

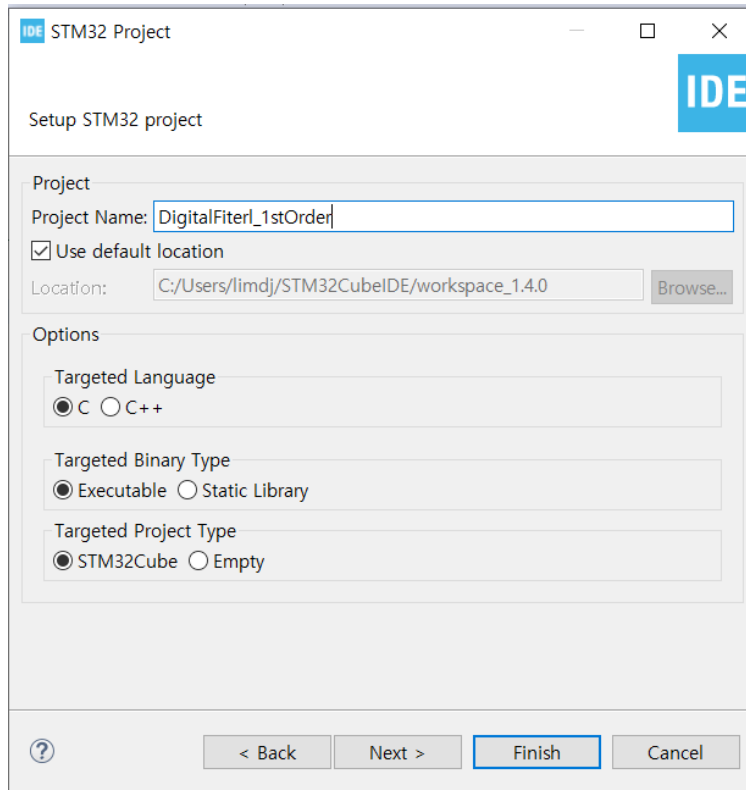
# Lab 2

*Implementation of Digital Filter using Cortex-M4 Board*



# New STM32 Project

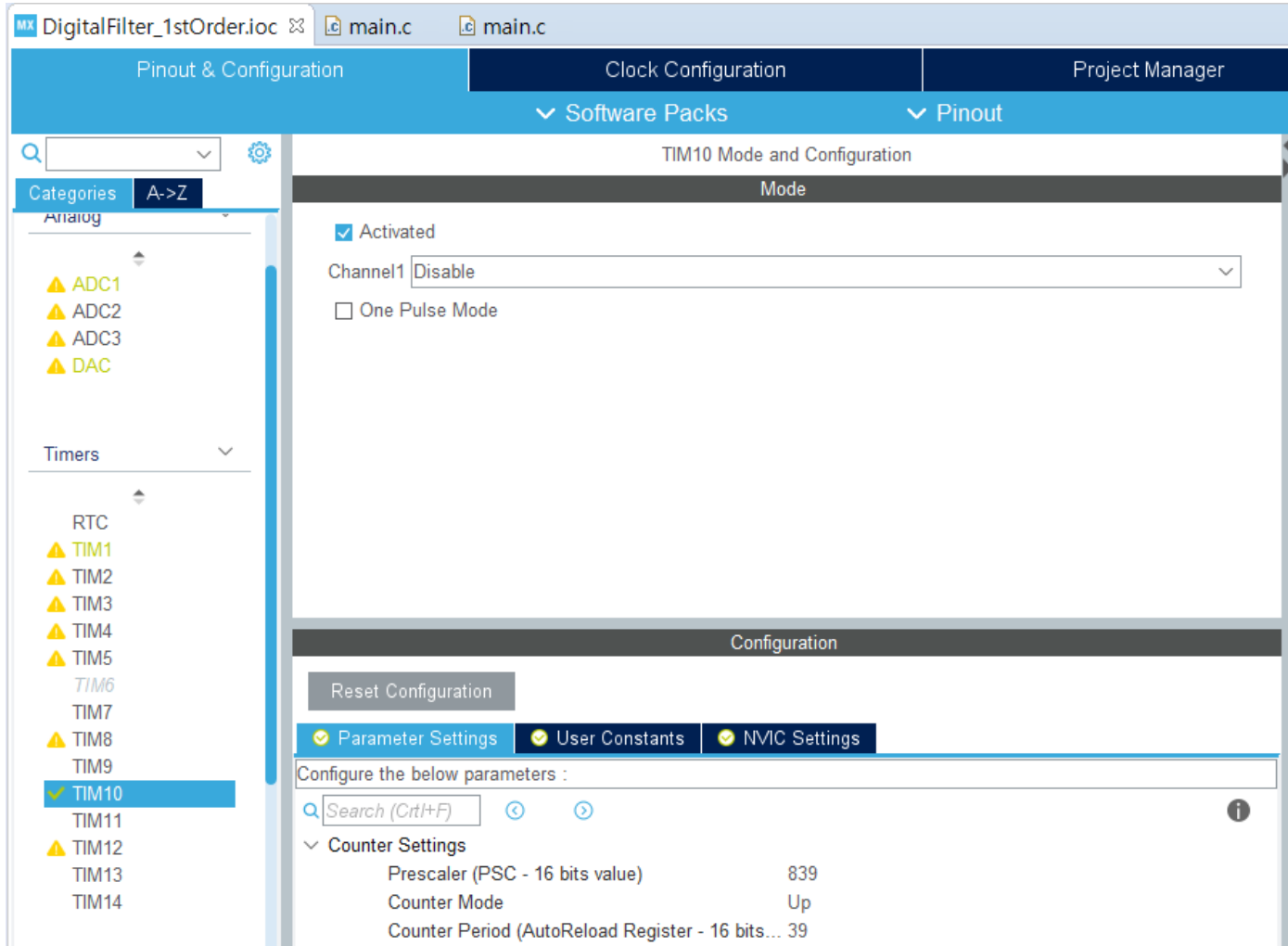
- Project Name: DigitalFilter\_1stOrder



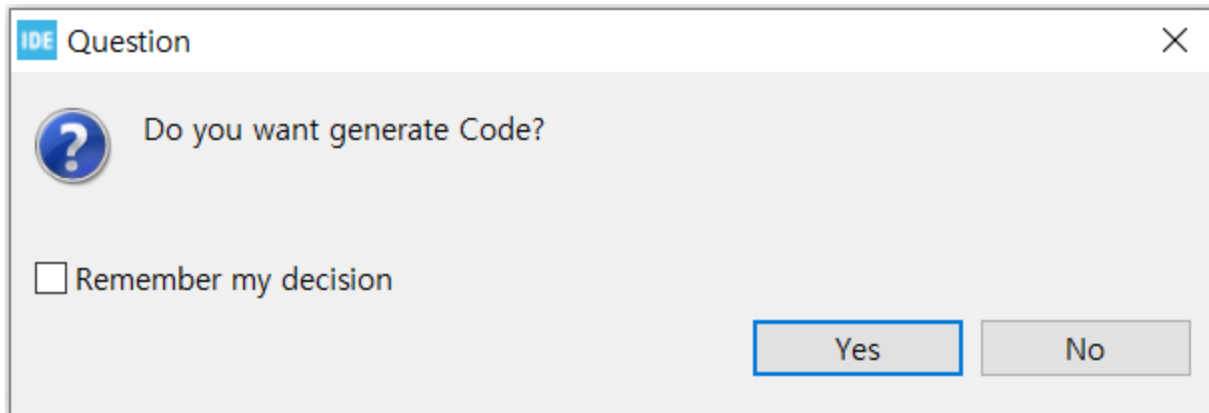
- Clock Configuration: HCLK 168Mhz
- Disable: FREERTOS

# Sampling Frequency: 5KHz

■  $168,000,000 / (840 * 40) = 5000$



■ Enable Timer10 Interrupt



# A/D D/A Code

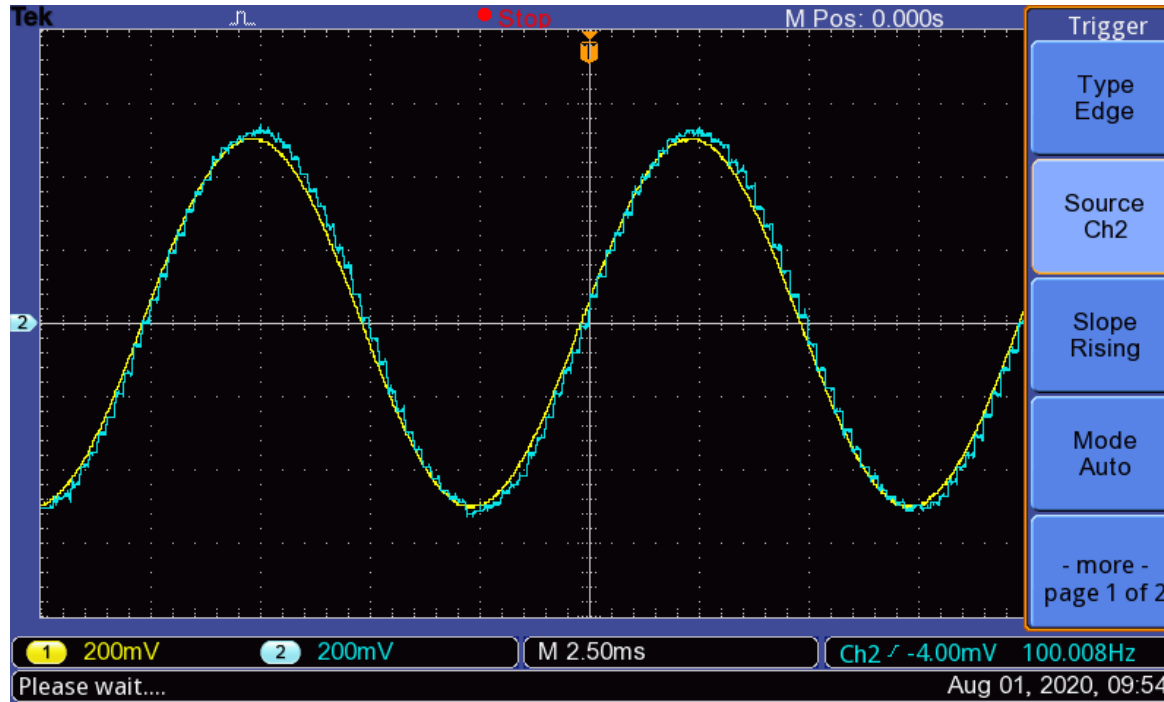
---

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim10);
HAL_DAC_Start(&hdac, DAC_CHANNEL_2);
/* USER CODE END 2 */

/* USER CODE BEGIN Callback 0 */
int da_value,ad_value;
if (htim->Instance == TIM10) {
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, 10000) == HAL_OK) {
        ad_value = HAL_ADC_GetValue(&hadc1);
    }
    da_value = ad_value;
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_2, DAC_ALIGN_12B_R, (uint32_t)(da_value));
}
/* USER CODE END Callback 0 */
```

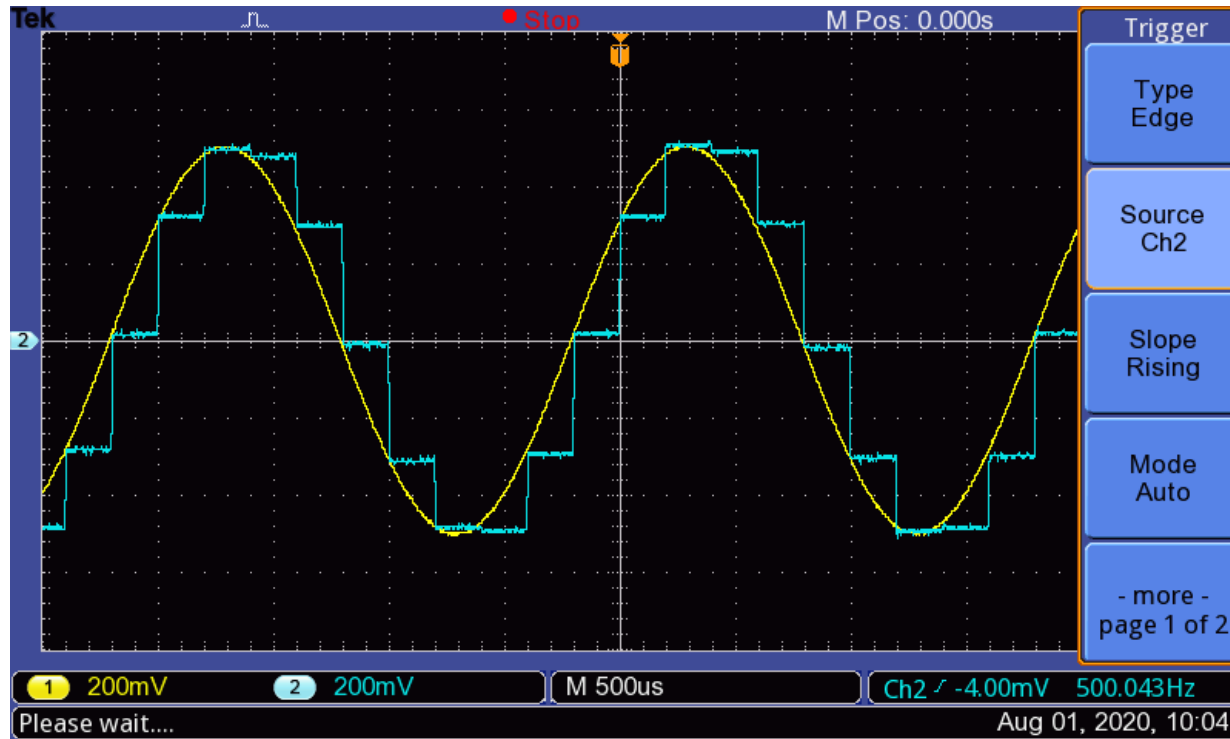
# Sampling Frequency: 5KHz

- Function Generator: 100Hz sine wave



# Sampling Frequency: 5KHz

- Function Generator: 500Hz sine wave



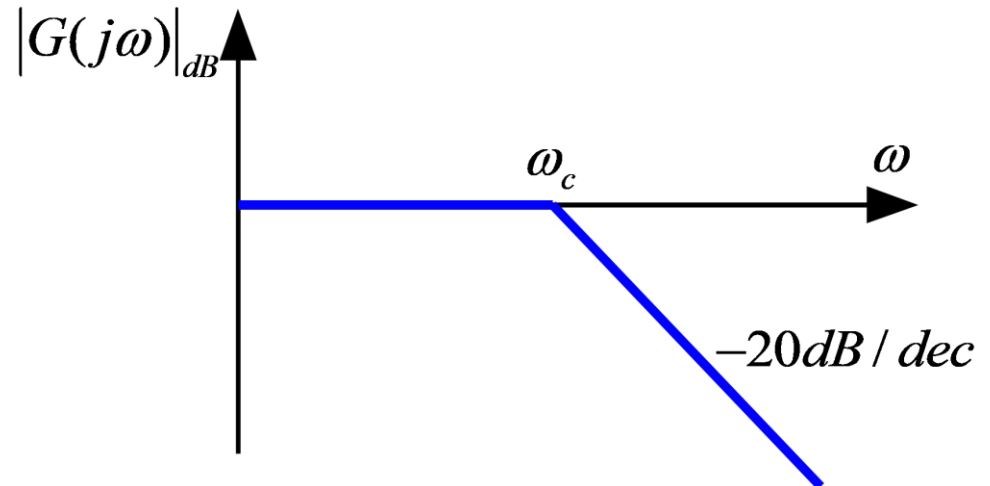
# 1st Order Digital Filter

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{1 + s / \omega_c} = \frac{\omega_c}{\omega_c + s}$$

$$sY(s) + \omega_c Y(s) = \omega_c U(s)$$

$$\dot{y}(t) + \omega_c y(t) = \omega_c u(t)$$

$$\dot{y}(t) \approx \frac{y(t) - y(t - \Delta t)}{\Delta t}$$



# Digital Approximation

---

$$\frac{y(t) - y(t - \Delta t)}{\Delta t} + \omega_c y(t) = \omega_c u(t)$$

$$y(t) - y(t - \Delta t) + \Delta t \omega_c y(t) = \Delta t \omega_c u(t)$$

$$(1 + \Delta t \omega_c) y(t) = y(t - \Delta t) + \Delta t \omega_c u(t)$$

$$y(t) = \frac{1}{1 + \Delta t \omega_c} (y(t - \Delta t) + \Delta t \omega_c u(t))$$

# A/D D/A Code

```
/* USER CODE BEGIN 0 */  
float delt, wc, u, y, oldy;  
/* USER CODE END 0 */  
  
/* USER CODE BEGIN 1 */  
delt=0.0002; wc=100*2*3.14;  
/* USER CODE END 1 */  
  
/* USER CODE BEGIN 2 */  
HAL_TIM_Base_Start_IT(&htim10);  
HAL_DAC_Start(&hdac, DAC_CHANNEL_2);  
/* USER CODE END 2 */
```

$$f_c = 100\text{Hz}$$

$$\omega_c = 2\pi \cdot 100(\text{rad} / \text{sec})$$

$$\Delta t = 1 / 5000$$

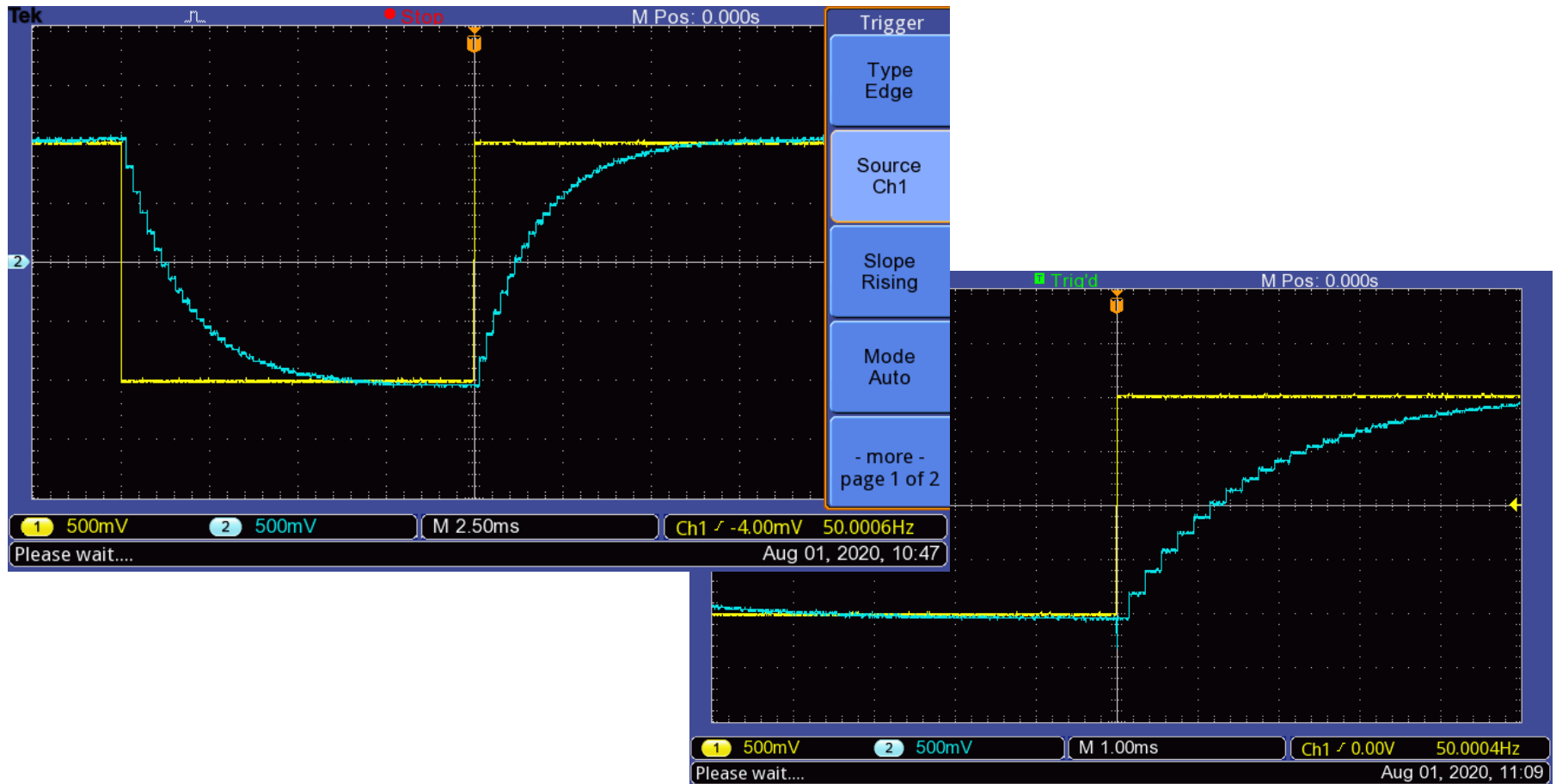
# A/D D/A Code

---

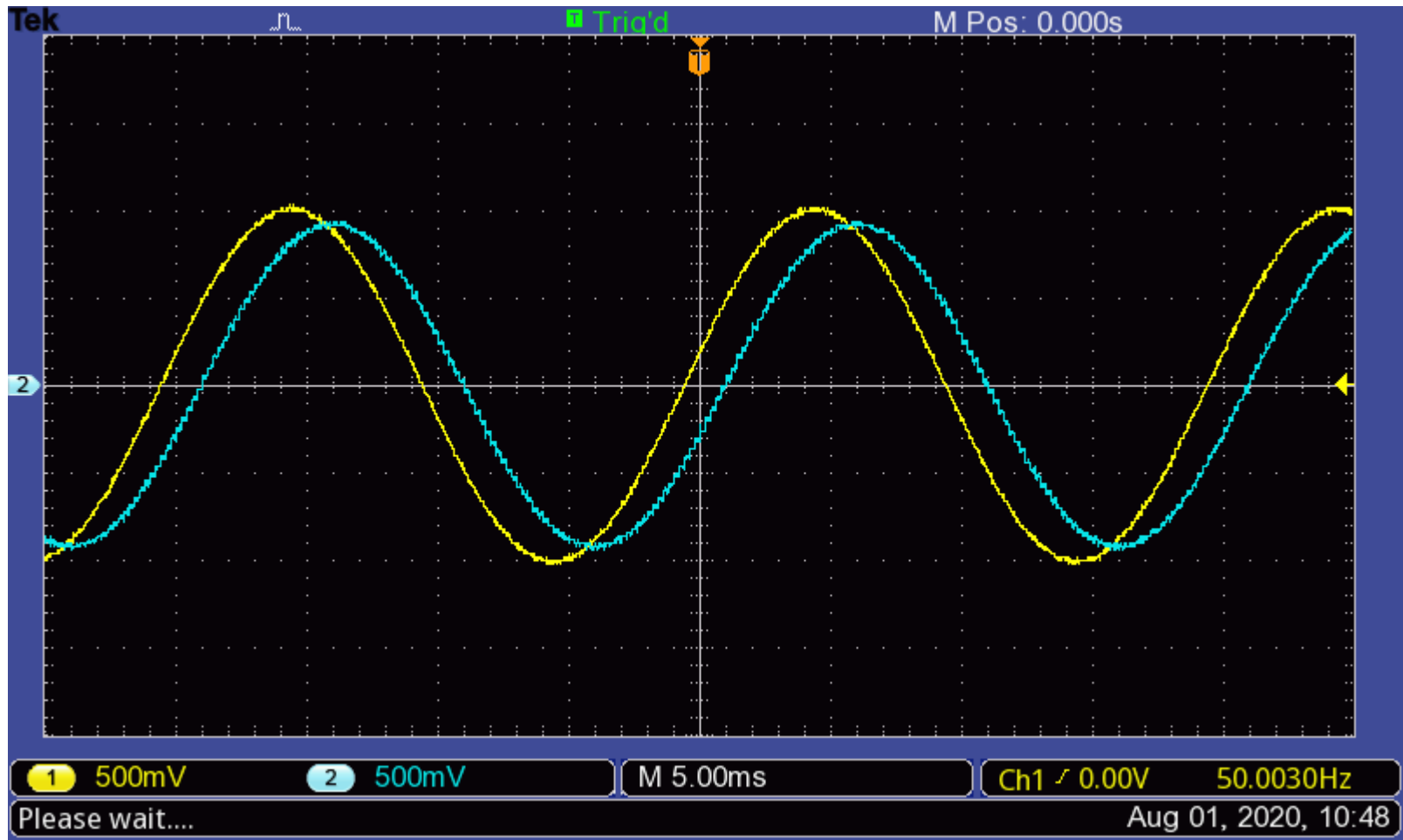
```
/* USER CODE BEGIN Callback 0 */
int da_value,ad_value;
if (htim->Instance == TIM10) {
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, 10000) == HAL_OK) {
        ad_value = HAL_ADC_GetValue(&hadc1);
    }
    u = (float)ad_value - 2048.0;
    y = (oldy + deltat*wc*u)/(1+deltat*wc);
    oldy = y;
    da_value = y + 2048.0;
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_2, DAC_ALIGN_12B_R, (uint32_t)(da_value));
}
/* USER CODE END Callback 0 */
```

# Step Response

- Function Generator: 50Hz square wave
- Time Constant:  $\tau = \frac{1}{\omega_c} = \frac{1}{2\pi \cdot 100} = 1.59 \text{ msec}$

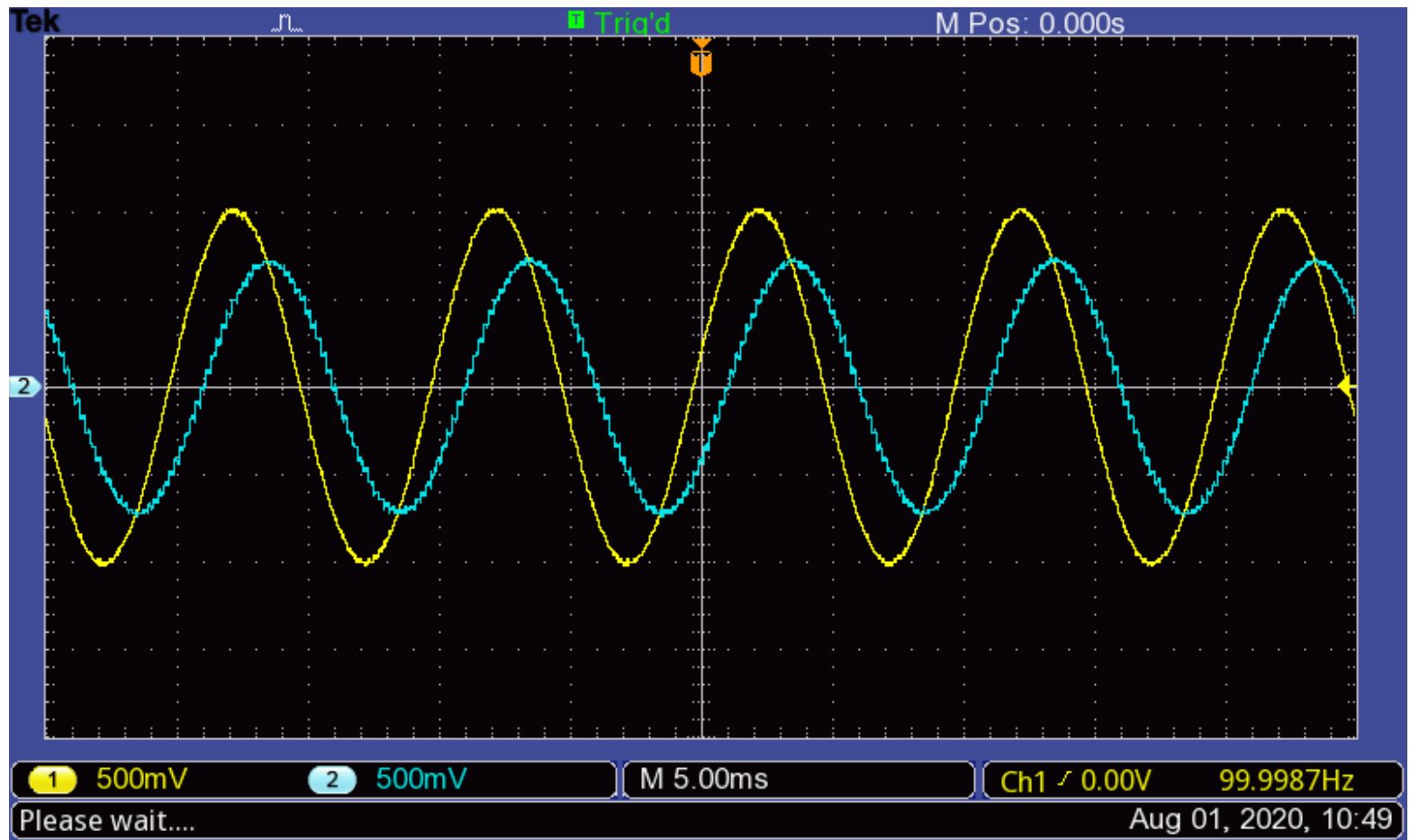


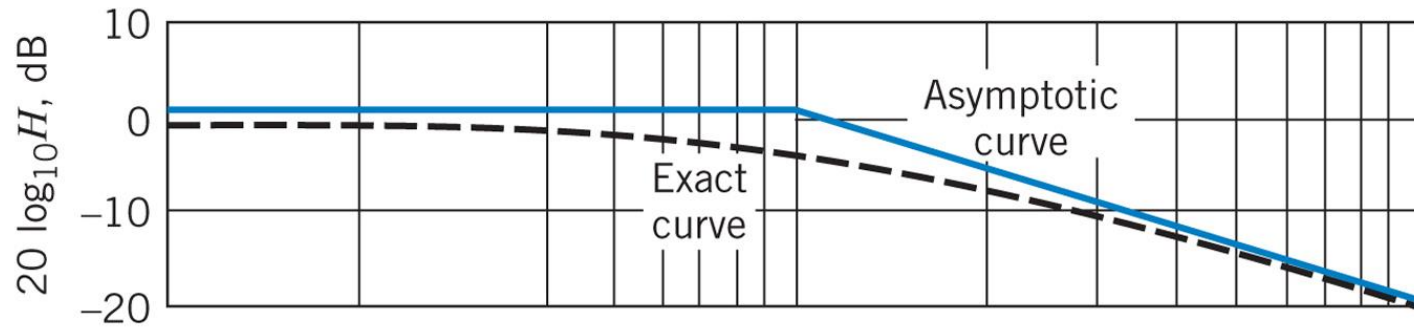
- Function Generator: 50Hz sine wave



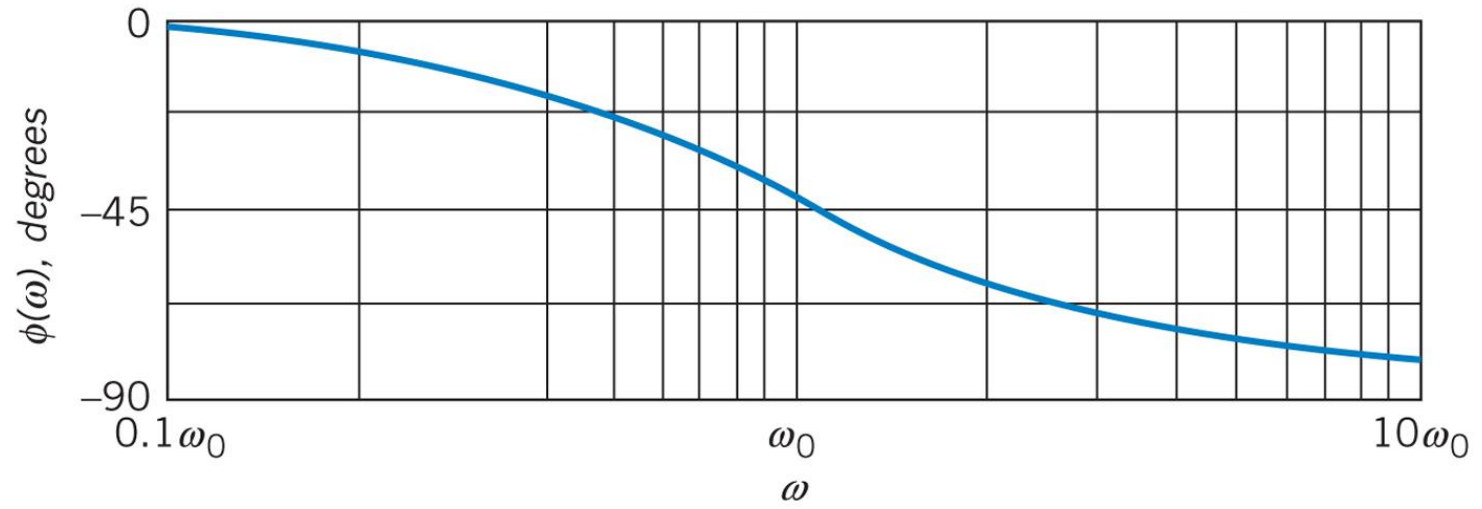
- Function Generator: 100Hz sine wave

$$20 \log \left( \frac{1}{\sqrt{2}} \right) = -3dB$$



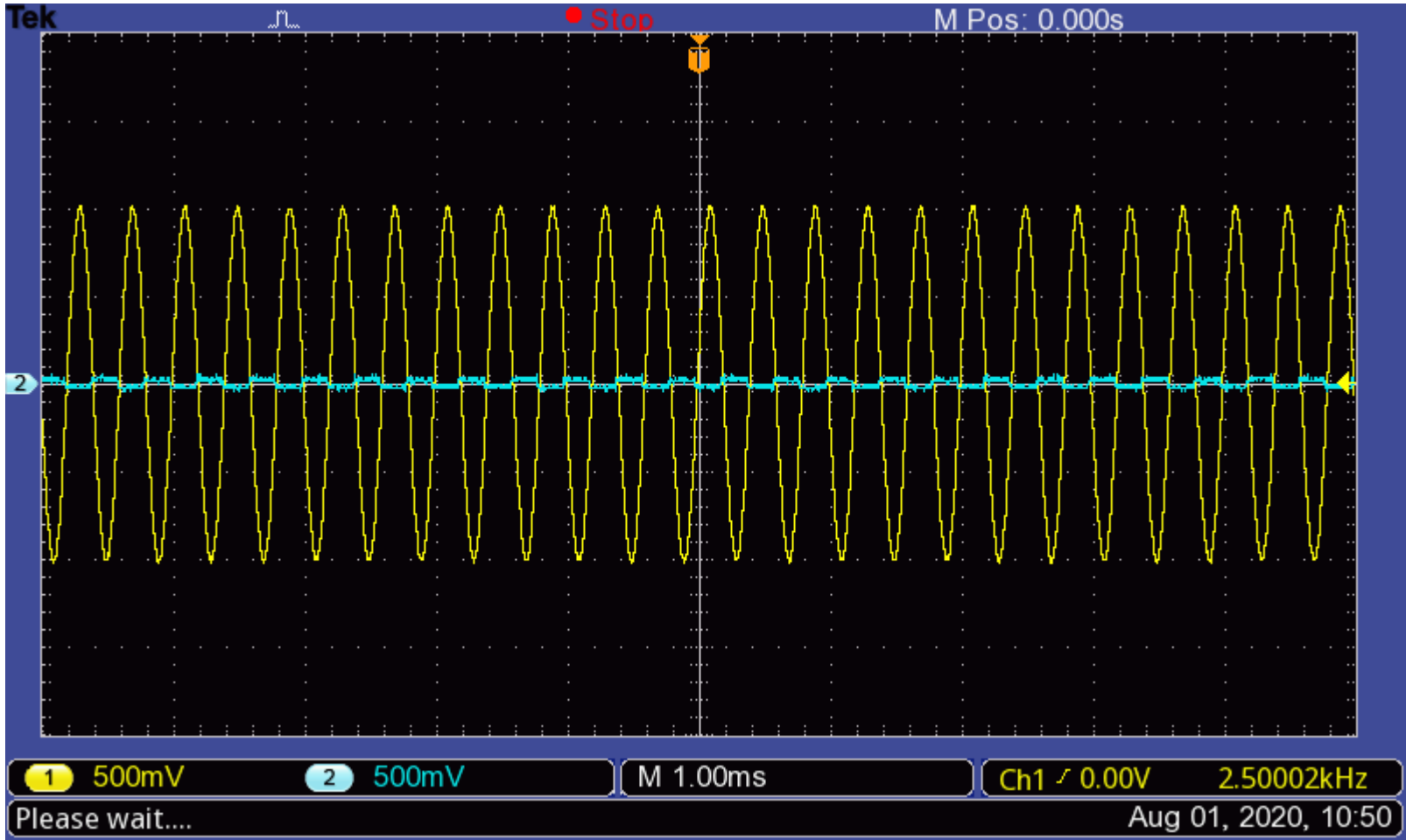


(a)

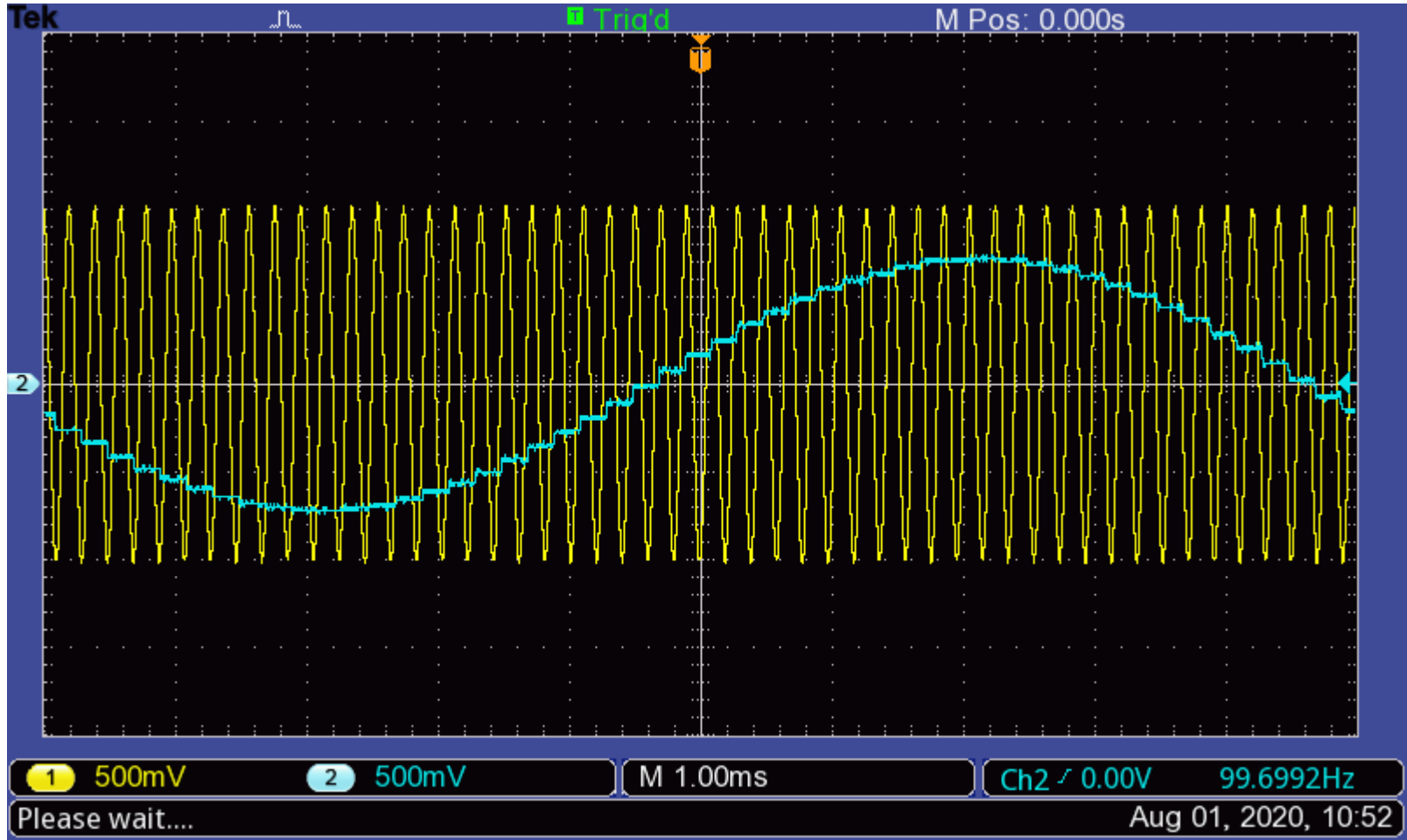


(b)

- Function Generator: 2500Hz sine wave



# Function Generator: 5100Hz sine wave



# 2nd Order Digital Filter

$$\frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$\frac{d^2 y}{dt^2} + 2\zeta\omega_n \frac{dy}{dt} + \omega_n^2 y = \omega_n^2 u$$

$$y(t) = x_1(t)$$

$$\dot{x}_1(t) = x_2(t)$$

$$\dot{x}_2(t) = -\omega_n^2 x_1(t) - 2\zeta\omega_n x_2(t) + \omega_n^2 u(t)$$

# 2nd Order Digital Filter

$$x_1(t) = \int_0^t x_2(\tau) d\tau$$

$$x_2(t) = \int_0^t \left[ -\omega_n^2 x_1(\tau) - 2\zeta\omega_n x_2(\tau) + \omega_n^2 u(\tau) \right] d\tau$$

Approximation by forward rectangular rule

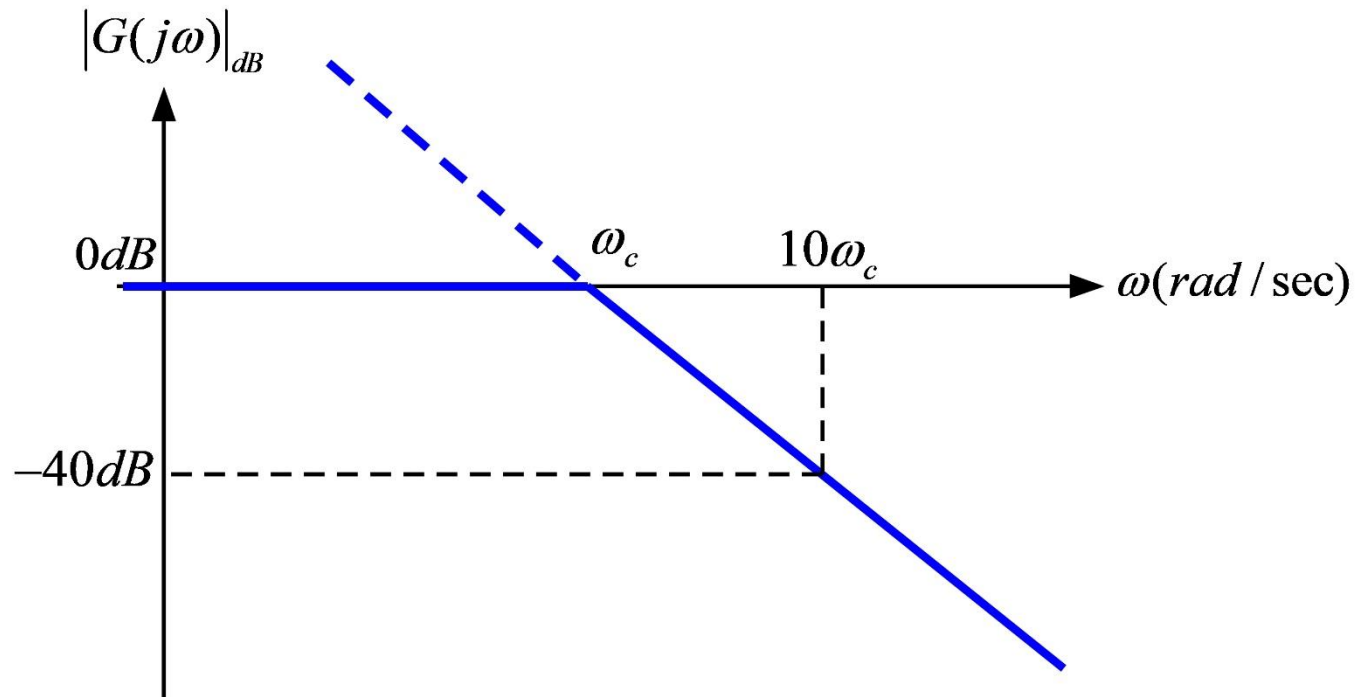
$$x_1(t) = x_1(t-T) + T \cdot x_2(t-T)$$

$$x_2(t) = x_2(t-T) + T \cdot \left[ -\omega_n^2 x_1(t-T) - 2\zeta\omega_n x_2(t-T) + \omega_n^2 u(t-T) \right]$$

# 2nd Order Digital Filter Implementation

$$\omega_n = 2\pi \times 100$$

$$\zeta = 0.2, 0.4, 0.8, 1$$



## Exercise 1: 주파수 응답 측정

- 위의 식을 이용하여 second-order digital filter를 구현 하시오. Resonant frequency는  $\omega_n = 2\pi \times 100$  로 고정하고, damping factor는

$$\zeta = 0.2, 0.4, 0.8, 1$$

의 4가지 값을 대입하여 다음의 내용을 실행 하시오.  
Sampling frequency는 10KHz로 하시오.

# 주파수 응답 측정

- 다음의 주파수 값에 대해서 입력 정현파의 크기와 출력 정현파 크기의 비 값을 오실로스코프로 측정하여 대략적인 magnitude frequency response를 구하여 그래프를 그리시오.

20, 50, 100, 200, 500, 1000, 2500,  
5000, 9500, 9800, 9900, 9980 Hz

- 주파수 응답은 MATLAB script file인 frplot.m 을 실행하여 그래프를 그릴 수 있다. 이를 위해서 텍스트 에디터(메모장)를 이용하여 다음과 같은 형식의 파일을 작성하여 파일 이름을 data로 하고, 실행할 MATLAB script와 같은 디렉토리에 넣은 후, MATLAB에서 frplot.m을 실행한다.

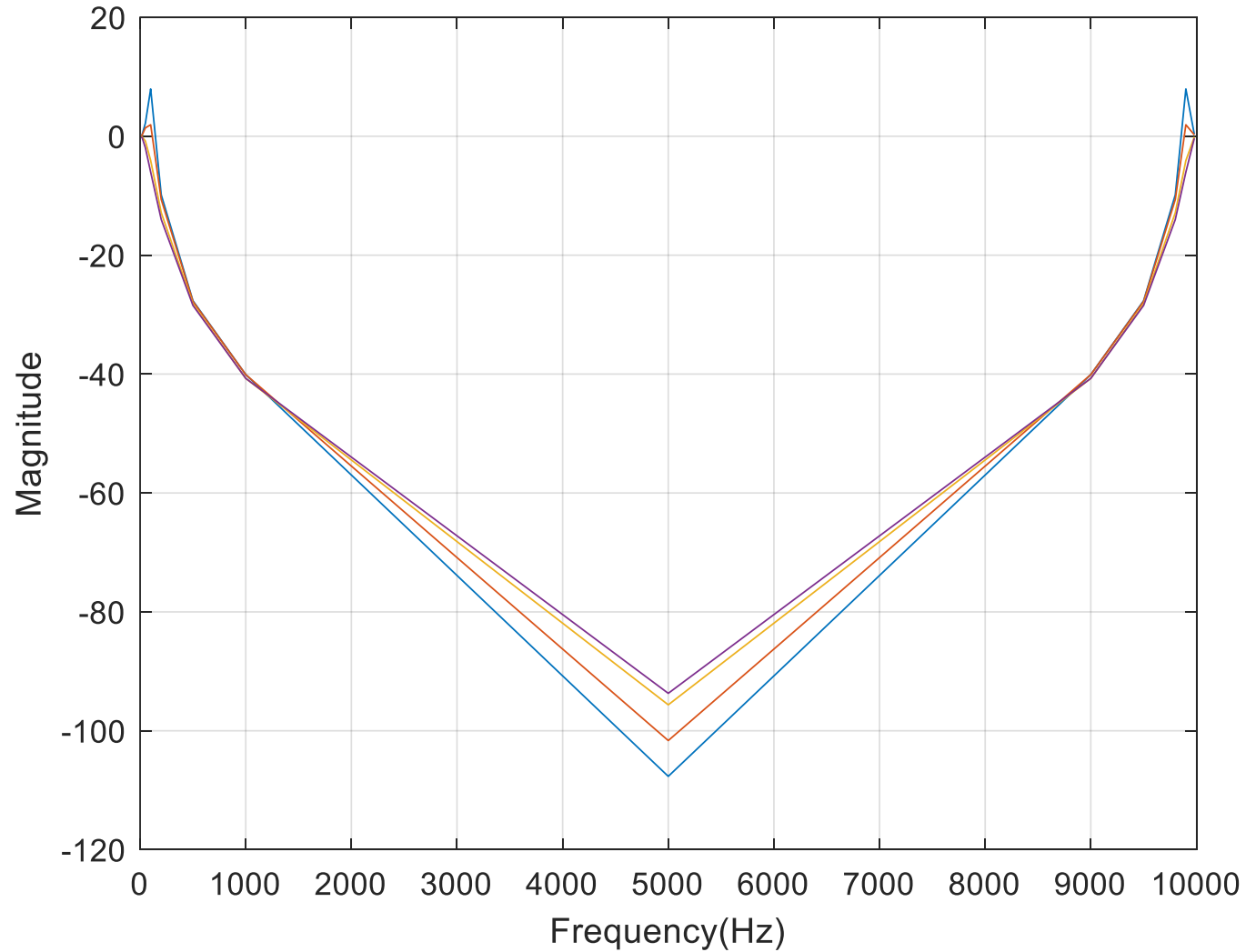
# 주파수 응답 측정

- 아래의 파일에서 첫째 줄은 주파수이며, 4개의 칼럼에는 각 damping factor에 해당하는 측정 값을 기입한다.
- 측정을 위해서 입력 정현파는 peak-to-peak가 2 Volt가 되도록 하고, 아래 파일에는 필터 출력 정현파의 peak-to-peak 측정 전압 값을 기입한다.
- 그래프는 두 개를 그리며, 한 개는 주파수를 linear scale로, 다른 하나는 주파수를 log scale로 그린다.
- 아래 형식의 파일을 작성할 때 아래의 표에 있는 것과 같은 구분 선은 필요가 없으며, 텍스트 에디터 (메모장)에서 단순히 숫자만 입력하고, 숫자와 숫자 사이에는 blank를 넣는다.

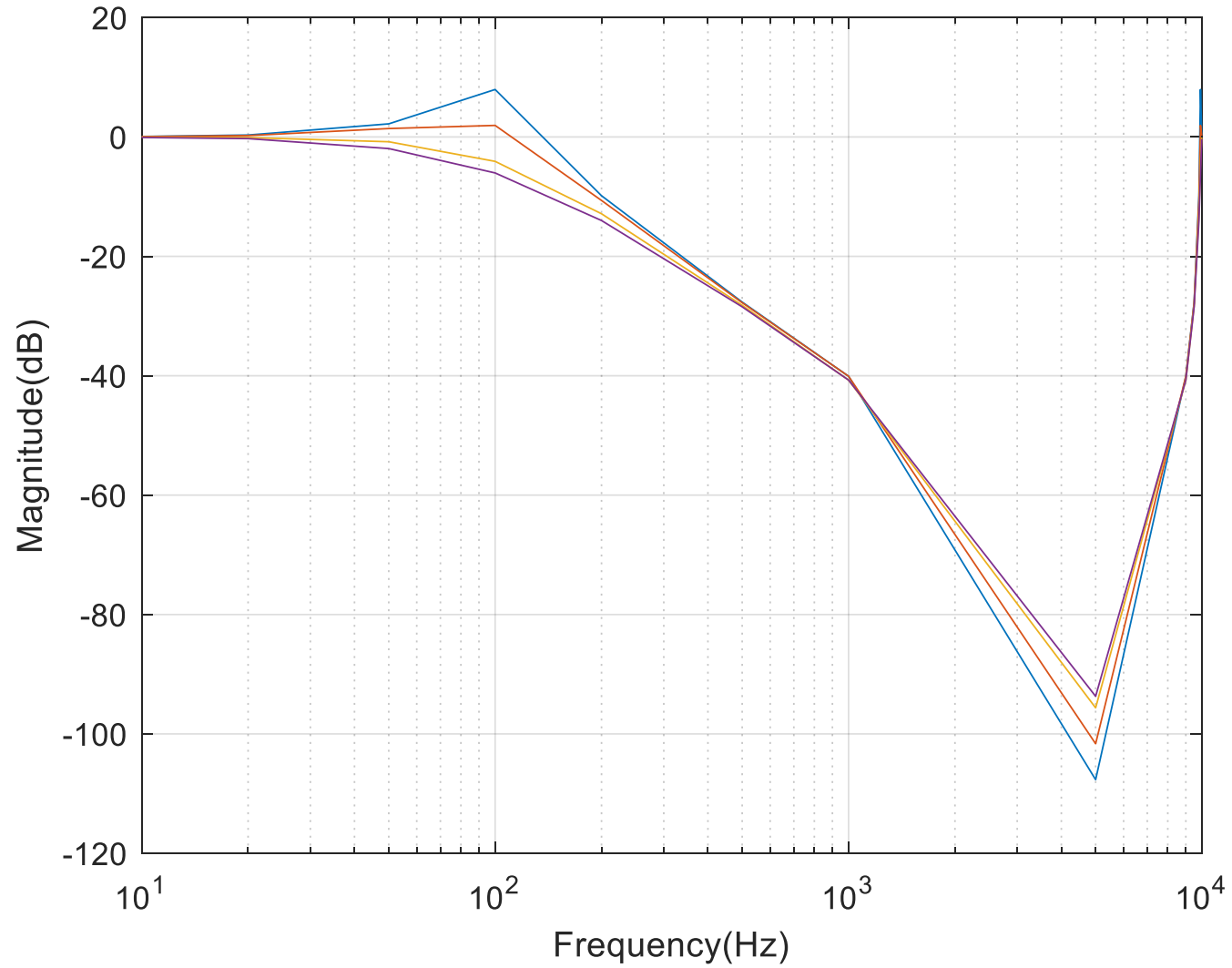
# “data” file format

20				
50				
100				
200				
500				
1000				
5000				
9000				
9500				
9800				
9900				
9980				

# Linear Scale

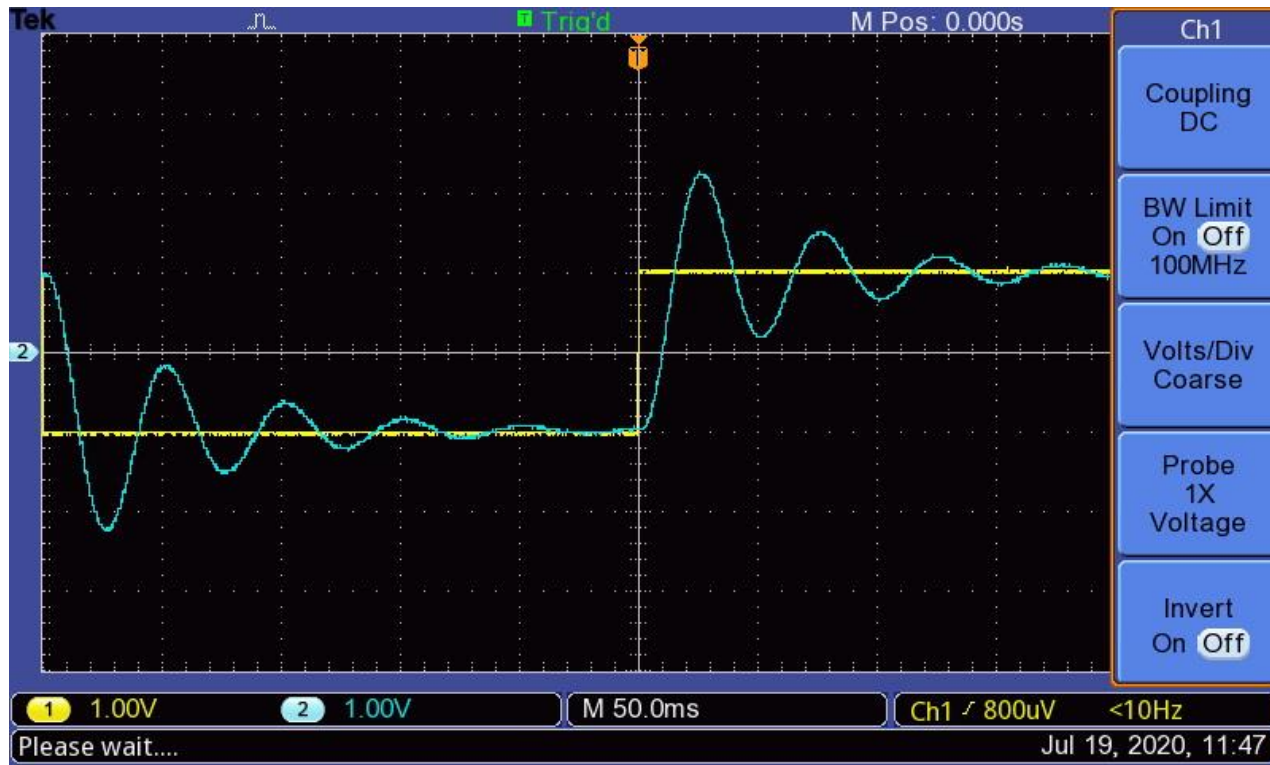


# Log Scale

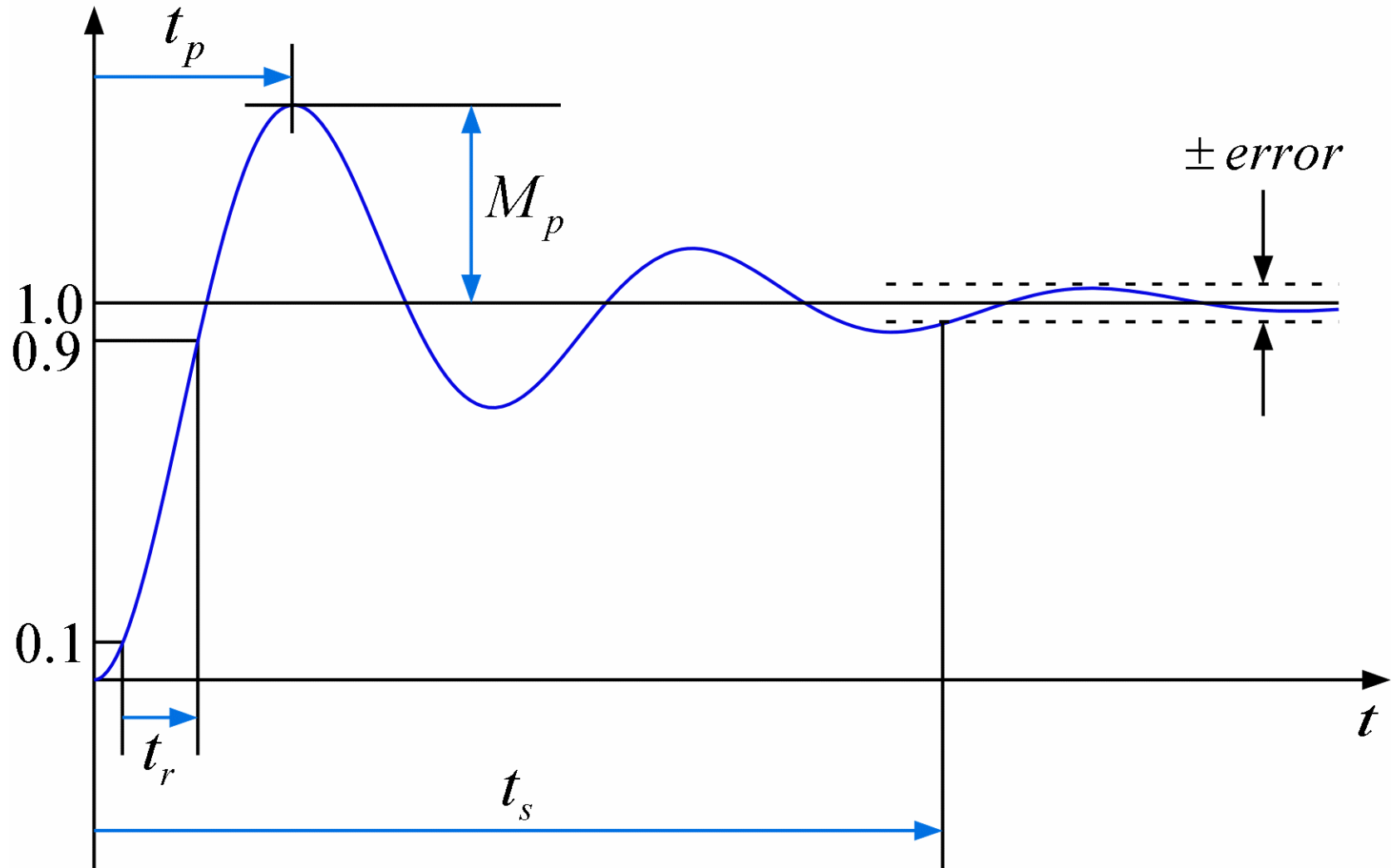


## Exercise 2: Overshoot 측정

- 함수 발생기로 주기 10 Hz 정도의 square wave를 발생 시켜서 입력 한 후, 출력 신호로부터 overshoot값을 오실로스코프로 측정한다.

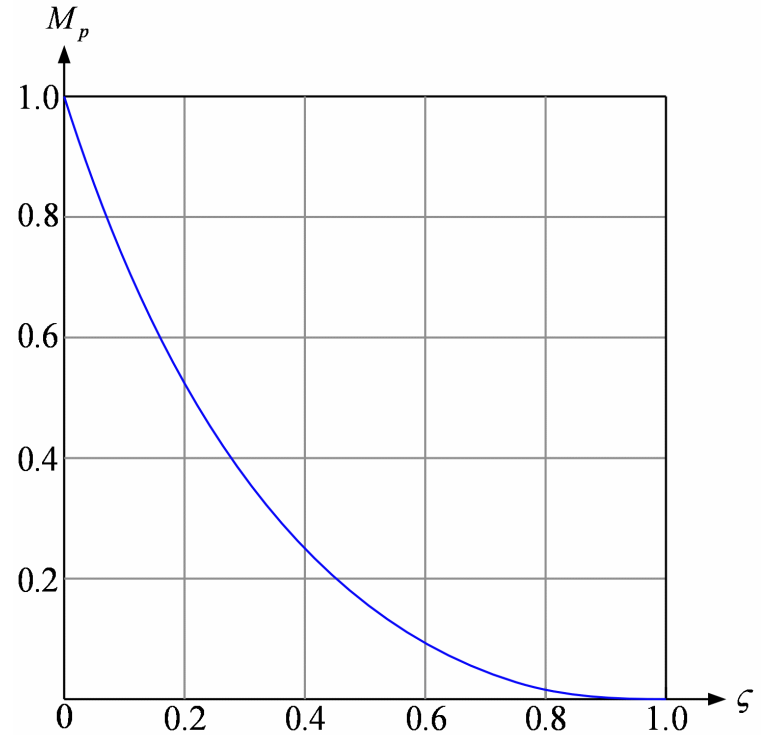


# Overshoot: $M_p$



# Overshoot (%)

$$\% M_p = e^{-\zeta\pi/\sqrt{1-\zeta^2}} \times 100$$



- 각 댐핑 값  $\zeta = 0.2, 0.4, 0.8, 1$  에 대한 overshoot의 이론 값과 측정 값을 비교한다.