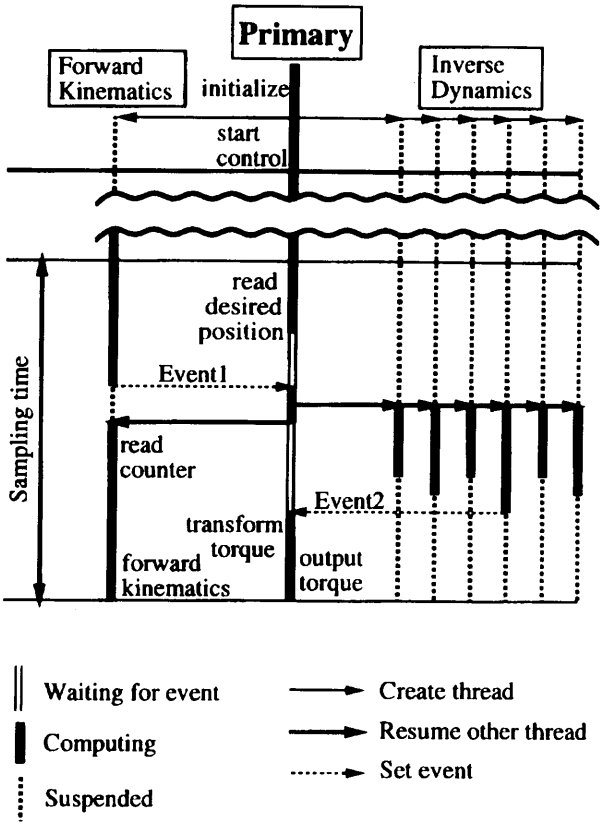

Figure 5: Scheduling of the threads

3.5 Computation Time (Multi Thread)

The sampling time was reduced by the parallel processing down to 2.0msec, which is less than half of the time when computed serially.

It is impossible to reduce the computation time to a quarter of the original even there are four processors because of two reasons: (1) not all four processors are always computing at one time, and (2) some idling time is unavoidable waiting for other thread to finish computation.

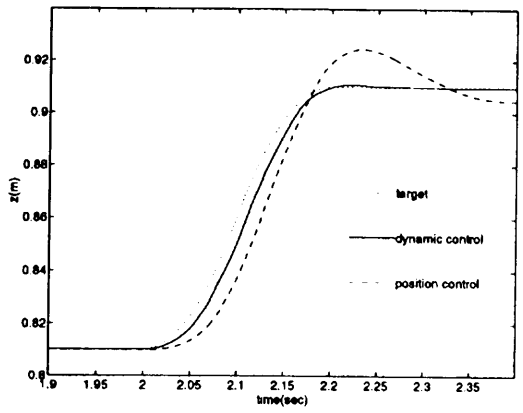

Figure 6: Endplate position in virtical motion

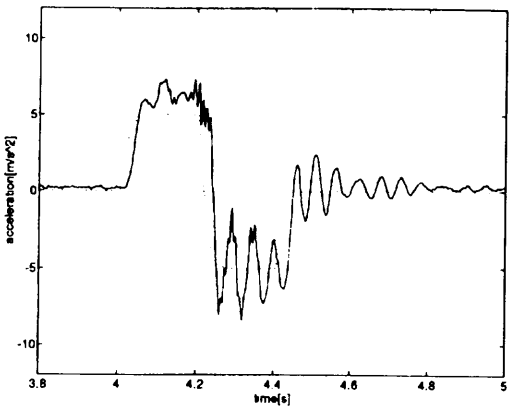

Figure 7: Acceleration measured on the endplate

3.6 Experimental Results

We executed experiments to see the advantage of dynamic control. The reference path is a simple constant accleration motion in vertical direction.

3.6.1 Position

The motion shown in Figure 6 appeared when upward reference acceleration of $10m/s^2$ for the first 0.1 seconds, and downward reference acceleration of $10m/s^2$ for the next 0.1 seconds, were given. The position is computed from the input angles via forward kinematics computation. It is observed that timelag and overshoot are greatly reduced by dynamic control.

3.6.2 Acceleration on the Endplate

The real acceleration was measured on the endplate for constant acceleration motion as shown in Figure 7. Step reference acceleration of $5m/s^2$ was given for the first 0.2 seconds, $-5m/s^2$ for the next 0.2 seconds. Although dynamic control achives precise motion in terms of position, the acceleration measured on the endplate has large error and vibration due to mechanical flexibility of links of the parallel structure.

It is impossible to remove the mechanical vibration of the manipulator by this control method because it is not detected by the value of the encoder. The timelag, supposed to be caused by the friction of the joints, are also difficult to be estimated.

4 Acceleration Feedback Control

4.1 Joint Design of Identification and Controller

"Joint Design of Identification and Controller" [8, 9] is an approach to design a desirable controller which satisfies the control specification iteratively, based on H_∞or H_2 control theory. Since the controller is designed systematically based on the experimental results using identification theory, this method can avoid trial-and-error steps.

In this paper, we identified the closed-loop system including dynamic controller and then designed an H_∞controller because of the following reasons:

- By system identification, we can get a model of unmodeled dynamics such as the vibration due to the mechanical flexibility.
- As a system is stabilized with the dynamic controller, we can apply open-loop identification [1].
- Since H_∞controller is designed in frequency domain, it is suitable for removing vibrations of known frequency.

[1] To identify a plant of an open-loop system, while closed-loop identification implies to identify a plant *inside* a closed loop. It is known that closed-loop one is much more difficult than open-loop one. Our problem is to identify the mechanism with dynamic controller, for which open-loop identification can be applied.

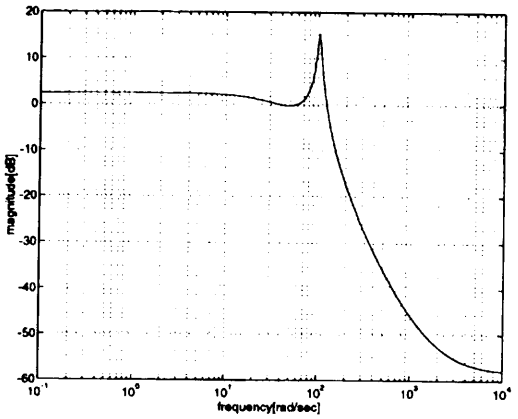

Figure 8: Bode plot of virtical direction system

4.2 Designing Controller

It is desirable to identify the whole 6DOF system with 6 inputs and 6 outputs. In this case the designed controller will be high dimensional and require a large computational load. Regarding the cross terms as disturbance, we assume that six axes are independent of each others and identify the system of each axis separately. A rationale behind this assumption is the fact that the dynamic controller approximately decouples the system's behavior.

The controller is designed in the following steps:

1. System Identification
   First, we identify the system G as a linear time-invariant system G_m including the dynamic controller, whose input is the reference accleration and output is the measured acceleration of the endplate. Note that both input and output are acceleration. Here we identify the SISO system of six axes separately assuming all axes are independent of each others. For the identification an M-sequence signal [10] was used for reference. The model was assumed to be of the third order, which includes vibration of the second order and the first order delay. The Bode plot of the identified system of virtical direction is shown in Figure 8. There is a pole near the frequency of 100 rad/s as expected from Figure 7.

2. Controller Design

   Next, we design the controller K using the identified model G_m, which satisfies the following cost function J:

$$J = \left\| \begin{array}{c} W_T K(I + G_m K)^{-1} G_m \\ W_S G_m (I + K G_m)^{-1} \end{array} \right\|_\infty < 1 \qquad (8)$$

   where I is the unit matrix, W_T and W_S are the weighting functions. Model error Δ is assumed to be in the form of $G = G_m(1 + \Delta)$. Robustness against model error and low-sensitivity against noise are realized by W_T and W_S, respectively. W_T and W_S are obtained from the spectral analysis of the experimental results.

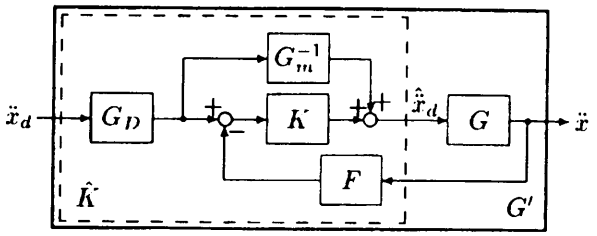

Figure 9: Acceleration feedback system

3. Adding the controller to the system
   The controller K is put into the closed-loop system shown in Figure 9, where G_D and F are the desired response of the system and the filter to remove noise of the sensor, and set as follows, respectively.

$$G_D = \frac{100^2}{(s + 100)^2} \qquad F = \frac{1}{s + 300} \qquad (9)$$

   Finally we obtain the controller $\hat{K}$, whose inputs are the original reference acceleration given by the user $\ddot{x}_d$ and the sensed acceleration $\ddot{x}$, output is the reference acceleration to the dynamic controller $\hat{\ddot{x}}_d$. If $\Delta = 0$, the transfer function from $\ddot{x}_d$ to $\ddot{x}$ becomes G_D.

4. Repeat 1.-3.
   When the system show unsatisfactory responses, we identify the closed-loop system as a new system G' and repeat the procedures of identification, designing and adding the new controller.

4.3 Experimental Results

We designed and implemented a controller in the way described in the previous section. The same reference acceleration as in Figure 7 was given to the new system with acceleration feedback. The acceleration measured on the endplate is shown in Figure 10.

Comparing to the result in dynamic control only (Figure 7), it is observed that the vibration and time-lag are greatly reduced. The influence of the cross terms with the other axes has turned out to be sufficiently small, which experimentally supports our assumption of independency of the six axes.

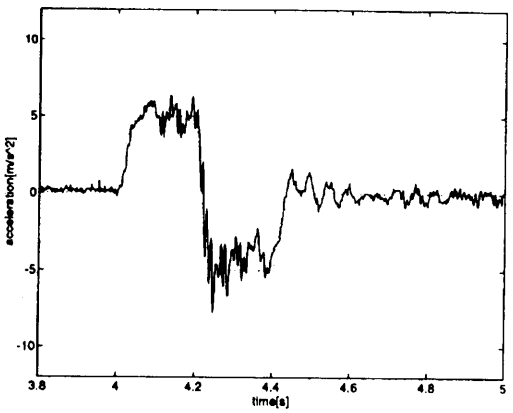

Figure 10: Measured acceleration with feedback

5 Conclusions

The following four conclusions concerning the control of parallel mechanisms for accleration displays were earned in this study:

1. A new dynamic control algorithm was established for parallel manipulators taking the mechanical parallelism into account. The algorithm is more efficient when computed in parallel.

2. The algorithm was implemented using multi-thread programming, and real-time dynamic control was realized without simplification.

3. In order to remove the vibration observed in high-speed motion, acceleration feedback controller was designed through identification and H_∞control theory.

4. Experimental results illustrated that the whole system achives very smooth and accurate acceleration on the endplate.

Acknowledgments

This work was partially supported by the Ministry of Education, Culture, and Sports (the Grant in Aid of Scientific Research: Prototyping Research (B)(2) 06555065), the International Robotics and Factory Automation Center (Committe for High Speed Parallel Mechanisms, 1995-97), and the Hayao Nakayama Foundation for Science & Technology and Culture. The second author was supported by the "Reserach for the Future" Program from the Japan Society for the Promotion of Science (JSPS-RFTF 96P00801). The authors appreciate and acknowledge that all of these supports were essential for completing this work.

References

[1] Stewart, D.: "A Platform with Six Degrees of Freedom", Proceedings of the Institution of Mechanical Engineers, Vol.180 Pt.1 No.15, 1965-1966.

[2] Nakamura, Y. and Ghodoussi, M.: "Dynamics Computation of Closed-Link Robot Mechanisms with Nonredundant and Redundant Actuators", IEEE Transactions on Robotics and Automation Vol.5 No.3, 1989.

[3] Nakamura, Y.:"Dynamics Computation of Parallel Mechanisms", Journal of Robotics Society of Japan, Vol.10 No.6 pp.709-714, 1992 (in Japanese).

[4] Luh, J.Y.S., Walker, M.W. and Paul, R.P.C.: "Resolved Acceleration Control of Mechanical Manipulators", IEEE Trans. Automatic Control, pp.468-474, 1980.

[5] Nakamura, Y., Yokokohji, Y., Hanafusa, H. and Yoshikawa, T.: "Unified Computation of Kinematics and Dynamics for Robot Manipulators", Transactions of Society of Instrument and Control Engineers, Vol.23 No.5 pp.71-78, 1987 (in Japanese).

[6] Lewis, B. and Berg, D.J.:"Threads Primer", Sun Soft, Inc., 1996.

[7] Takeda, Y., Funahashi, H.:"A Development of a Spatial In-parallel Actuated Manipulator with Six Degree of Freedom with Consideration of Motion Transmissibility", Proceedings of 1993 Conference of Robotics Society of Japan, pp.853-856, 1993 (in Japanese).

[8] Sugie, T. and Okada, M.:"Iterative Controller Design Method on Closed-Loop Identification", Proceedings of the 3rd ECC, Vol.2, pp.1243-1248, 1995.

[9] Gevers, M.:"Towards a Design of Identification and Control?", Essays on Control, Trentelman, Willems Eds.(Birkhäuser), pp.111-151, 1993.

[10] Ljung, L.:"System Identification Theory for the User", Prentice Hall, 1987.