

8. 상태 변수 방정식을 이용한 제어 시스템 해석

8.8 실험 실습: Lab 8

이 실험에서는 4장의 4.7.2절과 5장의 5.5.1절에서 사용한 아날로그 다이내믹 시뮬레이터의 상태 변수 방정식을 세우고, 간단한 피드백 제어 시스템을 구성해서 상태 변수들의 동작 특성을 관찰한다. 먼저 그림 8-6과 같이 상태 변수를 정의한다.

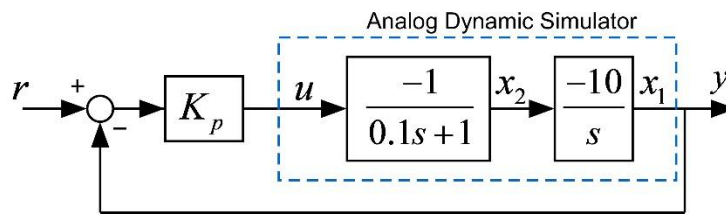


그림 8-6 아날로그 다이내믹 시뮬레이터의 상태 변수

그림 8-6에서 상태 변수 x_1 은 출력단 OP 앰프의 출력 신호이며, 상태 변수 x_2 는 입력단 OP 앰프의 출력 신호이다. $X_1(s)$, $X_2(s)$, $U(s)$ 를 각각 $x_1(t)$, $x_2(t)$, $u(t)$ 의 라플라스 변환이라고 정의하면 다음과 같은 관계식을 얻을 수 있다.

$$\frac{X_1(s)}{X_2(s)} = \frac{-10}{s} \quad (8.140)$$

$$\frac{X_2(s)}{U(s)} = \frac{-1}{0.1s+1} \quad (8.141)$$

위의 관계식을 다시 정리해서 다음과 같은 식을 얻는다.

$$sX_1(s) = -10X_2(s) \quad (8.142)$$

$$0.1sX_2(s) + X_2(s) = -U(s) \quad (8.143)$$

위의 식에서 다음의 상태 변수 방정식을 얻을 수 있다.

$$\dot{x}_1(t) = -10x_2(t) \quad (8.144)$$

$$\dot{x}_2(t) = -10x_2(t) - 10u(t) \quad (8.145)$$

위의 식은 다음과 같은 벡터 형태의 상태 변수 방정식으로 나타낼 수 있다.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & -10 \\ 0 & -10 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -10 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$
(8.146)

위의 다이내믹 시뮬레이터는 불안정한 시스템이므로 응답을 관찰하기 위해서는 피드백 제어기에 의해서 안정화해야 한다. 다음과 같은 간단한 형태의 P 제어를 적용해서 계단 응답을 관찰한다. 아래의 식에서 $r(t)$ 는 기준 입력 신호이다.

$$u(t) = K_p (r(t) - y(t))$$
(8.147)

위의 P 제어를 적용하면 상태 변수 방정식은 다음과 같다.

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & -10 \\ 0 & -10 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -10 \end{bmatrix} K_p \left(r(t) - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \right)$$
(8.148)

아래의 코드는 P 제어가 적용된 다이내믹 시뮬레이터의 시뮬레이션을 하는 MATLAB 코드이다.

코드 8.1

```
A=[0 -10;0 -10];B=[0;-10];C=[1 0];D=0;
Kp=5;
t=0:0.001:2;
G=ss(A,B,C,D);
Gc=ss(A-B*Kp*C,Kp*B,C,D);
R=ones(size(t));
[y,t,x]=lsim(Gc,R,t);
subplot(2,1,1)
plot(t,x(:,1))
grid
axis([0 2 0 2])
ylabel('x1');
subplot(2,1,2)
plot(t,x(:,2))
grid
axis([0 2 -2 2])
ylabel('x2');xlabel('time(sec)');
```

그림 8-7은 위의 MATLAB 코드를 실행한 결과이며, 기준 입력이 계단 입력 신호일 때 각 상태 변수의 그래프이다.

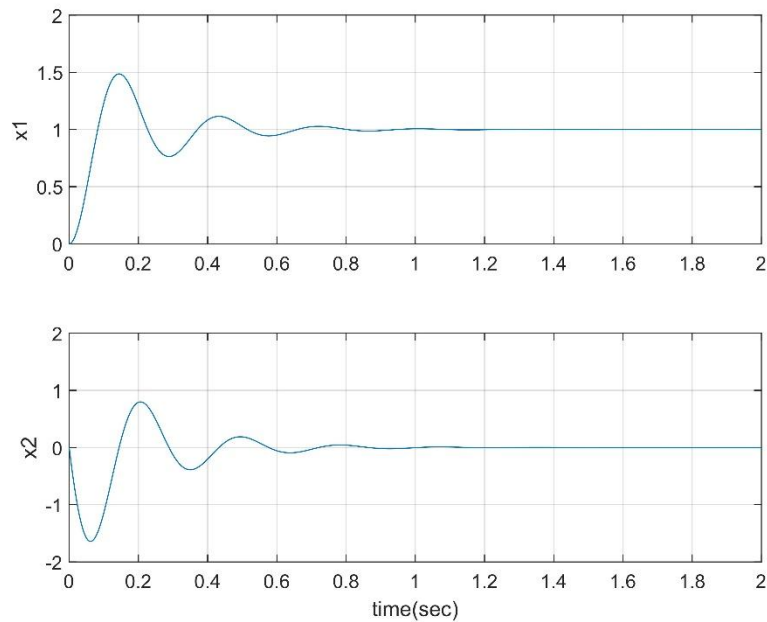


그림 8-7 MATLAB 시뮬레이션에 의한 상태 변수 그래프

다음으로 다이내믹 시뮬레이터에 P 제어를 적용해서 다이내믹 시뮬레이터의 상태 변수를 관찰해 본다. STM32CubeIDE의 설정은 5장의 [5.5.1 아날로그 다이내믹 시뮬레이터 대한 PD 제어기](#)와 거의 같지만, 이 실험에서는 x_2 변수를 입력하기 위한 A/D 변환기 채널을 한 개 더 추가한다. 5장의 5.5.1절과 동일한 과정으로 설정한 후, 추가로 그림 8-8과 같이 ADC3을 활성화 한다. ADC2의 입력 핀은 다른 용도로 사용되므로 사용할 수 없다.

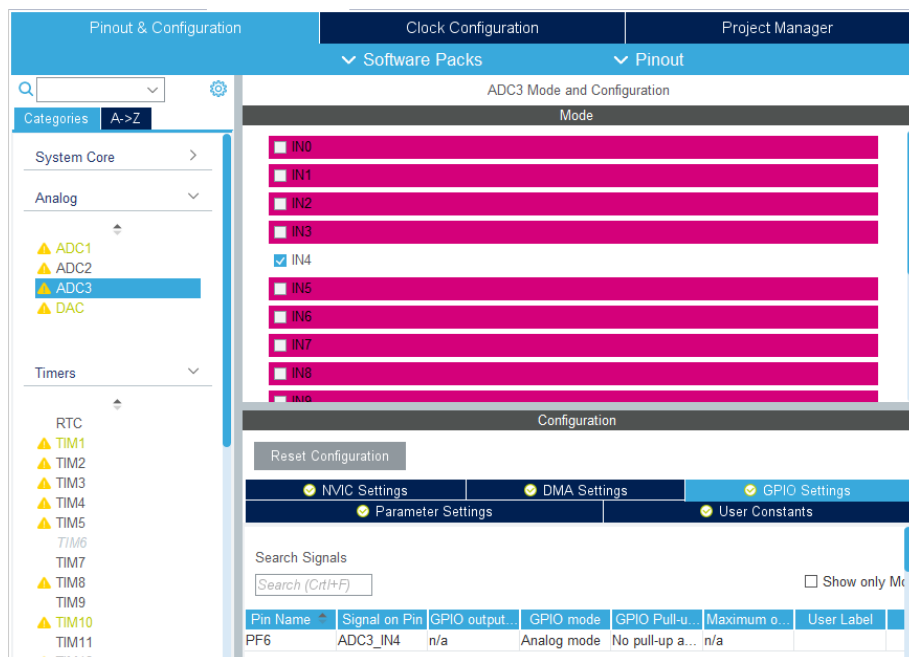


그림 8-8 ADC3 활성화

코드 5.2, 코드 5.4, 코드 5.6, 코드 5.8을 각각 코드 8.2, 코드 8.3, 코드 8.4, 코드 8.5로 대체한다. 추가된 코드의 내용은 2개의 상태 변수를 캡처해서 저장하기 위한 코드이다.

코드 8.2

```
/* USER CODE BEGIN PV */
float Kp,control;
volatile int32_t x1,x2,ref,interrupt_counter,sampling_frequency,data_counter;
int16_t data[4000],data2[4000];
volatile uint8_t data_flag=0,data_done=0;
/* USER CODE END PV */
```

코드 8.3

```
/* USER CODE BEGIN 1 */
Kp=5.0;
sampling_frequency=1000;
/* USER CODE END 1 */
```

코드 8.4

```
/* USER CODE BEGIN WHILE */
while (1)
{
    if(data_done == 1) {
        for (int i=0; i < sampling_frequency*4 ;i++){
            printf("%d %d %d\r\n",i,data[i],data2[i]);
        }
        data_flag=0;
        data_done=0;
        HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_14);
    }
}
/* USER CODE END WHILE */
```

코드 8.5

```
/* USER CODE BEGIN Callback 0 */
int32_t da_value,ad_value,sum;
if (htim->Instance == TIM10) {
    sum=0;
    for (int i=0; i<20 ; i++) {
        HAL_ADC_Start(&hadc1);
        if (HAL_ADC_PollForConversion(&hadc1, 10000) == HAL_OK) {
            ad_value = HAL_ADC_GetValue(&hadc1);
            sum += ad_value;
        }
    }
    x1 = sum/20 - 2048;
    sum=0;
    for (int i=0; i<20 ; i++) {
        HAL_ADC_Start(&hadc3);
        if (HAL_ADC_PollForConversion(&hadc3, 10000) == HAL_OK) {
            ad_value = HAL_ADC_GetValue(&hadc3);
            sum += ad_value;
        }
    }
}
```

```

    }
    x2 = sum/20 - 2048;
    interrupt_counter++;
    if (interrupt_counter >= sampling_frequency*4) {
        interrupt_counter=0;
        if (data_flag==1) {
            data_counter=0;
            data_flag=2;
        }
        ref=205;
    }
    if (interrupt_counter >= sampling_frequency*2) {
        ref=0;
    }
    if (data_flag==2) {
        if (data_counter<sampling_frequency*4) {
            data[data_counter]=(int16_t)x1;
            data2[data_counter]=(int16_t)x2;
            data_counter++;
        }
        else {
            data_done=1;
        }
    }
    control = Kp*(float)(ref-x1);
    if (control > 2047) control = 2047;
    if (control < -2048) control = -2048;
    da_value = control + 2048;
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_2, DAC_ALIGN_12B_R,
(uint32_t)(da_value));
}
/* USER CODE END Callback 0 */

```

아래의 MATLAB 코드는 상태 변수 캡처한 $x_1(t)$, $x_2(t)$ 를 그래프로 그리는 코드이다.

코드 8.6

```

clear
sf=1000;
load data
for i=1:4*sf
    t(i)=data(i,1)/sf;
    x1(i)=data(i,2)*10/2048;
    x2(i)=data(i,3)*10/2048;
end
subplot(2,1,1);
plot(t,x1)
axis([0 2 0 2])
ylabel('x1');
grid on
subplot(2,1,2);
plot(t,x2)
axis([0 2 -2 2])
ylabel('x2');xlabel('time(sec)');

```

grid on

그림 8-9는 아날로그 다이내믹 시뮬레이터의 상태 변수 응답을 보여준다.

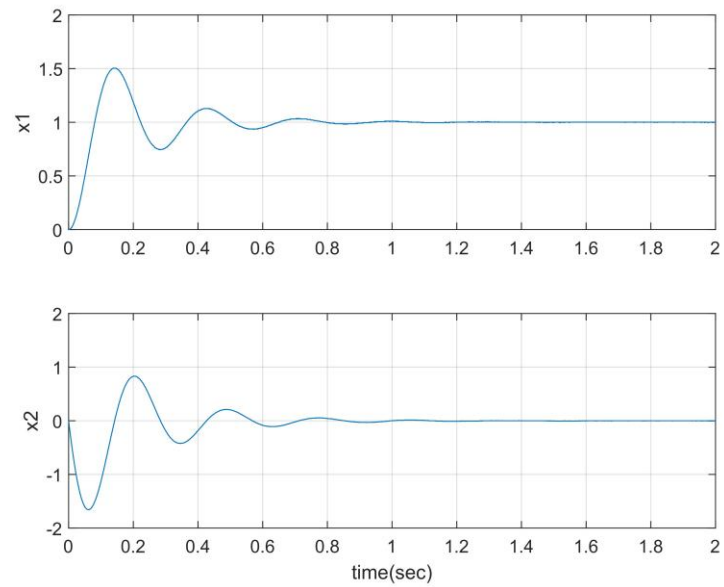


그림 8-9 아날로그 다이내믹 시뮬레이터의 상태 변수 응답