Ver. 1.0 (November 14th , 2024)

Benchmark Problem for Magnetic-Head Positioning Control in HDD ~ Track-Seek Control ~

User Manual

Investigating R&D Committee on “Basis of Collaborative Technologies for Precision Servo Systems”
The Institute of Electrical Engineers of Japan

1. Introduction

2. Track-seek control in HDDs

3. Benchmark problem

1. Introduction

2. Track-seek control in HDDs

3. Benchmark problem

Objective

To encourage research on magnetic-head positioning control in hard disk drives (HDDs) , we release benchmark problems that work on MATLAB. The magnetic-head positioning system has two control modes: one is track-seek control, which moves the magnetic head onto the target track, and the other is track-following control, which maintains the magnetic-head position on the target track. This benchmark problem can simulate track-seek control using user-designed controllers. For track-following control, we release another HDD benchmark problem at the following site: https://jp.mathworks.com/matlabcentral/fileexchange/111515-magnetic-head-positioning-control-system-in-hdds

Benchmark problem

- We can download the code from the MathWorks File Exchange https://jp.mathworks.com/matlabcentral/fileexchange/175619-hdd-benchmark-problem-for-track-seek-control
- Requires: MATLAB, Control System Toolbox
- Compatible with R2019b and later releases

Feature of the benchmark problem

- Platform: MATLAB (requires Control System Toolbox)

- Simulated HDD: For cloud server (so-called “Nearline storage”)
  - Track pitch: 482 kTPI (52.7 nm)
  - 7200 rpm
  - Helium sealed
  - Dual-actuator system: Voice coil motor (VCM) + PZT actuator

- Control design problem
  - Track-seek control for short spans in storage boxes for the cloud server
  - Sampled-data control system with multi-rate holds
  - The controlled object: Dual-Input-Single-Output system

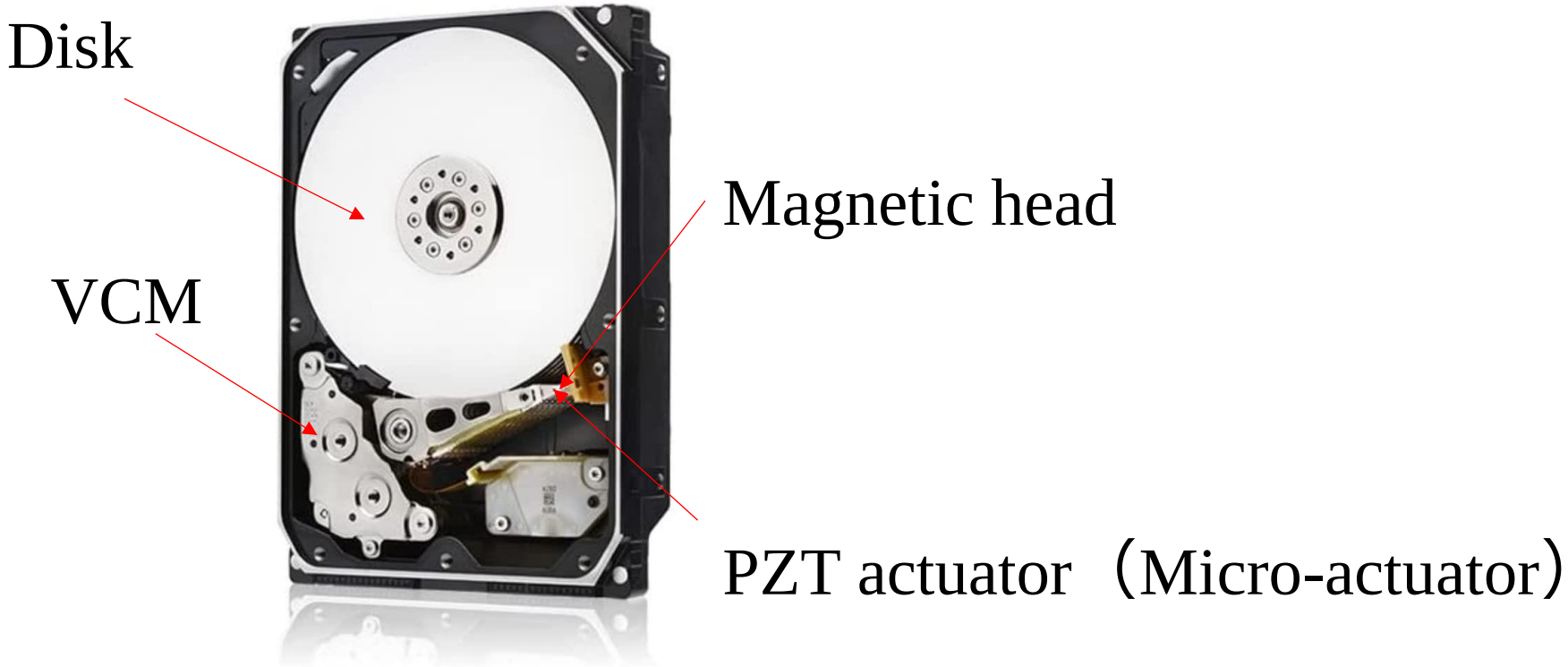
1. Introduction

2. Track-seek control in HDDs

3. Benchmark problem

Magnetic-Head Positioning Control System in the Benchmark Problem

- The control system has two actuators: a VCM and a PZT actuator.
- The control variable is the magnetic-head position.
- The PZT actuator has a stroke limitation of 50 nm in this benchmark.
- We focus on track-seek control with a seek-span of less than or equal to 1024 tracks.



The mechanical characteristics of the actuators and the example controllers are derived from the following papers

T. Atsumi and S. Yabui, “Quadruple-Stage Actuator System for Magnetic-Head Positioning System in HDDs,” The IEEE Transactions on Industrial Electronics, Vol. 67, No. 11, pp. 9184-9194, (2020-11)

- Mechanical characteristics of VCM and PZT
- Example feedback controllers

T. Atsumi, K. Suzuki, S. Nakamura, and M. Ohta, “Vibration Control with Thin-Film-Coil Actuator for Head-Positioning System in Hard Disk Drives,” Journal of Advanced Mechanical Design, Systems, and Manufacturing, vol. 9, no. 1, Paper No. 14-00466, pp. 1-12, (2015-3)

- Mechanical characteristics of VCM (torsional mode around 3.4 kHz)

T. Atsumi, “Feedforward Control Using Sampled-Data Polynomial for Track Seeking in Hard Disk Drives,” The IEEE Transactions on Industrial Electronics, vol. 56, no. 5, pp. 1338-1346, (2009-5)

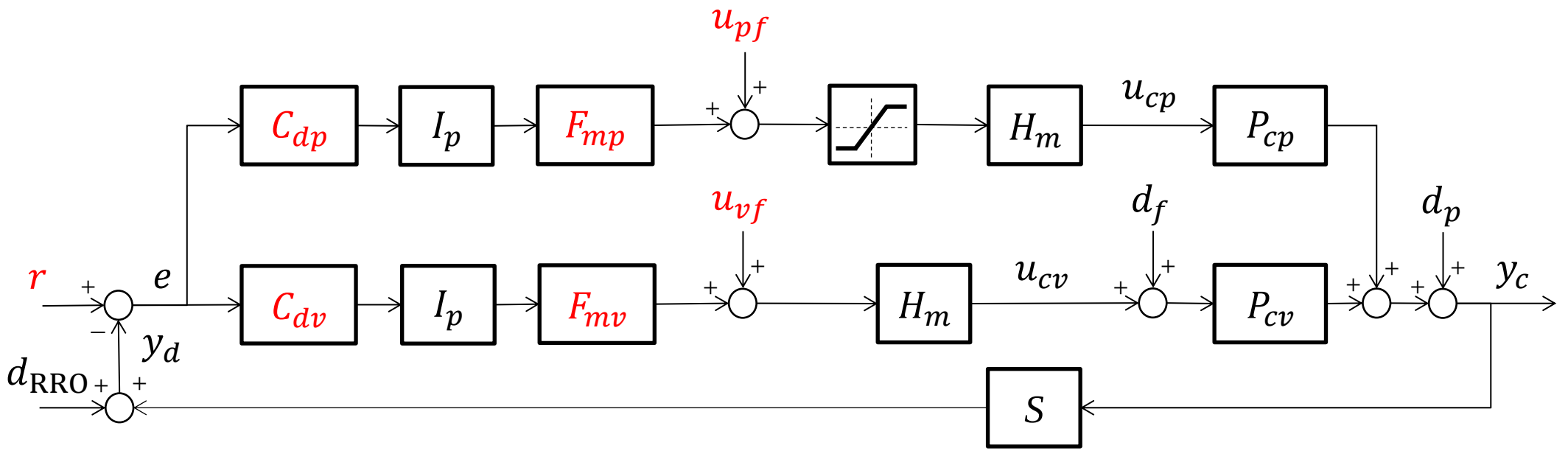
- Example feedforward controllers

Block Diagram of Magnetic-Head Positioning Control System

P_{cv} : VCM in continuous-time system, P_{cp} : PZT actuator in continuous-time system, C_{dv} : Feedback controller for VCM, C_{dp} : Feedback controller for PZT actuator, F_{mv} : Multi-rate filter for VCM, F_{mp} : Multi-rate filter for PZT actuator

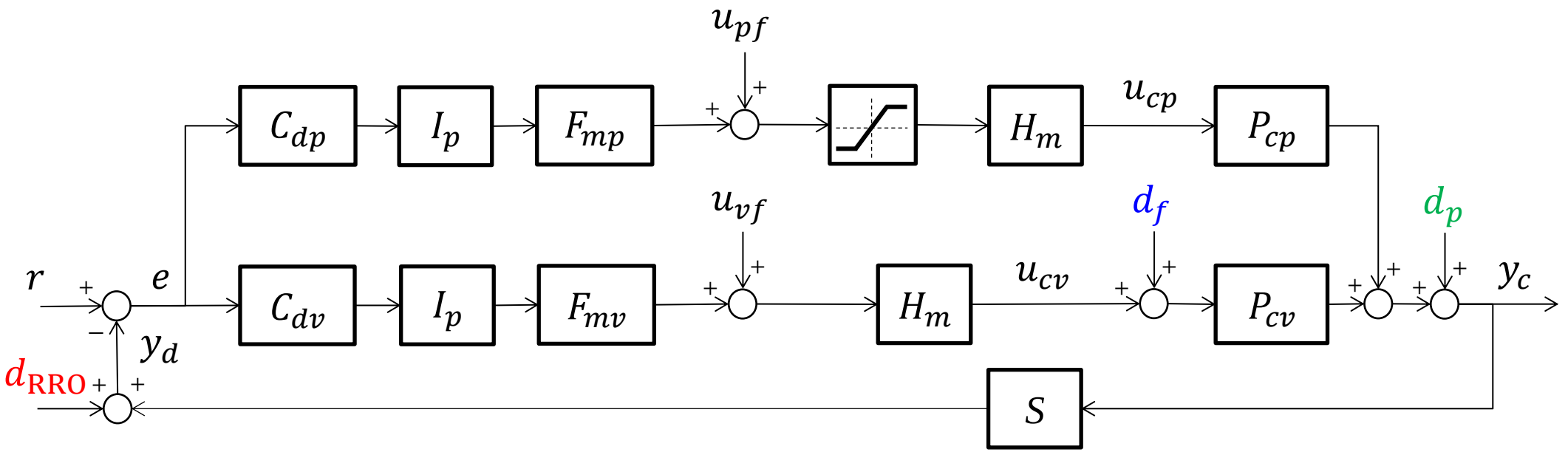
I_p : Interpolator, H_m : Multi-rate zero-order hold, S : Sampler, d_p : Fan-induced vibration, d_f : Rotational vibration (RV), d_{RRO}: Repeatable Run-Out (RRO), y_c : Magnetic-head position in continuous time, y_d : Measured magnetic-head position, e : Tracking error, u_{cv}: Control input for VCM in continuous time, u_{cp}: Control input for PZT actuator in continuous time, r : Reference, u_{vf} : Feedforward control input for VCM, u_{pf} : Feedforward control input for PZT actuator

*User-designed controllers are shown in red font.



Disturbances in the Benchmark Problem

- d_{RRO}：RRO（Repeatable Run-Out）
  Oscillation of target tracks written on the disk. Note that it is regarded as noise.
- d_f：RV（Rotational Vibration）
  External vibration caused by other HDDs in a storage box.
- d_p：Fan-Induced Vibration
  External vibration caused by cooling fans in a storage box.



1. Introduction

2. Track-seek control in HDDs

3. Benchmark problem

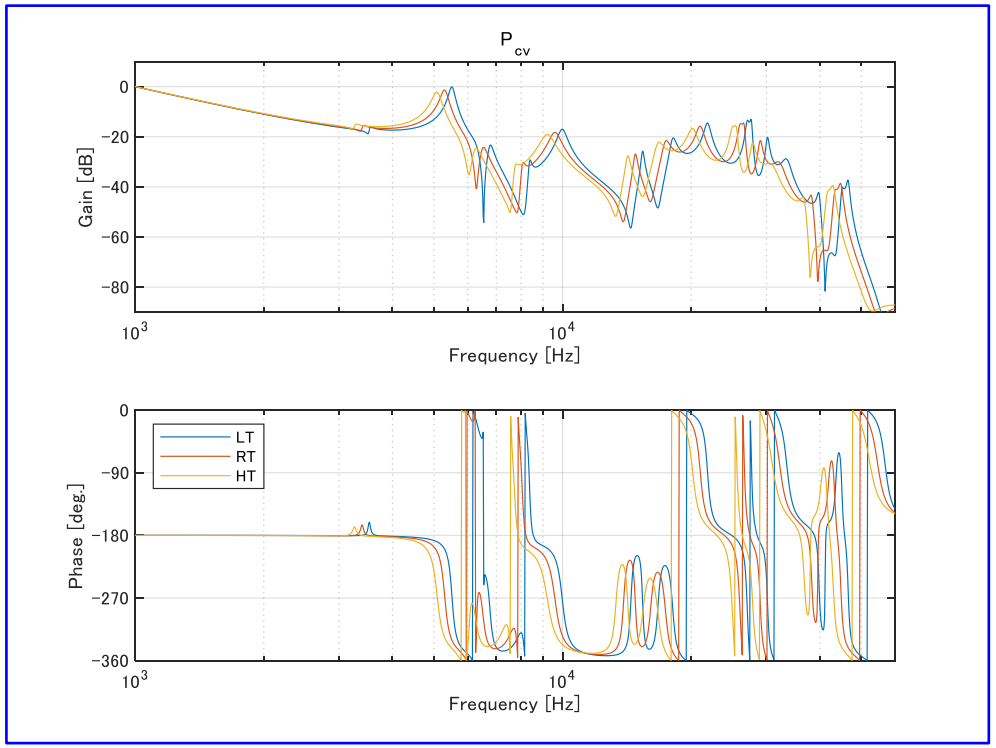
- Control system in the benchmark problem
- How to use the benchmark problem
- Example
- MATLAB code

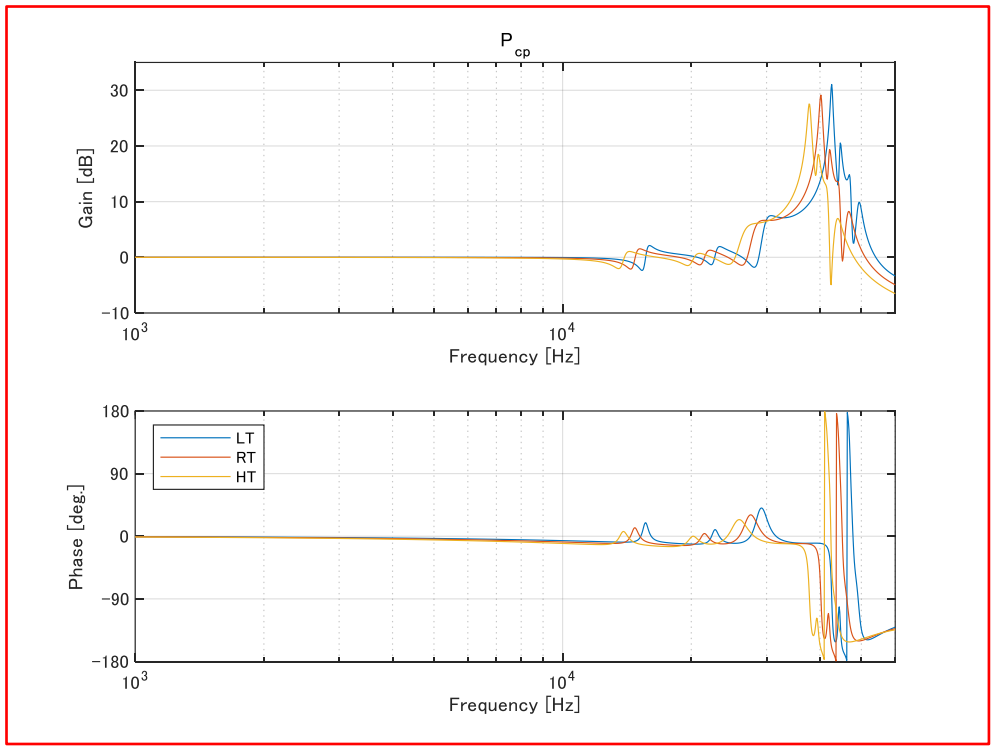
- Control system in the benchmark problem
- How to use the benchmark problem
- Example
- MATLAB code

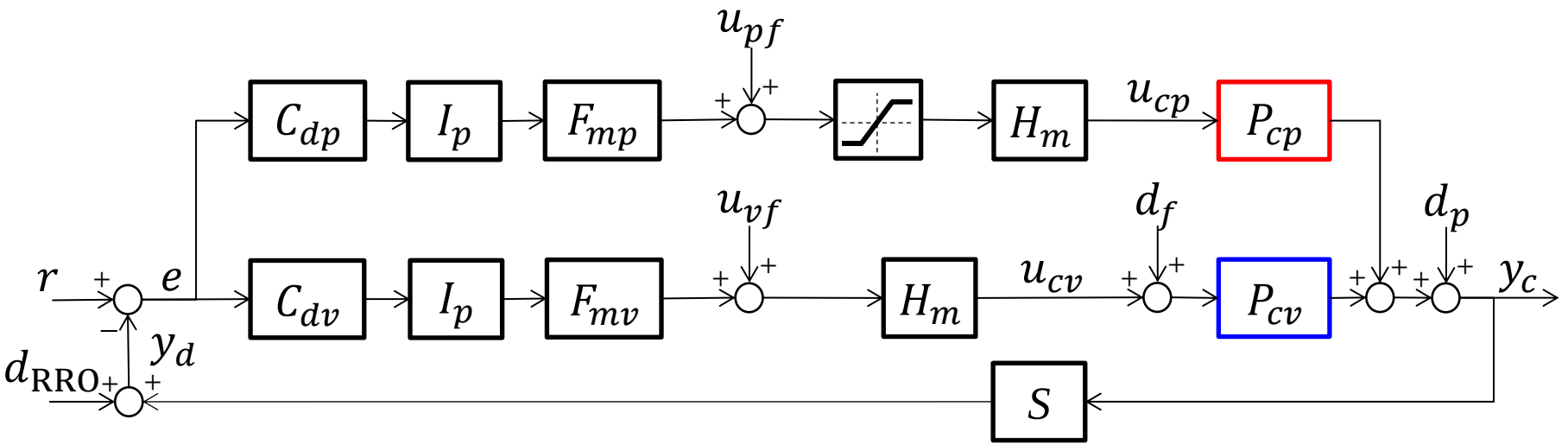
Model of Controlled Object

- Parameters of the nominal controlled object are derived from the following papers.
  - T. Atsumi and S. Yabui, “Quadruple-Stage Actuator System for Magnetic-Head Positioning System in HDDs,” The IEEE Transactions on Industrial Electronics, Vol. 67, No. 11, pp. 9184-9194, (2020-11)
  - T. Atsumi, K. Suzuki, S. Nakamura, and M. Ohta, “Vibration Control with Thin-Film-Coil Actuator for Head-Positioning System in Hard Disk Drives,” Journal of Advanced Mechanical Design, Systems, and Manufacturing, vol. 9, no. 1, Paper No. 14-00466, pp. 1-12, (2015-3)
- Sampling time: $1/(7200/60)/420 \approx 1.9841 \times 10^{-6}$ (7200rpm, 420 sector)
- Number of multi-rate: 2
- Track pitch：482 kTPI (52.7 nm)
- Controlled object variations
  - The number of cases for controlled objects: 3
    - Case 1: LT (Low temp.)
    - Case 2: RT (Room temp.)
    - Case 3: HT (High temp.)
  - Temperature dependencies of mechanical resonant frequencies
    - LT(Low temp.)：VCM: +4 % from nominal model, PZT actuator: +6 % from nominal model
    - RT(Room temp.)：Same as nominal models
    - HT(High temp.)： VCM: -4% from nominal model, PZT actuator: -6% from nominal model
  - Temperature dependencies of damping ratios
    - LT(Low temp.)：VCM: -20 % from nominal model, PZT actuator: -20 % from nominal model
    - RT(Room temp.)：Same as nominal models
    - HT(High temp.)： VCM: $+20\%$ from nominal model, PZT actuator: $+20\%$ from nominal mode

Frequency Responses of Controlled Object for All Cases







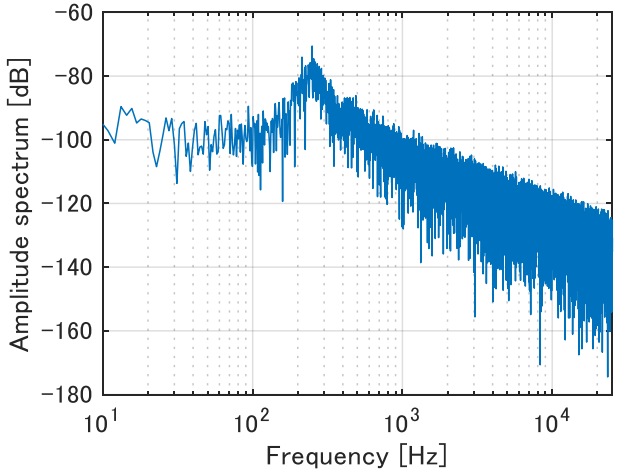
Disturbances in Benchmark Problem

- RV: Reproduced by the disturbance shown in the following paper
  T. Atsumi and S. Yabui, “Quadruple-Stage Actuator System for Magnetic-Head Positioning System in HDDs,” The IEEE Transactions on Industrial Electronics, Vol. 67, No. 11, pp. 9184-9194, (2020-11)

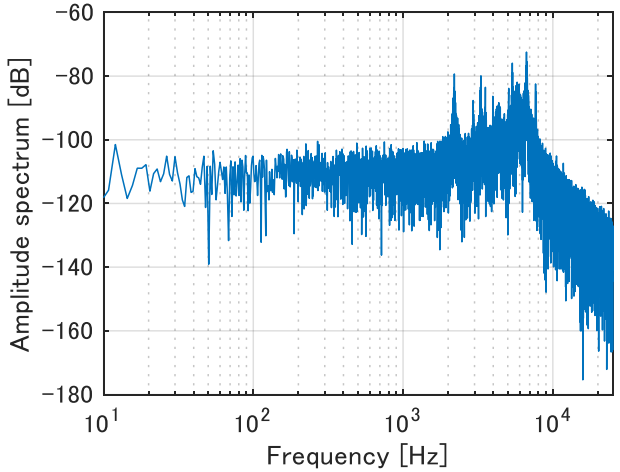
- Fan-Induced Vibration: : Reproduced by the disturbance shown in the following paper
  T. Eguchi, Y. Asai, K. Ichikawa, and M. Takada: ``Airborne and Structure-Borne Transmission of High Frequency Fan Vibration in a Storage Box", Proceedings of the ASME 2017 Conference on Information Storage and Processing Systems, Paper No. 5406 (2017)

- RRO: Time-domain signal which has the same amplitude for all frequencies except 0 Hz.

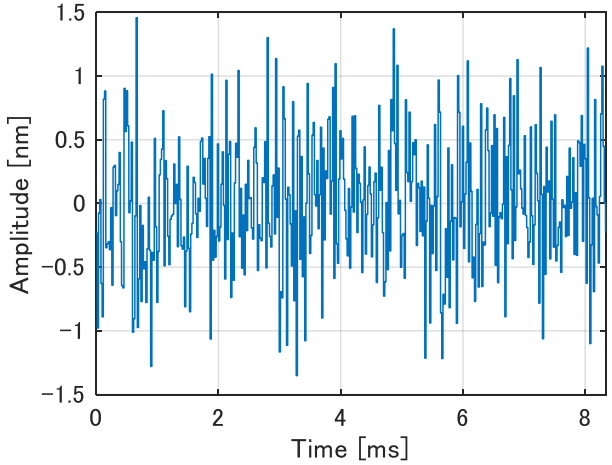
RV



Fan-Induced Vibe.



RRO



- Control system in the benchmark problem
- How to use the benchmark problem
- Example
- MATLAB code

Setting

Step 1. Download the MATLAB code from MathWorks File Exchange (https://jp.mathworks.com/matlabcentral/fileexchange/175619-hdd-benchmark-problem-for-track-seek-control) .

Step 2. Unzip the “HDDTrackSeek.zip”.

Step 3. Start MATLAB.

Step 4. Set MATLAB Current Folder as the unzipped folder.

Usage

Step 1. Define the look-up tables for your designed feedforward control input as follows.

- r: r_table (sampling time： 1/120/420 [s])
- u_{vf}: uvf_table (sampling time： 1/120/420/2 [s]) The table length must be 2 times that of r_table.
- u_{vp}: upf_table (sampling time： 1/120/420/2 [s]) The table length must be 2 times that of r_table.

Step 2. Define the model objects (MATLAB command: “ss” or “tf”) of your designed controllers as follows.

- C_{dv} : Sys_Cdv (sampling time： 1/120/420 [s])
- C_{dp} : Sys_Cdp (sampling time： 1/120/420 [s])
- F_{mv} : Sys_Fmv (sampling time： 1/120/420/2 [s])
- F_{mp} : Sys_Fmp (sampling time： 1/120/420/2 [s])

Step 3. Save “r_table”, “uvf_table”, and “upf_table” as a MAT file named “Data_FF_xx.mat”. xx means seek span.

Step 4. Save “Sys_Cdv” and “Sys_Cdp” as a MAT file named “Data_Cd.mat”.

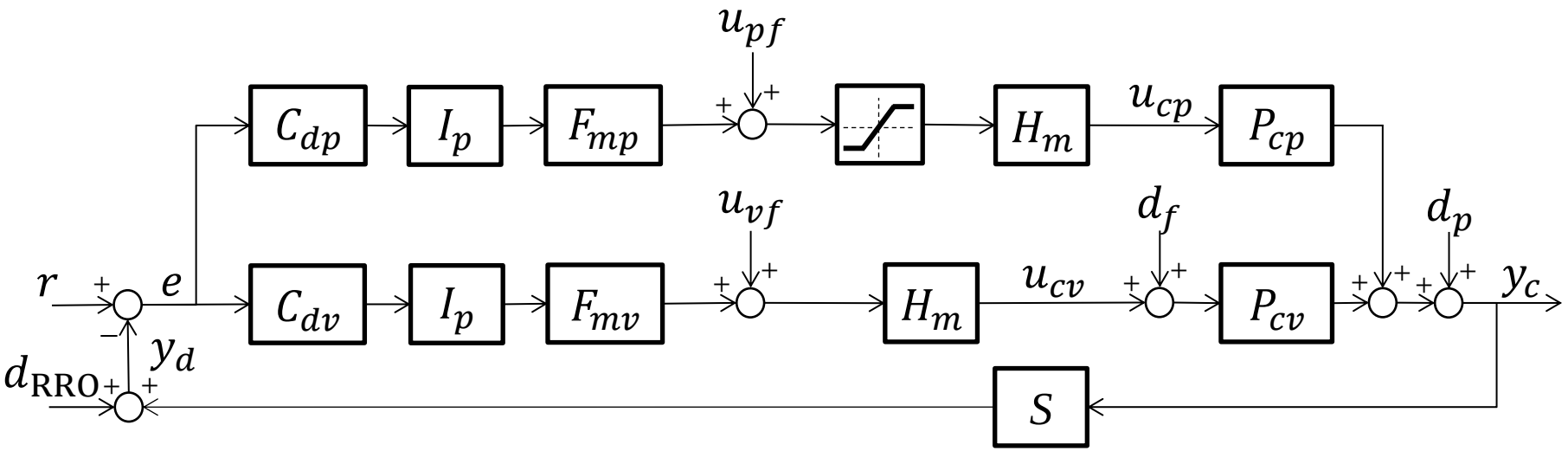
Step 5. Save “Sys_Fmv” and “Sys_Fmp” as a MAT file named “Data_Fm.mat”.

Step 6. Execute “Simulation_trackseek” to show simulation results of the track seek control.

Step 7. Execute “Plot_control_system” to show frequency responses of the control system.

Validation of Track-Seek Control Results

- We evaluate the moving time of magnetic-head position y_c from 0 to target track (user defined). Generally, track-seek control can be considered complete when the magnetic head position is within $\pm 10\%$ of the track width relative to the target track.
- This benchmark problem focuses solely on short-span seek. Therefore, the target seek span must be less than or equal to 1024 tracks.
- The maximum value of $|u_{cp}|$ must be smaller than 50×10^{-9} to ensure that the stroke of the PZT actuator is not larger than 50 nm.



Disturbance Condition

If you want to evaluate without the influence of disturbances, set the value of “SW_dist” to 0 on the fourth line of “Simulation_trackseek.m”.

Simulation_trackseek.m

```
1    clc;clear
2
3    %% Switch for disturbances
4    SW_dist=0;     % Disable: 0, Enable: 1
5    SW_source=0;   % Fixed: 0 (for evaluation), Unfixed: 1 (for learning)
```

If you want to employ a machine-learning method using tracking error with disturbances where the sources are not fixed, set the values of “SW_dist” and “SW_source” to 1 on the fourth and fifth lines of “Simulation_trackseek.m”.

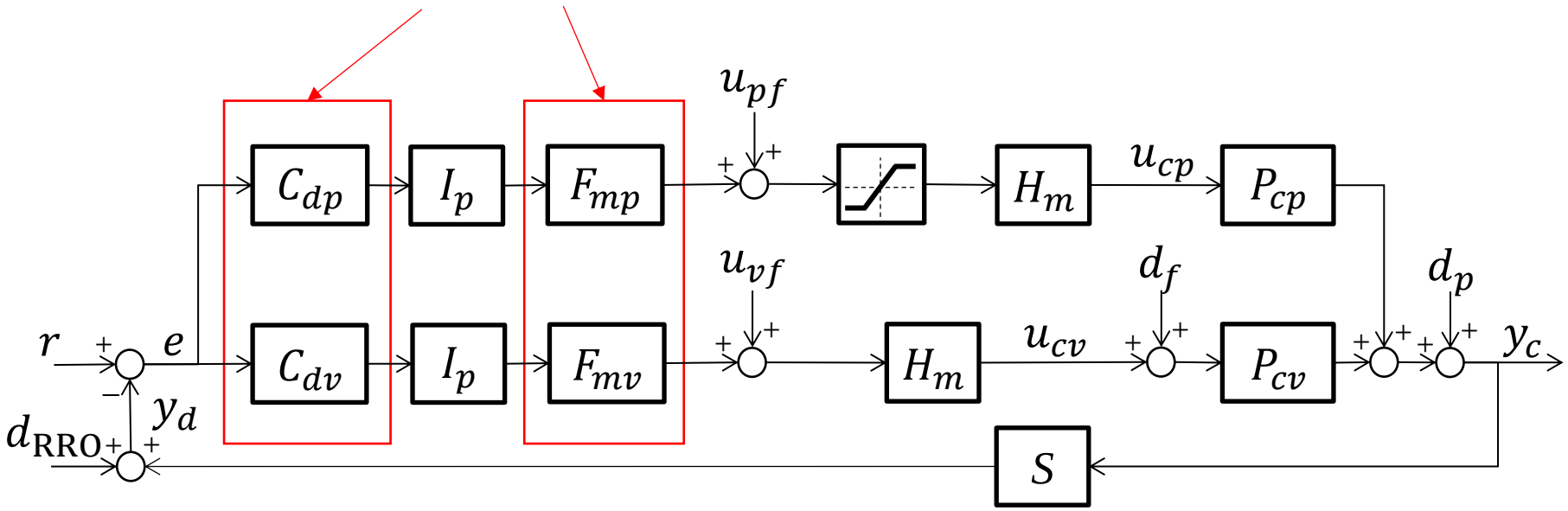
Simulation_trackseek.m

```
1    clc;clear
2
3    %% Switch for disturbances
4    SW_dist=1;     % Disable: 0, Enable: 1
5    SW_source=1;   % Fixed: 0 (for evaluation), Unfixed: 1 (for learning)
```

Additional information

If you employ feedback controllers that are not LTI systems, you can rewrite lines 162 to 168 and lines 198 to 204 in “Function_simulation_seek.m”. The code explanation is on pages 37 and 38 of this manual.

Rewrite the MATLAB code “Function_simulation_seek.m” by yourself.



- Control system in the benchmark problem
- How to use the benchmark problem
- Example
- MATLAB code

The example controllers are derived from the following papers

T. Atsumi and S. Yabui, “Quadruple-Stage Actuator System for Magnetic-Head Positioning System in HDDs,” The IEEE Transactions on Industrial Electronics, Vol. 67, No. 11, pp. 9184-9194, (2020-11)

- Example feedback controllers

T. Atsumi, “Feedforward Control Using Sampled-Data Polynomial for Track Seeking in Hard Disk Drives,” The IEEE Transactions on Industrial Electronics, vol. 56, no. 5, pp. 1338-1346, (2009-5)

- Example feedforward controller (u_{vf})

9184 IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 67, NO. 11, NOVEMBER 2020

Quadruple-Stage Actuator System for Magnetic-Head Positioning System in Hard Disk Drives

Takenori Atsumi, *Member, IEEE*, and Shota Yabui, *Member, IEEE*

***Abstract*—In this article, we present a magnetic-head positioning system in hard disk drives (HDDs) with a quadruple-stage actuator system. The quadruple-stage actuator system consists of a voice coil motor, a milliactuator with two piezoelectric (PZT) elements, a microactuator with two PZT elements, and a thermal actuator. The milliactuator is the first generation of a secondary actuator for a dual-stage actuator in the HDDs and moves a sway mode of a suspension in a head-stack assembly. The microactuator is the second generation of the secondary actuator for the dual-stage actuator in the HDDs and moves a yaw mode of the suspension. The thermal actuator is an actuator in a development phase for future HDDs. It consists of heaters embedded in the magnetic head and moves read/write elements a few nanometers in a horizontal direction with thermal expansion. The achievable disturbance-rejection performance of the proposed quadruple-stage-actuator system is higher than that of the conventional triple-stage actuator system because the proposed system can decrease a negative impact caused by the stroke limitation of the thermal actuator. As a result, the magnetic-head positioning system with the quadruple-stage-actuator system enables us to improve the positioning accuracy under the external vibrations by about 48% from that with the triple-stage-actuator system.**

***Index Terms*—Hard disks, microactuators, positioning control, servo systems.**

I. Introduction

TO INCREASE the data capacity of hard disk drives (HDDs), we have to improve the positioning accuracy of magnetic heads in the HDDs so that the size of bits for information stored on a disk decreases [1]–[3].

In the magnetic-head positioning system, one of the most critical issues is positioning errors caused by external vibrations of file servers used in data centers. Most of the positioning errors are below 2 kHz [4]. Thus, to compensate for the negative impacts of the external vibrations, we have to increase a servo bandwidth of the control system in the magnetic-head positioning system. However, the mechanical characteristics limit the servo bandwidth [5], [6]. Therefore, the magnetic-head positioning system should be fabricated with multiple actuators so that the servo bandwidth can be increased [7].

In current HDDs, a dual-stage-actuator system has been employed for the magnetic-head positioning systems. In the dual-stage-actuator system, the first actuator is a voice coil motor (VCM), and the second actuator is a piezoelectric (PZT) actuator [8]–[18]. Since 2016, we have had two kinds of PZT actuators in shipped HDDs. The first one is “milliactuator” which moves a sway mode of a suspension to control the magnetic-head position. The second one is “microactuator” which moves a yaw mode of the suspension. The frequency of the primary mechanical resonance of the microactuator is much higher than that of the milliactuator [19], [20].

For the magnetic-head positioning system, we have reported that a triple-stage actuator system was dramatically able to improve the positioning accuracy during a track-following control from the dual-stage actuator systems [21]–[23]. Since 2015, other research groups have also reported advantages of the triple-stage actuator system [24]–[26]. In the triple-stage actuator system, magnetic heads have a heater located in a horizontal direction of read/write elements in order to control the position of the read/write elements with thermal expansion induced by the heater with an electric current. However, in the triple-stage actuator system, a bottleneck of the servo bandwidth is not its mechanical resonances, but its stroke limitation.

In this article, to overcome the negative impact caused by the stroke limitation of the thermal actuator, we present a magnetic-head positioning system in an HDD with a quadruple-stage actuator system. The first stage is the VCM for moving a head-stack assembly (HSA), the second stage is the milliactuator for moving a sway mode of a suspension, the third stage is the microactuator for moving a yaw mode of a suspension, and the fourth stage is the thermal actuator. The proposed approach is verified by simulations based on measured controlled objects and a disturbance source.

II. Quadruple-Stage Actuator System for HDD

A current shipped HDD is comprised of a cover, a base, a VCM, several PZT actuators, several magnetic heads, several

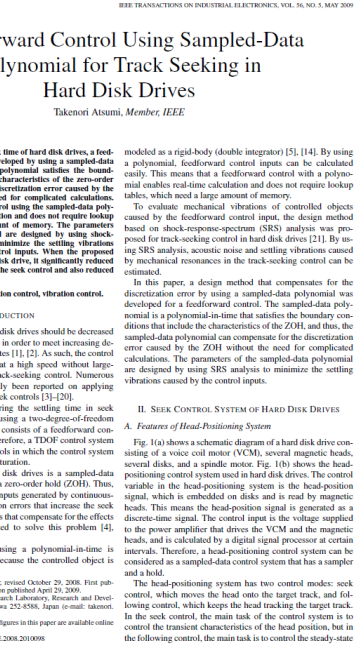
Manuscript received April 8, 2019; revised July 18, 2019; accepted November 5, 2019. Date of publication November 28, 2019; date of current version July 14, 2020. *(Corresponding author: Takenori Atsumi.)*

T. Atsumi is with the Department of Mechanical Engineering, Chiba Institute of Technology, Narashino 275-0016, Japan (e-mail: takenori.atsumi@p.chibakoudai.jp).

S. Yabui is with the Department of Mechanical Systems Engineering, School of Engineering, Nagoya University, Nagoya 464-8601, Japan (e-mail: yabui@nuem.nagoya-u.ac.jp).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2019.2955432



1338 IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 56, NO. 5, MAY 2009

Feedforward Control Using Sampled-Data Polynomial for Track Seeking in Hard Disk Drives

Takenori Atsumi, *Member, IEEE*

***Abstract*—To decrease the seek time of hard disk drives, a feedforward control method was developed by using a sampled-data polynomial. The sampled-data polynomial satisfies the boundary conditions that include the characteristics of the zero-order hold, and compensates for the discretization error caused by the zero-order hold without the need for complicated calculations. Therefore, the feedforward control using the sampled-data polynomial enables real-time calculation and does not require lookup tables which need a large amount of memory. The parameters of the sampled-data polynomial are designed by using shock-response-spectrum analysis to minimize the settling vibrations caused by the feedforward control inputs. When the proposed method was applied on a hard disk drive, it significantly reduced the amount of tracking error in the seek control and also reduced the seek time.**

***Index Terms*—Hard disks, motion control, vibration control.**

I. Introduction

THE SEEK TIME OF hard disk drives should be decreased to reduce data access time in order to meet increasing demand for higher data transfer rates [1], [2]. As such, the control system has to move the head at a high speed without large-magnitude vibrations in the track-seeking control. Numerous research efforts have previously been reported on applying advanced control methods to seek controls [3]–[20].

The transient vibrations during the settling time in seek controls can be decreased by using a two-degree-of-freedom (TDOF) control system, which consists of a feedforward control and a feedback control. Therefore, a TDOF control system is used in short-span seek controls in which the control system has no possibility of actuator saturation.

The control system in hard disk drives is a sampled-data system that has a sampler and a zero-order hold (ZOH). Thus, using the feedforward control inputs generated by continuous-time design causes discretization errors that increase the seek time. The TDOF control systems that compensate for the effects of the ZOH have been reported to solve this problem [4], [8]–[10].

A TDOF control system using a polynomial-in-time is widely used in seek controls because the controlled object is modeled as a rigid-body (double integrator) [5], [14]. By using a polynomial, feedforward control inputs can be calculated easily. This means that a feedforward control with a polynomial enables real-time calculation and does not require lookup tables, which need a large amount of memory.

To evaluate mechanical vibrations of controlled objects caused by the feedforward control input, the design method based on shock-response-spectrum (SRS) analysis was proposed for track-seeking control in hard disk drives [21]. By using SRS analysis, acoustic noise and settling vibrations caused by mechanical resonances in the track-seeking control can be estimated.

In this paper, a design method that compensates for the discretization error by using a sampled-data polynomial was developed for a feedforward control. The sampled-data polynomial is a polynomial-in-time that satisfies the boundary conditions that include the characteristics of the ZOH, and thus, the sampled-data polynomial can compensate for the discretization error caused by the ZOH without the need for complicated calculations. The parameters of the sampled-data polynomial are designed by using SRS analysis to minimize the settling vibrations caused by the control inputs.

II. Seek Control System of Hard Disk Drives

A. Features of Head-Positioning System

Fig. 1(a) shows a schematic diagram of a hard disk drive consisting of a voice coil motor (VCM), several magnetic heads, several disks, and a spindle motor. Fig. 1(b) shows the head-positioning control system used in hard disk drives. The control variable in the head-positioning system is the head-position signal, which is embedded on disks and is read by magnetic heads. This means the head-position signal is generated as a discrete-time signal. The control input is the voltage supplied to the power amplifier that drives the VCM and the magnetic heads, and is calculated by a digital signal processor at certain intervals. Therefore, a head-positioning control system can be considered as a sampled-data control system that has a sampler and a hold.

The head-positioning system has two control modes: seek control, which moves the head onto the target track, and following control, which keeps the head tracking the target track. In the seek control, the main task of the control system is to control the transient characteristics of the head position, but in the following control, the main task is to control the steady-state

Manuscript received April 19, 2008; revised October 29, 2008. First published December 2, 2008; current version published April 29, 2009.

The author is with the Central Research Laboratory, Research and Development Group, Hitachi, Ltd., Kanagawa 252-8588, Japan (e-mail: takenori.atsumi.ta@hitachi.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2008.2010098

In this benchmark problem, we provide design examples for representative seek spans. To select the MAT-file corresponding to the seek span you want to simulate, choose a line between lines 17 and 25 in “Simulation_trackseek.m”. You can select it by removing the % at the beginning of the desired line.

Simulation_trackseek.m

```
1   clc;clear
2
3   %% Switch for disturbances
4   SW_dist=1;    % Disable: 0, Enable: 1
5   SW_source=1;  % Fixed: 0 (for evaluation), Unfixed: 1 (for learning)
6
7   %% Cotrolled object
8   Plant
9
10  %% Feedback Controller
11  load Data_Cd.mat
12
13  %% Multi-rate filter
14  load Data_Fm.mat
15
16  %% FF data
17  %load Data_FF_4.mat     % Seek-span is 4
18  %load Data_FF_8.mat     % Seek-span is 8
19  %load Data_FF_16.mat    % Seek-span is 16
20  %load Data_FF_32.mat    % Seek-span is 32
21  load Data_FF_64.mat    % Seek-span is 64
22  %load Data_FF_128.mat    % Seek-span is 128
23  %load Data_FF_256.mat    % Seek-span is 256
24  %load Data_FF_512.mat    % Seek-span is 512
25  %load Data_FF_1024.mat   % Seek-span is 1024
26
```

Results of “Simulation_trackseek” with the Example Controller

Simulation condition

- Seek span: 64
- SW_dist: 1 (with disturbance condition)
- SW_source: 0 (fixed source)

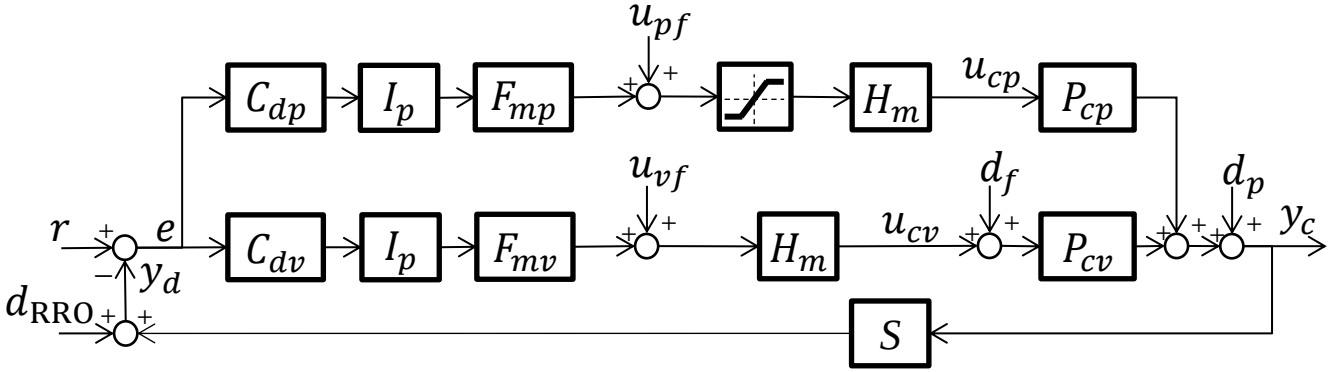


Figure 1

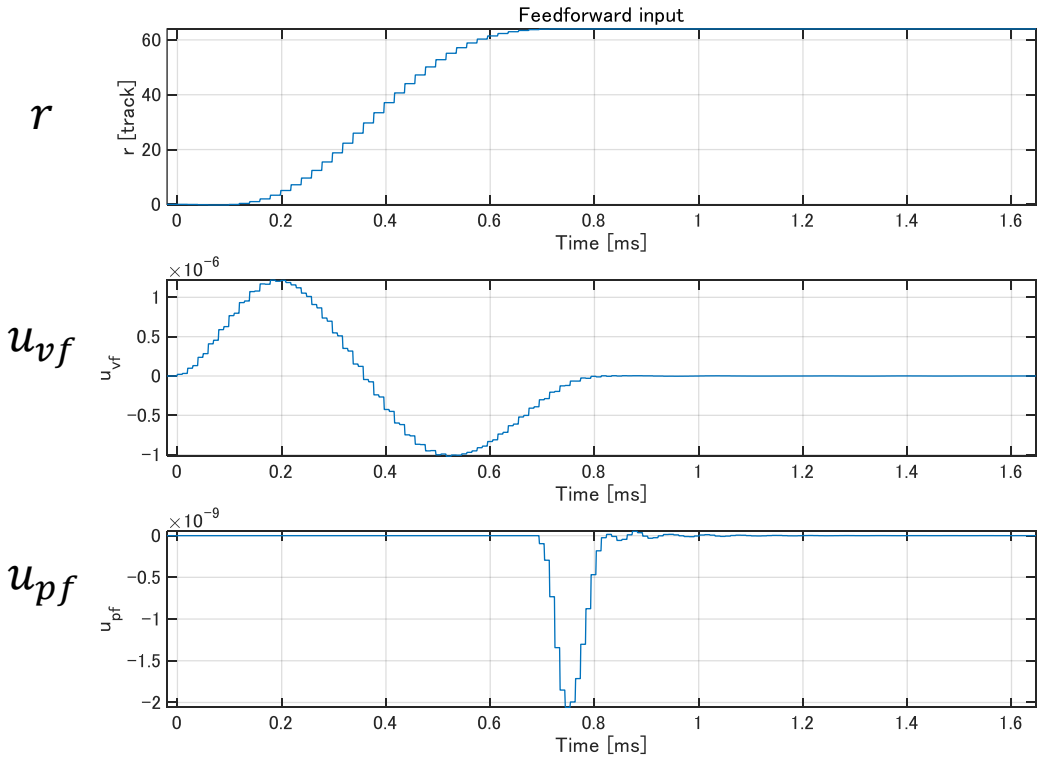
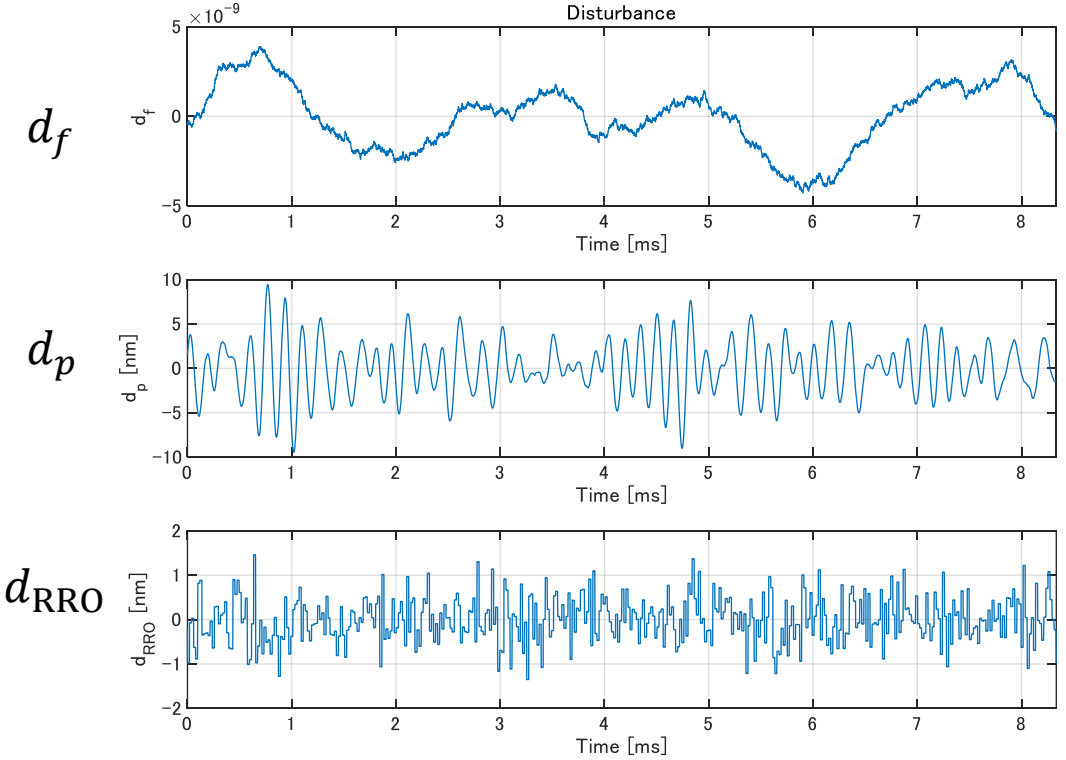


Figure 2



Results of “Simulation_trackseek” with the Example Controller (Cont.)

Simulation condition

- Seek span: 64
- SW_dist: 1 (with disturbance condition)
- SW_source: 0 (fixed source)

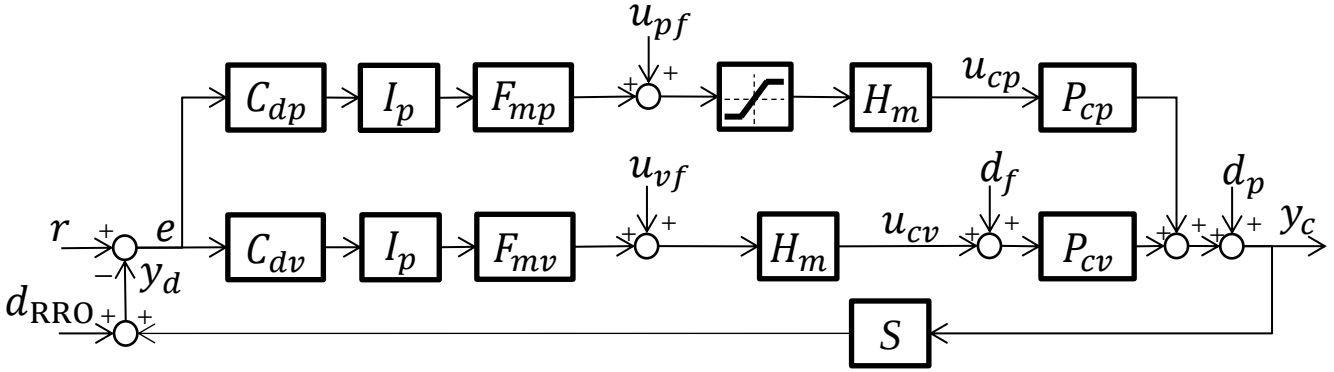


Figure 3 (e)

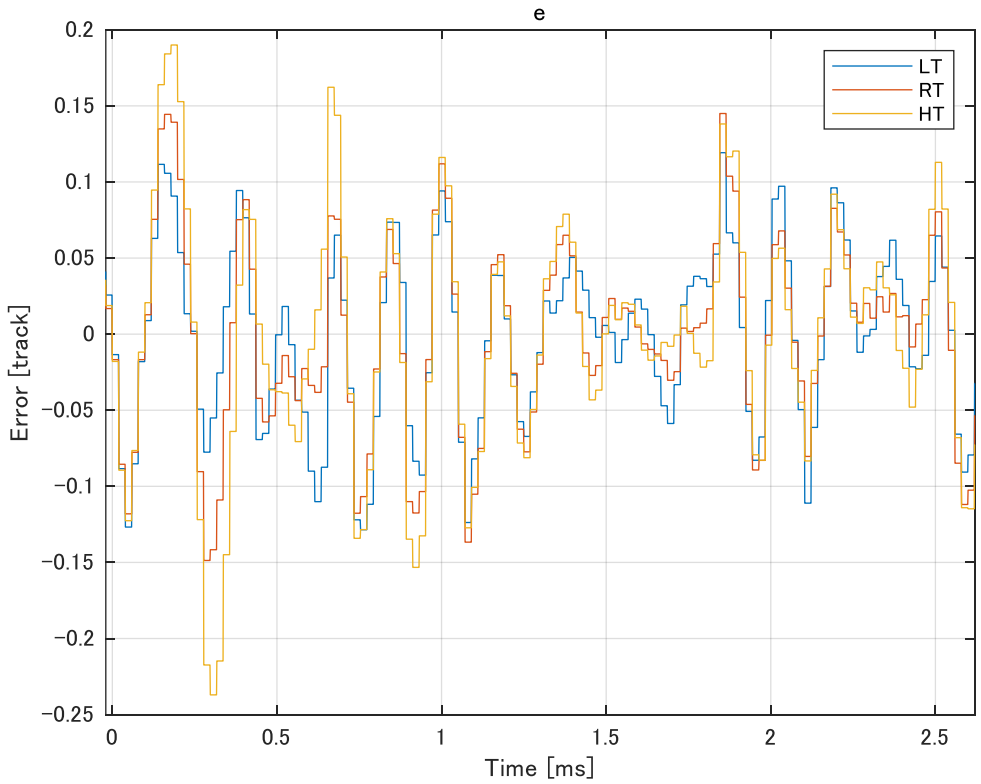
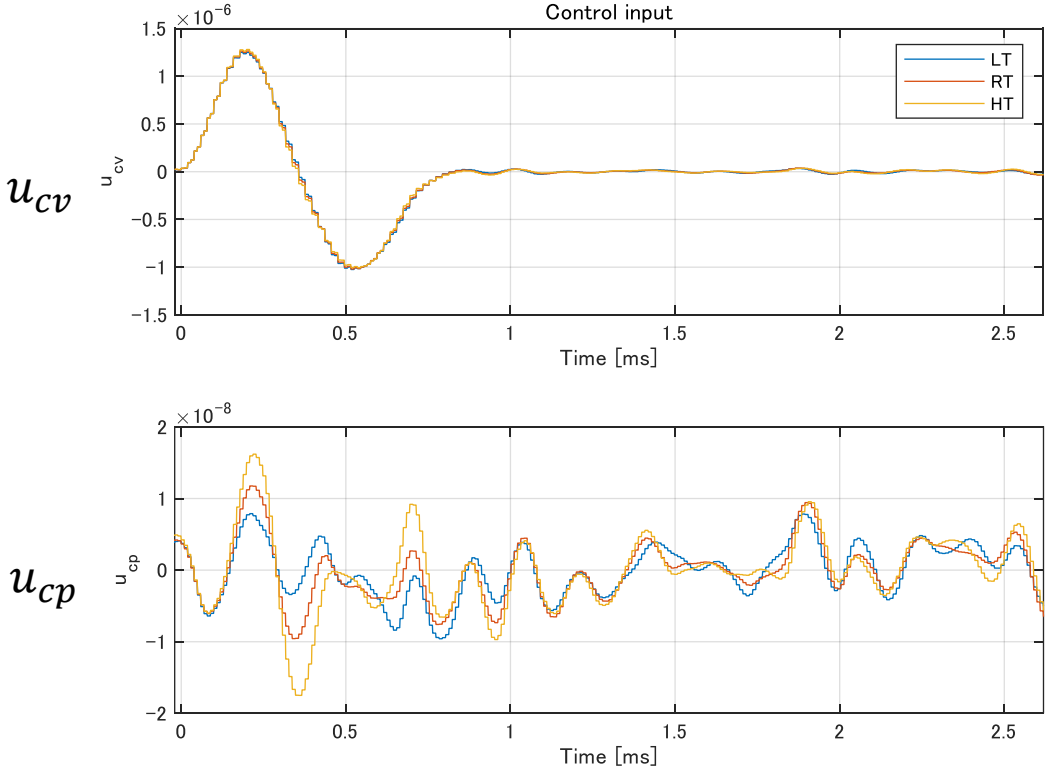


Figure 4



Results of “Simulation_trackseek” with the Example Controller (Cont.)

Simulation condition

- Seek span: 64
- SW_dist: 1 (with disturbance condition)
- SW_source: 0 (fixed source)

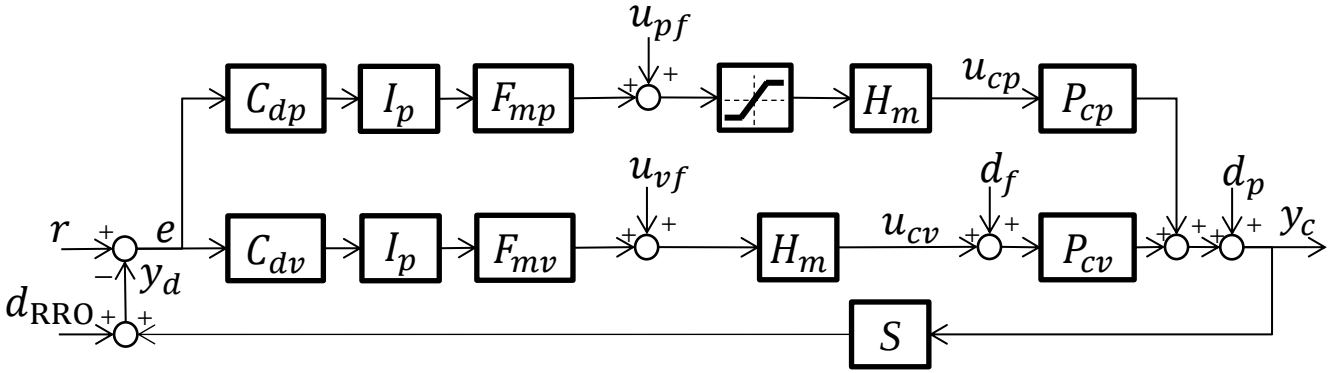


Figure 5 ($y_c$and r: Overall vier)

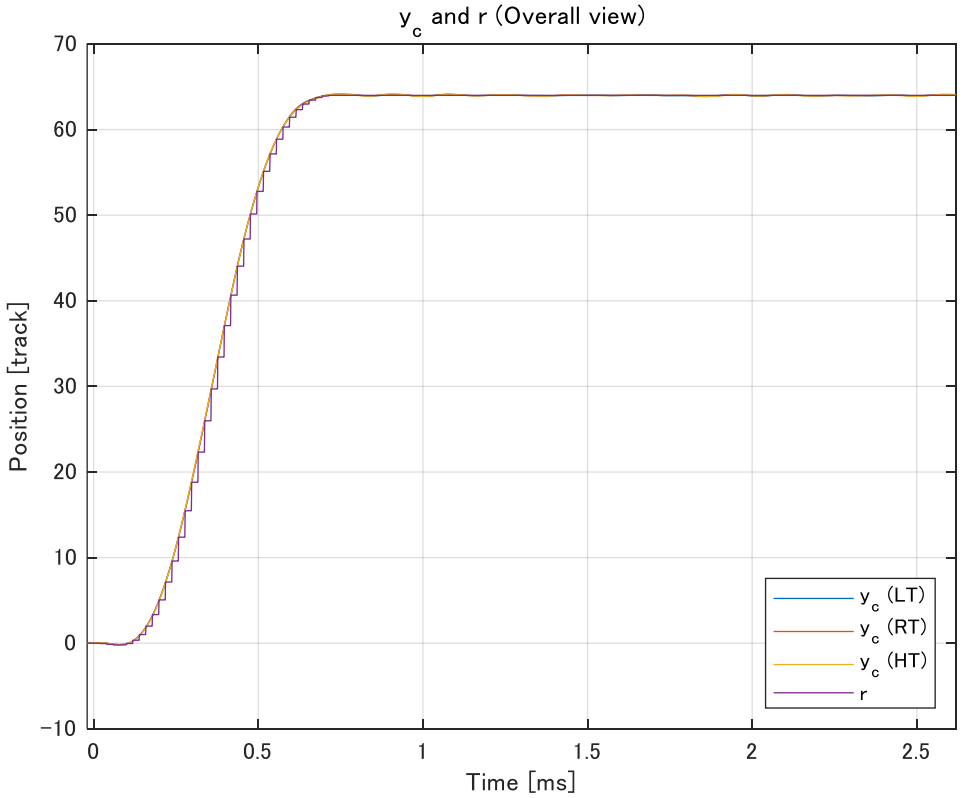
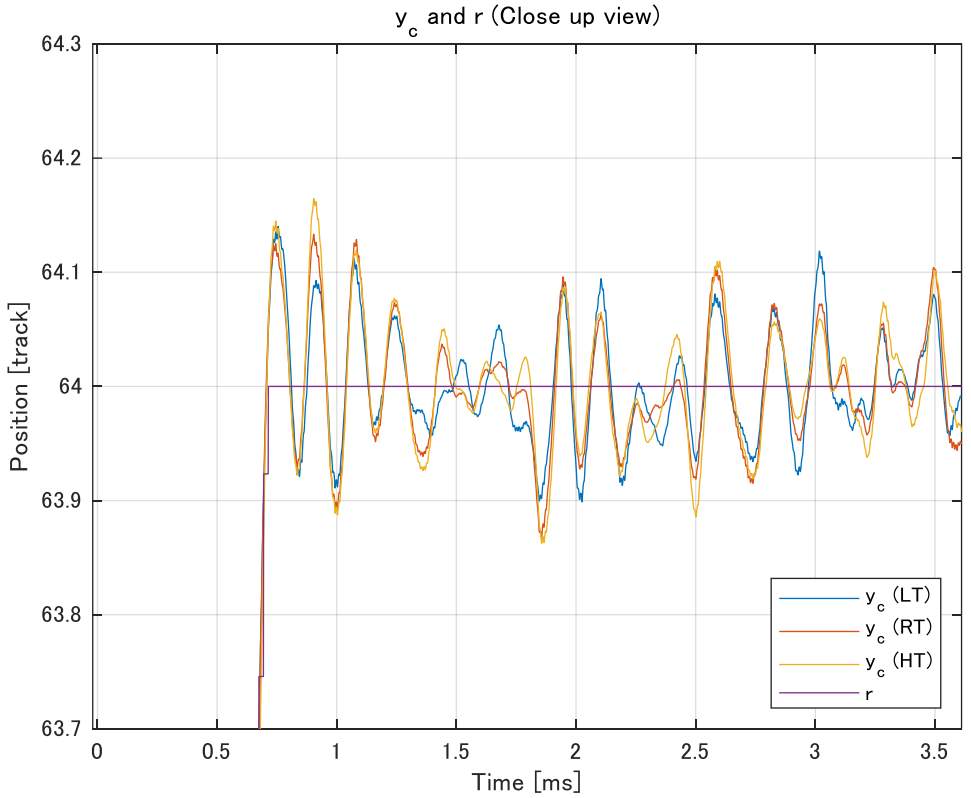


Figure 6 ($y_c$and r: Close up view)



- Control system in the benchmark problem
- How to use the benchmark problem
- Example
- MATLAB code

Files in the Benchmark Problem

Main-Program Code

1. Simulation_trackseek.m: This M-file executes the track-seek simulation with the given feedforward signals (r, u_{fv}, and u_{fp}) and controllers (C_{dv}, C_{dp}, F_{mv}, and F_{mp}) in "Data_FF_xx.mat", "Data_Cd.mat" and "Data_Fm.mat".
2. Plot_control_system.m: This M-file shows the frequency responses of the control system with the given controllers (C_{dv}, C_{dp}, F_{mv}, and F_{mp}) in "Data_Cd.mat" and "Data_Fm.mat".

Sub-Program Code

1. Plant.m: This M-file includes parameters of the control system and sets the dynamic system models of the controlled objects. This code is used in "Simulation_trackseek.m" and "Plot_control_system.m".
2. Function_simulation_seek.m: This M-file is a function file for the track-seek simulation.

Data Files

1. Data_FF_xxx.mat: This MAT file includes feedforward signals (r: r_table, u_{fv}: uvf_table, and u_{fp}: upf_table).
2. Data_Cd.mat: This MAT file includes feedback controllers (C_{dv}: Sys_Cdv, C_{dp}: Sys_Cdp).
3. Data_Fm.mat: This MAT file includes multi-rate filters (F_{mv}: Sys_Fmv, F_{mp}: Sys_Fmp).
4. RROdata.txt: This text file includes the RRO data in time domain.

Usage of “Function_simulation_seek”

This function is used in “Simulation_trackseek.m” (from lines 29 to 33).

Definition

```
Function sim_result=Function_simulation_seek(Sys_Pcv,Sys_Pcp,Sys_Cdv,Sys_Fmv,Sys_Cdp,Sys_Fmp,r_table,uvf_table,
                    upf_table,SW_dist,SW_source)
```

OUTPUT

sim_result: Simulation results of track-seek control with the input system.

INPUT

Sys_Pcv: Model object of P_{cv} (VCM)
Sys_Pcp: Model object of P_{cp} (PZT actuator)
Sys_Cdv: Model object of C_{dv} (Feedback controller for VCM)
Sys_Fmv: Model object of F_{mv} (Multi-rate filter for VCM)
Sys_Cdp: Model object of C_{dp} (Feedback controller for PZT actuator)
Sys_Fmp: Model object of F_{mp} (Multi-rate filter for PZT actuator)
r_table: Look-up table of Feedforward input for r
uvf_table: Look-up table of Feedforward input for u_{vf}
upf_table: Look-up table of Feedforward input for u_{pf}
SW_dist: Switch for disturbance (Disable: 0, Enable: 1)
SW_source: Switch for disturbance source (Fixed: 0 (for evaluation), Unfixed: 1 (for learning))

Code description: Function_simulation_seek.m

```
function sim_result=Function_simulation_seek(Sys_Pcv,Sys_Pcp,Sys_Cdv,Sys_Fmv,Sys_Cdp,Sys_Fmp,r_table,uvf_table,upf_table,SW_dist,SW_source)

% Switch for disturbances
if SW_dist~=1
   SW_dist=0;
end

%% Sampling time and Multi-rate number
num_sector=420;                 % Number of sector
num_rpm=7200;                   % Number of RPM
Ts = 1/(num_rpm/60*num_sector);   % Sampling time
Mr_f=2;                         % Multi-rate number

%% Laplace operator
s=tf('s');

%% Simulation condition
Mr_p=20;                        % Multi-rate for continuous-time system
Tsc=Ts/Mr_p;                    % Sampling time for continuous-time system
Tsim=0.3;                       % End of simulation time

%% Controlled object
Sys_Pcdv=c2d(ssbal(ss(Sys_Pcv)),Tsc);[A_Pcdv,B_Pcdv,C_Pcdv,~]=ssdata(ssbal(Sys_Pcdv));
Sys_Pcdp=c2d(ssbal(ss(Sys_Pcp)),Tsc);[A_Pcdp,B_Pcdp,C_Pcdp,~]=ssdata(ssbal(Sys_Pcdp));

%% Feedback Controller & Multi-rate filter
[A_Cdv,B_Cdv,C_Cdv,D_Cdv]=ssdata(ssbal(ss(Sys_Cdv)));
[A_Fmv,B_Fmv,C_Fmv,D_Fmv]=ssdata(ssbal(ss(Sys_Fmv)));
[A_Cdp,B_Cdp,C_Cdp,D_Cdp]=ssdata(ssbal(ss(Sys_Cdp)));
[A_Fmp,B_Fmp,C_Fmp,D_Fmp]=ssdata(ssbal(ss(Sys_Fmp)));
```

Define disturbance condition

Define sampling time and multi-rate number for control system

Define Laplace operator

Define approximate continuous-time systems

Define state-space matrixes for controlled objects

Define state-space matrixes for controller and multi-rate filters

Code description: Function_simulation_seek.m (Cont.)

```
%% Disturbance signal
% FAN-Induced Disturbance
Sys_Dp1_c=0.6/(s^2+2*0.008*2200*2*pi*s+(2200*2*pi)^2);Sys_Dp1=c2d(ssbal(ss(Sys_Dp1_c)),Tsc);[A_Dp1,B_Dp1,C_Dp1,~]=ssdata(ssbal(Sys_Dp1));
Sys_Dp2_c=0.3/(s^2+2*0.005*2937*2*pi*s+(2937*2*pi)^2);Sys_Dp2=c2d(ssbal(ss(Sys_Dp2_c)),Tsc);[A_Dp2,B_Dp2,C_Dp2,~]=ssdata(ssbal(Sys_Dp2));
Sys_Dp3_c=1/(s^2+2*0.005*3300*2*pi*s+(3300*2*pi)^2);Sys_Dp3=c2d(ssbal(ss(Sys_Dp3_c)),Tsc);[A_Dp3,B_Dp3,C_Dp3,~]=ssdata(ssbal(Sys_Dp3));
Sys_Dp4_c=0.5/(s^2+2*0.005*3545*2*pi*s+(3545*2*pi)^2);Sys_Dp4=c2d(ssbal(ss(Sys_Dp4_c)),Tsc);[A_Dp4,B_Dp4,C_Dp4,~]=ssdata(ssbal(Sys_Dp4));
Sys_Dp5_c=0.3/(s^2+2*0.002*3980*2*pi*s+(3980*2*pi)^2);Sys_Dp5=c2d(ssbal(ss(Sys_Dp5_c)),Tsc);[A_Dp5,B_Dp5,C_Dp5,~]=ssdata(ssbal(Sys_Dp5));
Sys_Dp6_c=1/(s^2+2*0.01*4220*2*pi*s+(4220*2*pi)^2);Sys_Dp6=c2d(ssbal(ss(Sys_Dp6_c)),Tsc);[A_Dp6,B_Dp6,C_Dp6,~]=ssdata(ssbal(Sys_Dp6));
Sys_Dp7_c=1/(s^2+2*0.008*4380*2*pi*s+(4380*2*pi)^2);Sys_Dp7=c2d(ssbal(ss(Sys_Dp7_c)),Tsc);[A_Dp7,B_Dp7,C_Dp7,~]=ssdata(ssbal(Sys_Dp7));
Sys_Dp8_c=0.5/(s^2+2*0.002*5072*2*pi*s+(5072*2*pi)^2);Sys_Dp8=c2d(ssbal(ss(Sys_Dp8_c)),Tsc);[A_Dp8,B_Dp8,C_Dp8,~]=ssdata(ssbal(Sys_Dp8));
Sys_Dp9_c=2/(s^2+2*0.003*5370*2*pi*s+(5370*2*pi)^2);Sys_Dp9=c2d(ssbal(ss(Sys_Dp9_c)),Tsc);[A_Dp9,B_Dp9,C_Dp9,~]=ssdata(ssbal(Sys_Dp9));
Sys_Dp10_c=15/(s^2+2*0.04*5850*2*pi*s+(5850*2*pi)^2);Sys_Dp10=c2d(ssbal(ss(Sys_Dp10_c)),Tsc);[A_Dp10,B_Dp10,C_Dp10,~]=ssdata(ssbal(Sys_Dp10));
Sys_Dp11_c=10/(s^2+2*0.008*6660*2*pi*s+(6660*2*pi)^2);Sys_Dp11=c2d(ssbal(ss(Sys_Dp11_c)),Tsc);[A_Dp11,B_Dp11,C_Dp11,~]=ssdata(ssbal(Sys_Dp11));
Sys_Dp12_c=1.5/(s^2+2*0.003*7670*2*pi*s+(7670*2*pi)^2);Sys_Dp12=c2d(ssbal(ss(Sys_Dp12_c)),Tsc);[A_Dp12,B_Dp12,C_Dp12,~]=ssdata(ssbal(Sys_Dp12));
Sys_Dp13_c=2.5/(s^2+2*0.07*9200*2*pi*s+(9200*2*pi)^2);Sys_Dp13=c2d(ssbal(ss(Sys_Dp13_c)),Tsc);[A_Dp13,B_Dp13,C_Dp13,~]=ssdata(ssbal(Sys_Dp13));
```

Define fan-induced vibrations

```
% RRO
RRO_data=readmatrix("Data_RRO.txt")*0.5e-10;
```

Define RRO

```
% Rotational vibration
Sys_Df_c=3e-10*(s+50*2*pi)/(s+3*2*pi)*(s^2+2*20*2000*2*pi*s+(2000*2*pi)^2)/(s^2+2*0.1*250*2*pi*s+(250*2*pi)^2);
Sys_Df=c2d(ssbal(ss(Sys_Df_c)),Tsc);
[A_Df,B_Df,C_Df,~]=ssdata(ssbal(Sys_Df));
```

Define RV

```
% Random signal for disturbance
if SW_source~=1
    rng(1);u_random1=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(2);u_random2=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(3);u_random3=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(4);u_random4=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(5);u_random5=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(6);u_random6=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(7);u_random7=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(8);u_random8=(rand(round(Tsim/Tsc)+1,1)-0.5);
```

Define fixed noise sources for evaluation

Code description: Function_simulation_seek.m (Cont.)

```
 rng(9);u_random9=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(10);u_random10=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(11);u_random11=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(12);u_random12=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(13);u_random13=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng(14);u_random14=(rand(round(Tsim/Tsc)+1,1)-0.5);
else
    rng('shuffle');u_random1=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random2=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random3=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random4=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random5=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random6=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random7=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random8=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random9=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random10=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random11=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random12=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random13=(rand(round(Tsim/Tsc)+1,1)-0.5);
    rng('shuffle');u_random14=(rand(round(Tsim/Tsc)+1,1)-0.5);
end

%% Simulation
% Initial value
x_Cdv=zeros(length(A_Cdv),1);x_Fmv=zeros(length(A_Fmv),1);
x_Cdp=zeros(length(A_Cdp),1);x_Fmp=zeros(length(A_Fmp),1);
x_Df=zeros(length(A_Df),1);
dx_Dp1=zeros(length(A_Dp1),1);dx_Dp2=zeros(length(A_Dp2),1);dx_Dp3=zeros(length(A_Dp3),1);dx_Dp4=zeros(length(A_Dp4),1);dx_Dp5=zeros(le
ngth(A_Dp5),1);dx_Dp6=zeros(length(A_Dp6),1);dx_Dp7=zeros(length(A_Dp7),1);dx_Dp8=zeros(length(A_Dp8),1);dx_Dp9=zeros(length(A_Dp9),1);
dx_Dp10=zeros(length(A_Dp10),1);dx_Dp11=zeros(length(A_Dp11),1);dx_Dp12=zeros(length(A_Dp12),1);dx_Dp13=zeros(length(A_Dp13),1);
```

Define fixed noise sources for evaluation

Define unfixed noise sources for learning method

Define initial values for simulation

Code description: Function_simulation_seek.m (Cont.)

```
ddp=0;
dx_Pcdv=zeros(length(A_Pcdv),1);dy_Pcdv=0;
dx_Pcdp=zeros(length(A_Pcdp),1);dy_Pcdp=0;
dyc=0;
i_RRO=1;

% Initialize variables
t=zeros(1,round(Tsim/Tsc)+1);
dRRO=zeros(1,round(Tsim/Tsc)+1);
df=zeros(1,round(Tsim/Tsc)+1);
dp=zeros(1,round(Tsim/Tsc)+1);
udv=zeros(1,round(Tsim/Tsc)+1);
udp=zeros(1,round(Tsim/Tsc)+1);
ucv=zeros(1,round(Tsim/Tsc)+1);
ucp=zeros(1,round(Tsim/Tsc)+1);
y_Pcdv=zeros(1,round(Tsim/Tsc)+1);
y_Pcdp=zeros(1,round(Tsim/Tsc)+1);
yc=zeros(1,round(Tsim/Tsc)+1);
yd=zeros(1,round(Tsim/Tsc)+1);
r=zeros(1,round(Tsim/Tsc)+1);
e=zeros(1,round(Tsim/Tsc)+1);
uvf=zeros(1,round(Tsim/Tsc)+1);
upf=zeros(1,round(Tsim/Tsc)+1);
```

Define initial values for simulation

```
% Number of inter-sampling
nn_s=Mr_p;
nn_m=Mr_p/Mr_f;
```

Define number of intersampling for multi-rate sampled-data control system.

```
% Number of samples for track-seek
nd=0; % for single-rate
ndm=0; % for multi-rate
```

Define sample number of time during track-seek control.

Code description: Function_simulation_seek.m (Cont.)

```
% Simulation
for n=1:round(Tsim/Tsc)+1
  t(n)=Tsc*(n-1);

  % Discrete-time system (Single-rate)
  if nn_s==Mr_p
    nn_s=0;

    % RRO
    dRRO(n)=RRO_data(i_RRO);
    if i_RRO == num_sector
       i_RRO=1;
    else
       i_RRO=i_RRO+1;
    end

    % Measured magnetic-head position
    yd(n)=dyc+dRRO(n)*SW_dist;

    % Feedforard input for target position
    if t(n) < 0.2
      r(n)=0;
    else
      if nd >= length(r_table)
        r(n)=r_table(end);
      else
        r(n)=r_table(nd+1);
      end
      nd=nd+1;
    end

    % Tracking error
    e(n)= r(n)-yd(n);
```

Start of for-loop simulation

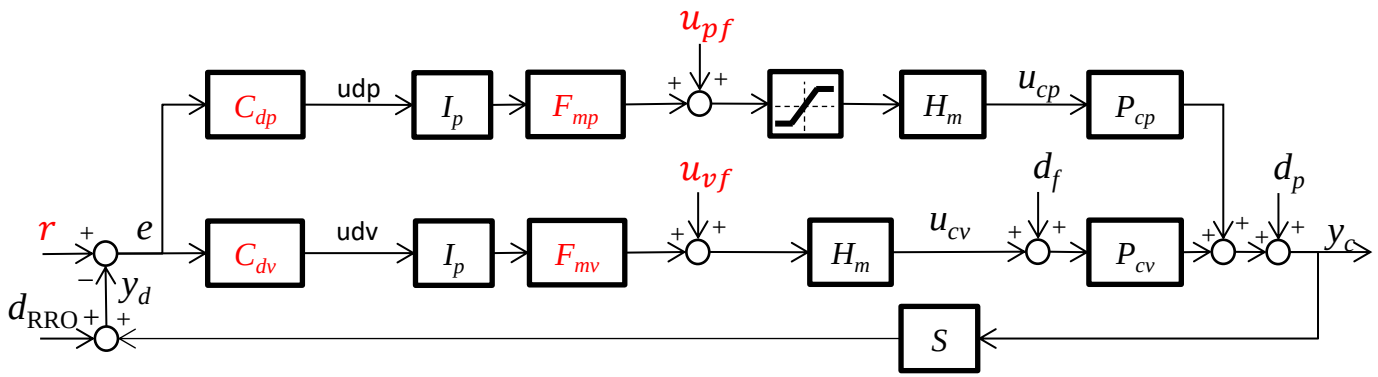
Start of calculation for single-rate digital filter (C_{dv}, C_{dp})

Set d_{RRO}

Set y_d

Set r

Set e



Code description: Function_simulation_seek.m (Cont.)

```
    % for VCM
    y_Cdv=C_Cdv*x_Cdv+D_Cdv*e(n);
    x_Cdv=A_Cdv*x_Cdv+B_Cdv*e(n);
    udv(n)=y_Cdv;

% for PZT
    y_Cdp=C_Cdp*x_Cdp+D_Cdp*e(n);
    x_Cdp=A_Cdp*x_Cdp+B_Cdp*e(n);
    udp(n)=y_Cdp;
 else
    dRRO(n)=dRRO(n-1);
    yd(n)=yd(n-1);
    r(n)=r(n-1);
    e(n)=e(n-1);
    udv(n)=udv(n-1);
    udp(n)=udp(n-1);
 end
```

Calculate C_{dv}

Calculate C_{dp}

If you employ controllers that are not LTI systems, you can rewrite this part yourself.

Set d_{RRO}, y_d, e, udv, udp, and r, during inter sampling.

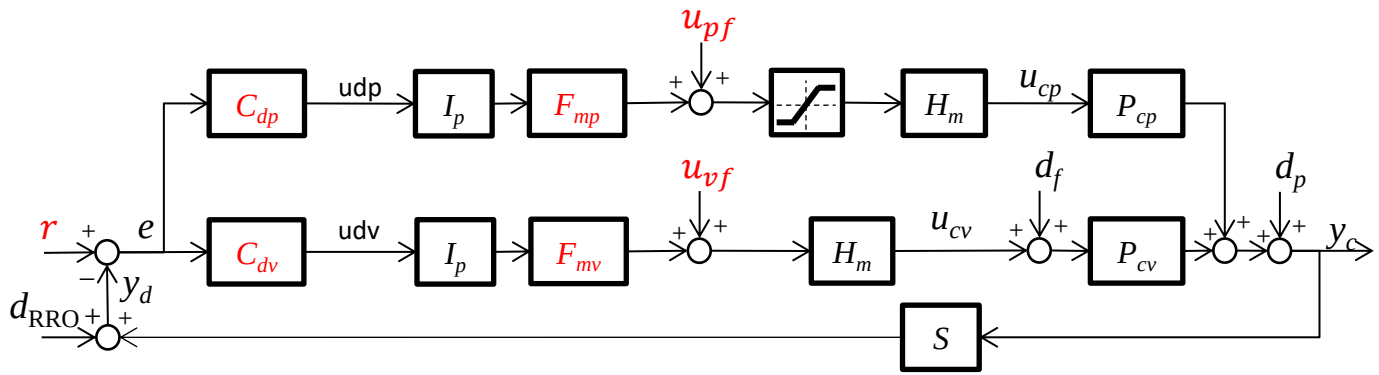
End of calculation for single-rate digital filter (C_{dv}, C_{dp})

```
% Discrete-time system (Multi-rate)
if nn_m==Mr_p/Mr_f
   nn_m=0;

   % Feedforard input for control inputs
   if nd==0
      uvf(n)=0;upf(n)=0;
   else
      if ndm >= length(uvf_table)
         uvf(n)=uvf_table(end);
         upf(n)=upf_table(end);
      else
         uvf(n)=uvf_table(ndm+1);
         upf(n)=upf_table(ndm+1);
      end
      ndm=ndm+1;
   end
```

Start of calculation for multi-rate digital filter (F_{mv}, F_{mp})

Set u_{vf} and u_{pf}.



Code description: Function_simulation_seek.m (Cont.)

```
% for VCM
y_Fmv=C_Fmv*x_Fmv+D_Fmv*(udv(n));
x_Fmv=A_Fmv*x_Fmv+B_Fmv*(udv(n));
ucv(n)=y_Fmv+uvf(n); % FB+FF

% for PZT
y_Fmp=C_Fmp*x_Fmp+D_Fmp*udp(n);
x_Fmp=A_Fmp*x_Fmp+B_Fmp*udp(n);
ucp(n)=y_Fmp+upf(n); % FB+FF
```

Calculate F_{mv}

Calculate F_{mp}

If you employ controllers that are not LTI systems, you can rewrite this part yourself.

```
% PZT saturation
if abs(ucp(n))>50e-9
  ucp(n)=sign(ucp(n))*50e-9;
end
else
  ucv(n)=ucv(n-1);
  ucp(n)=ucp(n-1);
  uvf(n)=uvf(n-1);
  upf(n)=upf(n-1);
end
```

Set PZT saturation (50×10^{-9})

Set u_{cv}, u_{cp}, u_{vf}, and u_{pf}, during inter sampling.

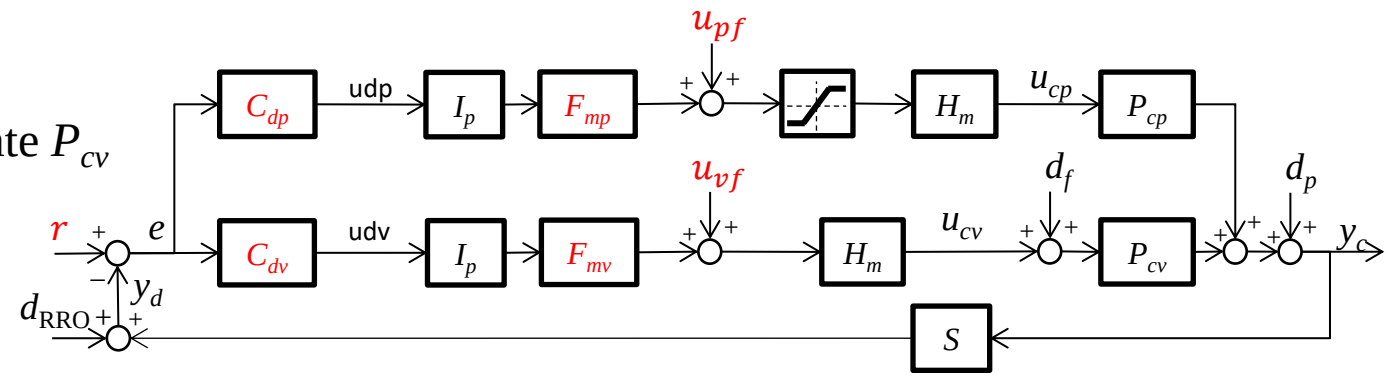
End of calculation for multi-rate digital filter (F_{mv}, F_{mp})

```
% RV disturbance
df(n)=C_Df*x_Df;
x_Df=A_Df*x_Df+B_Df*u_random1(n);
```

Calculate d_f (RV)

```
% Continuous-time system (VCM)
x_Pcdv=dx_Pcdv;
y_Pcdv(n)=dy_Pcdv;
dx_Pcdv=A_Pcdv*x_Pcdv+B_Pcdv*(ucv(n)+df(n)*SW_dist);
dy_Pcdv=C_Pcdv*dx_Pcdv;
```

Calculate P_{cv}



Code description: Function_simulation_seek.m (Cont.)

```
% Continuous-time system (PZT)
x_Pcdp=dx_Pcdp;
y_Pcdp(n)=dy_Pcdp;
dx_Pcdp=A_Pcdp*x_Pcdp+B_Pcdp*ucp(n);
dy_Pcdp=C_Pcdp*dx_Pcdp;
```

Calculate P_{cp}

```
% FAN-induced disturbance
x_Dp1=dx_Dp1;dx_Dp1=A_Dp1*x_Dp1+B_Dp1*u_random2(n);dy_Dp1=C_Dp1*dx_Dp1;
x_Dp2=dx_Dp2;dx_Dp2=A_Dp2*x_Dp2+B_Dp2*u_random3(n);dy_Dp2=C_Dp2*dx_Dp2;
x_Dp3=dx_Dp3;dx_Dp3=A_Dp3*x_Dp3+B_Dp3*u_random4(n);dy_Dp3=C_Dp3*dx_Dp3;
x_Dp4=dx_Dp4;dx_Dp4=A_Dp4*x_Dp4+B_Dp4*u_random5(n);dy_Dp4=C_Dp4*dx_Dp4;
x_Dp5=dx_Dp5;dx_Dp5=A_Dp5*x_Dp5+B_Dp5*u_random6(n);dy_Dp5=C_Dp5*dx_Dp5;
x_Dp6=dx_Dp6;dx_Dp6=A_Dp6*x_Dp6+B_Dp6*u_random7(n);dy_Dp6=C_Dp6*dx_Dp6;
x_Dp7=dx_Dp7;dx_Dp7=A_Dp7*x_Dp7+B_Dp7*u_random8(n);dy_Dp7=C_Dp7*dx_Dp7;
x_Dp8=dx_Dp8;dx_Dp8=A_Dp8*x_Dp8+B_Dp8*u_random9(n);dy_Dp8=C_Dp8*dx_Dp8;
x_Dp9=dx_Dp9;dx_Dp9=A_Dp9*x_Dp9+B_Dp9*u_random10(n);dy_Dp9=C_Dp9*dx_Dp9;
x_Dp10=dx_Dp10;dx_Dp10=A_Dp10*x_Dp10+B_Dp10*u_random11(n);dy_Dp10=C_Dp10*dx_Dp10;
x_Dp11=dx_Dp11;dx_Dp11=A_Dp11*x_Dp11+B_Dp11*u_random12(n);dy_Dp11=C_Dp11*dx_Dp11;
x_Dp12=dx_Dp12;dx_Dp12=A_Dp12*x_Dp12+B_Dp12*u_random13(n);dy_Dp12=C_Dp12*dx_Dp12;
x_Dp13=dx_Dp13;dx_Dp13=A_Dp13*x_Dp13+B_Dp13*u_random14(n);dy_Dp13=C_Dp13*dx_Dp13;
dp(n)=ddp;
ddp = dy_Dp1+dy_Dp2+dy_Dp3+dy_Dp4+dy_Dp5+dy_Dp6+dy_Dp7+dy_Dp8+dy_Dp9+dy_Dp10+dy_Dp11+dy_Dp12+dy_Dp13;
```

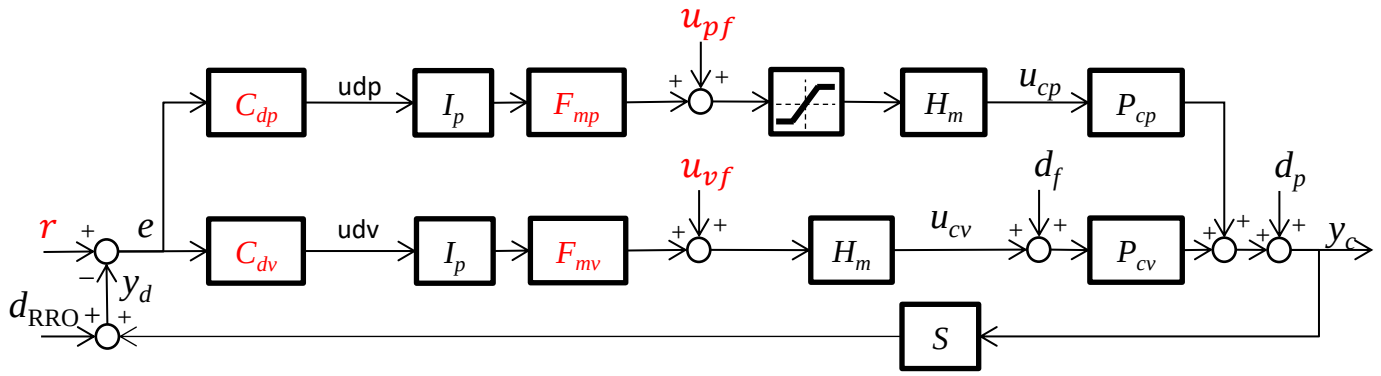
Calculate d_p (fan-induced vibe.)

```
% Magnetic-head position
yc(n)=dyc;
dyc=dy_Pcdv+dy_Pcdp+ddp*SW_dist;
```

Set y_c

```
nn_s=nn_s+1;nn_m=nn_m+1;
end
```

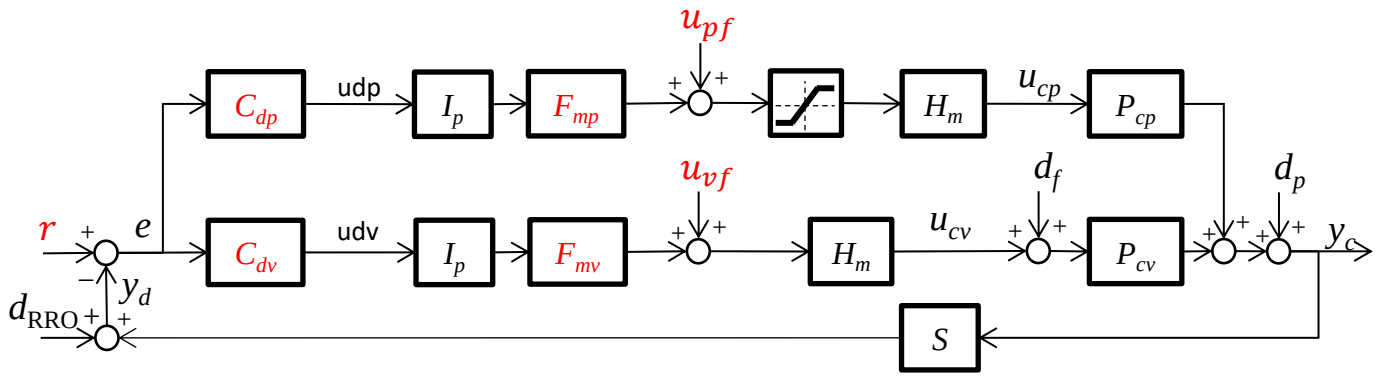
End of for-loop simulation



Code description: Function_simulation_seek.m (Cont.)

```
% Pick up simulation results during track-seek time
Nt=find(t>=0.1,1);
Ns=find(abs(uvf)>0,1);
sim_result.time=t(Nt:Nt+Mr_p*num_sector*24-1)-t(Ns);
sim_result.ucv=ucv(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.ucp=ucp(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.yc=yc(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.yd=yd(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.dp=dp(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.df=df(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.dRRO=dRRO(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.e=e(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.r=r(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.uvf=uvf(Nt:Nt+Mr_p*num_sector*24-1);
sim_result.upf=upf(Nt:Nt+Mr_p*num_sector*24-1);
```

Pick up simulation results during track-seek time



END

Benchmark problem for magnetic-head positioning control in HDD ~ Track-seek control ~

User manual

Investigating R&D Committee on “Basis of Collaborative Technologies for Precision Servo Systems”
The Institute of Electrical Engineers of Japan