CmpE244 (II)

II-1

Nov.10 (Wed)

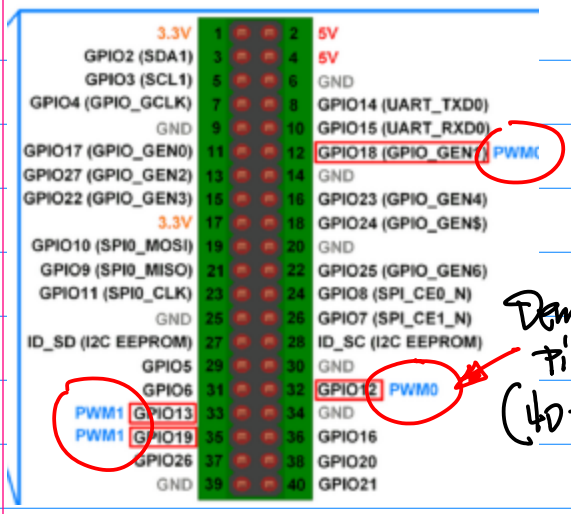
Example: PWM

Run Jetson-io.py to config the pin for PWM

jetson-io.py

First, Try to Run the jetson-io.py, wait for UI screen to Appear.

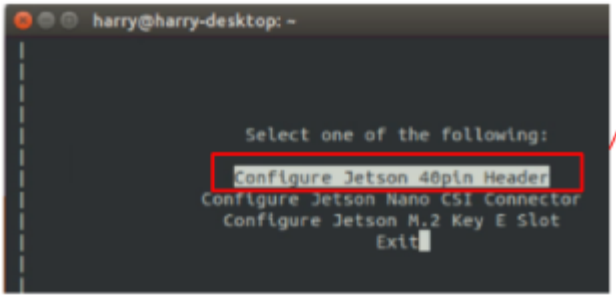
Note: If UI did not show, then do the following fix. (Step 1 in PPT).



Demo Example Pin-32 (40 pin Connector)

Step 2. Run jetson-io.py to configure pins

$sudo /opt/nvidia/jetson-io/jetson-io.py



Mapping of PWM function (Device Driver) to physical pin(s) Requires Software Configuration Tool written python.

(DT.: Device Tree)

Fix the "Bug" By 4 Steps

Note:

Note: I2C and UART pins are connected to hardware and should not be reassigned. By default, all other pins (except power) are assigned as GPIO. Pins labeled with other functions are recommended functions if using a different device tree.

Step 1. Fix bugs from the distribution

Configurati

1. $sudo find /opt/nvidia/jetson-io/ -mindepth 1 -ma
2. $sudo /opt/nvidia/jetson-io/config-by-pin.py -p 5
3. $sudo mkdir -p /boot/dtb
4. $ ls /boot/*.dtb | xargs -I{} sudo ln -s {} /boot/dtb/

Note, on the UI, Be sure to select "Save & Reboot"

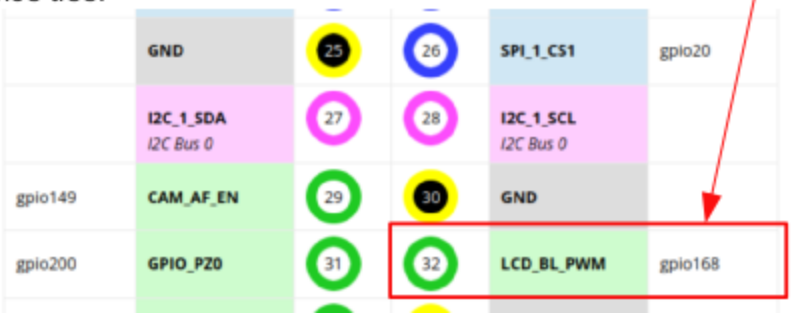
Choose pin32 as PWM pin.

Use pin 32 for PWM

Be sure to choose save and reboot to reboot the system

/boot/tegra210-

Then, use Command Line Instruction (CLI) to Access to PWM.

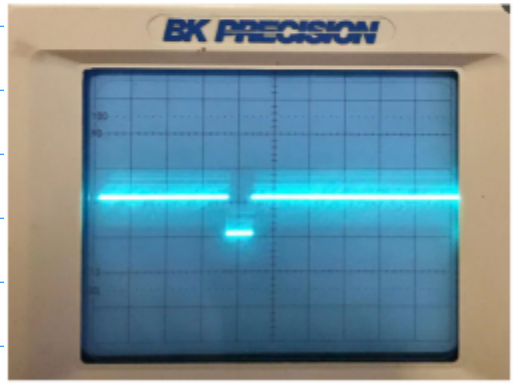


Then, enter the following instructions.
Use DSC. or Logic Analyzer to Observe the output.

```
cd /sys/class/pwm/pwmchip0
echo 0 > export
sleep 1
cd pwm0
echo 5000000 > period
echo 2500000 > duty_cycle
echo 1 > enable
```

Define as in Hz

Output high defined as in Hz



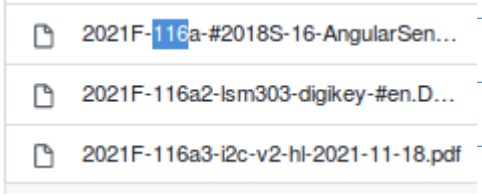
program. tied to PID (Proportional, Integral, Derivative)

2. Design Option(s) of your Choice. Please consider the timeline is tied.

Dec 1st.

Note: 1st Project Presentation is scheduled next week.

I2C Sensor Interface. Ref from the Class github



Nov 17 (Wed)

Project Requirements And Proposal Feedback.

1. Scope of the proposed work

(1) Device Driver in the kernel space will have to be a part of the implementation.

Stepper Motor Control { GPIO, PWM

Sensor Input (FeedBack) from LSM303 for example. or Equivalent I2C Sensors

(2) User Space Application

Example: I2C (I2C)

Table 9

Pin name	Pin description
SCL	I2C serial clock (SCL)
SDA	I2C serial data (SDA)

Clock

Data (Bi-Directional)

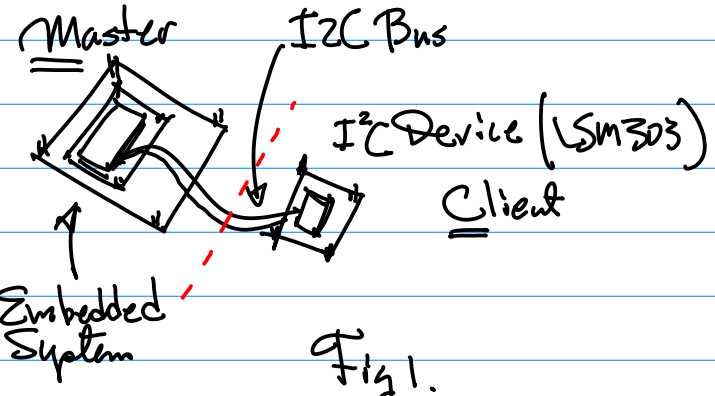


Fig 1.

CmpE244(II)

(1) START (2) Slave Addr. + "W" write (4) Sub Address (Address inside the Slave Device) (5) Data

Table 11. Transfer when master is writing one byte to slave, pp 20, datasheet

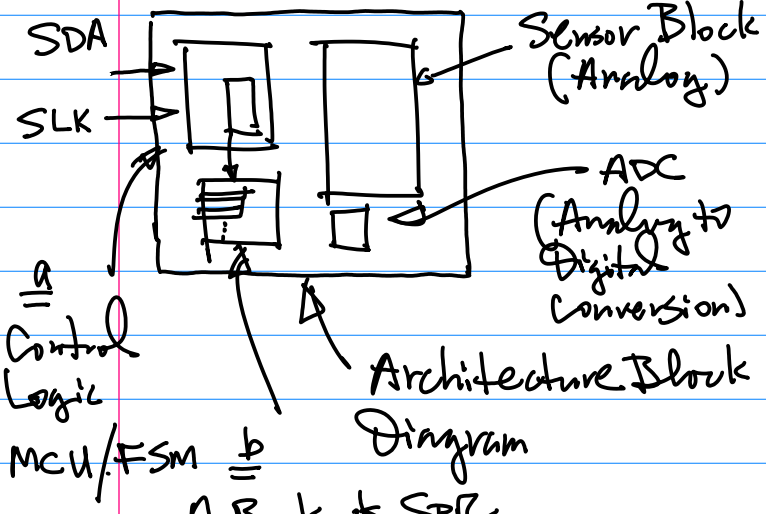
Master	ST	SAD + W		SUB		DATA		SP
Slave			SAK		SAK		SAK	

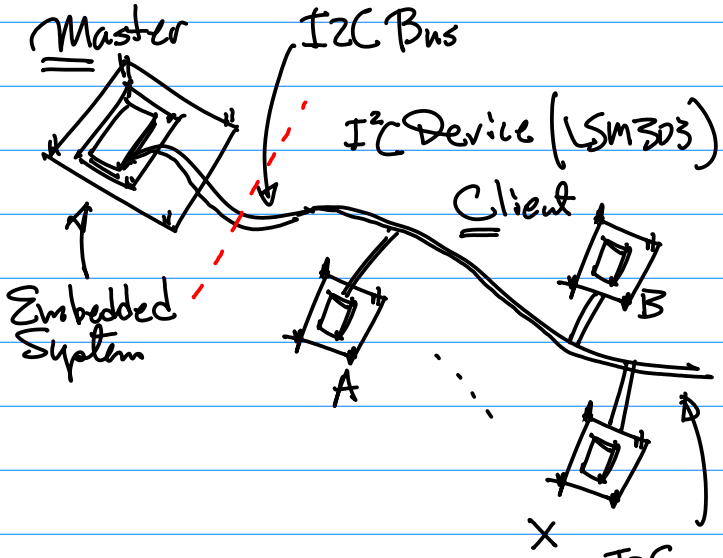
(3) Slave Acknowledgement (6) STOP

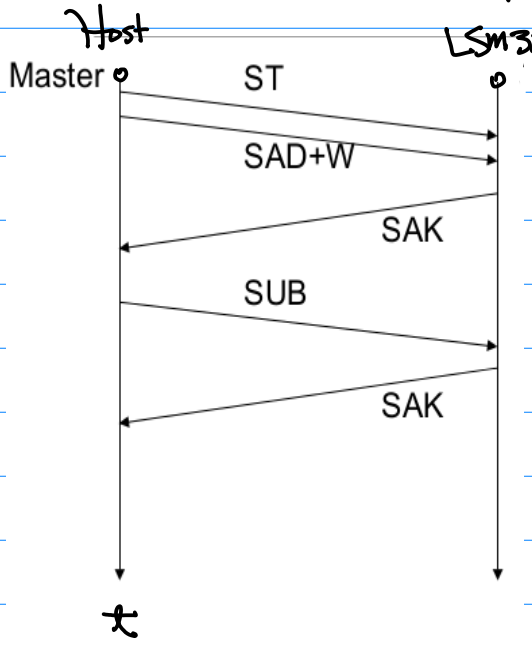
Sequence of I2C Communication.

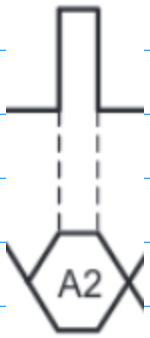
Note: memorize it when understood the "handshaking"

From Fig.1. Slave Device "I2C"









Waveform ⇒ a. Sync. Relation B/W SDA & SLK

b.

t_0 : Don't Care

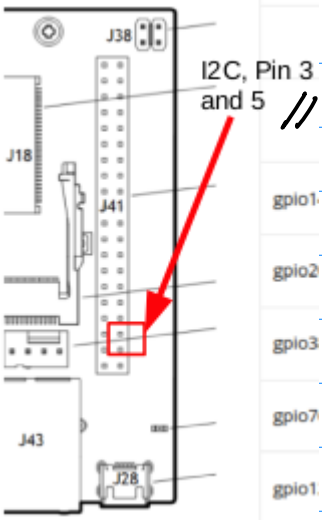
t_1 : Change State → Timing!

Ref: Datasheet for LSM303

2021F-116a2-lsm303-digikey-#en.D...

Required

Exercise: Please Complete this Interface Design with your target Board, For NAND, establish Connection By Drawing schematics



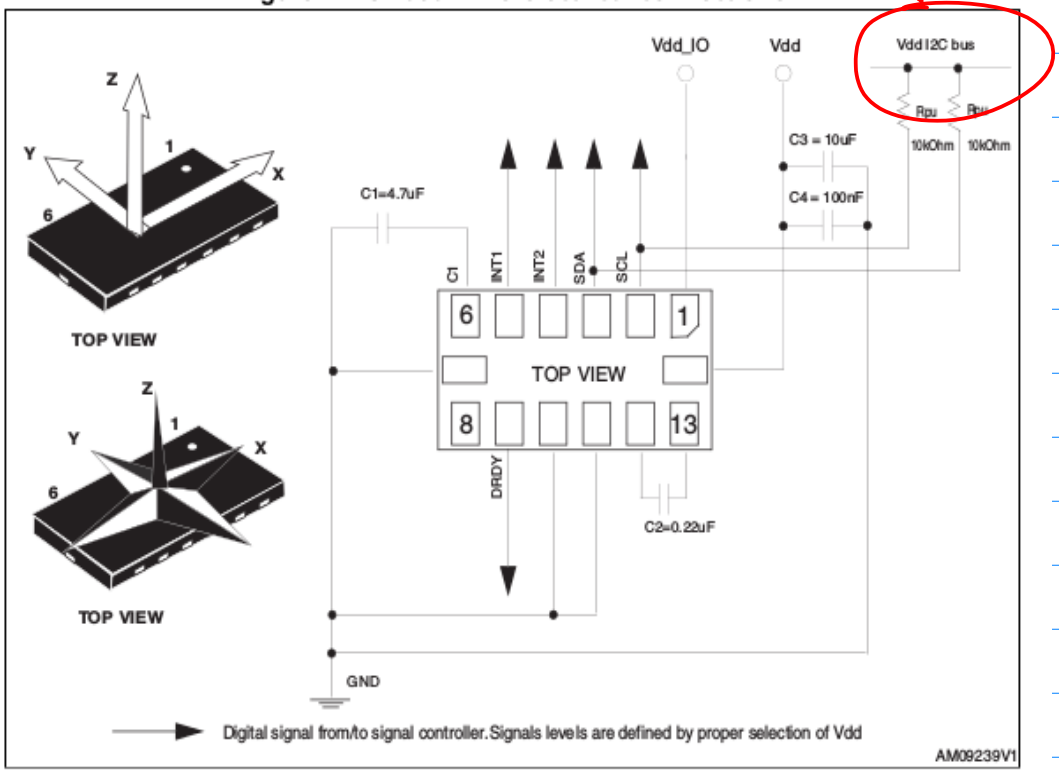
I2C, Pin 3 and 5

2021F-116a3-i2c-v2-hl-2021-11-18.pdf

To

(1) See J41 40Pin Connector.

Figure 4. LSM303DLHC electrical connections



Example: Implementation for I2C I/F to LSM303 Sensors.

Hardware Design: See Schematics On PP.4.

Software: Table 11
Realize Handshaking, e.g. Defined By Table 11.

① SAD+W for "M" sensor

0X90 (see Notes PP6)

Table 11. Transfer when master is writing one byte to slave, pp 20, datasheet

Master	ST	SAD + W		SUB ②		DATA ③		SP
Slave		0X3C	SAK	0X00	SAK	0X90	SAK	

Ref from LSM303 Datasheet

Mag. Sensor Addr. (SAD) 0X3C
Acceleration Sensor (SAD) 0X32

Table 14. SAD+Read/Write patterns

Command	SAD[7:1]	R/W	SAD+R/W
Read	0011001	1	00110011 (33h)
Write	0011001	0	00110010 (32h)

PP.21

② Config. Register for "M" sensor

SUB (Addr.) 0X00.

PP24.

First, Send 0X3C ⇒ Then send Write = 0X00

Table 17. Regis

Name	Slave address	Type
TIME_LIMIT_A	Table 14	rw
TIME_LATENCY_A	Table 14	rw
TIME_WINDOW_A	Table 14	rw
Reserved (do not modify)	Table 14	
CRA_REG_M	Table 16	rw
CRB_REG_M	Table 16	rw

7.2.1 CRA_REG_M (00h)

pp.37 (4) 8 bits, "Little Endian"

(3) (see Notes pp.5)

Table 70. CRA_REG_M register

TEMP_EN	0(1)	0(1)	DO2	DO1	DO0	0(1)	0(1)

1. This bit must be set to '0' for correct operation of the device.

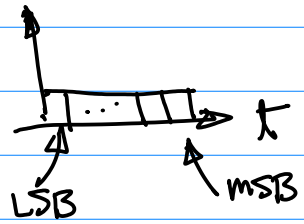
CRA_REG_M[7] = { 1 Enable "Temp" / 0 Disable "TEMP" }

CRA_REG_M[2]

(5) Define Sensor Reading Rate

Table 72. Data rate configurations

DO2	DO1	DO0	Minimum data outpu
0	0	0	0.75
0	0	1	1.5
0	1	0	3.0
0	1	1	7.5
1	0	0	15



(6) Example 0X90

1001:0000

Enable Temperature Sensor, Read Rate is 15 Hz.

Once init & Config is done, then start reading. (7)

LSM303DLHC

7.2.4 OUT_X_H_M (03), OUT_X_L_M (04h)

X-axis magnetic field data. The value is expresse

pp39

Note: the Sensor Data is in 2's Complement.

Read the Sensor Data

2021F-116a (PPT, Ref. github)

(3)

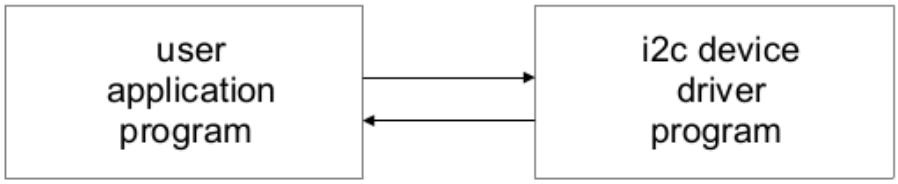
Master	ST	SAD +W		SUB		SR	SAD +R			MAK		MAK		NMAK	SP
Slave			SAK		SAK			SAK	DATA		DATA		DATA		

Repeated start

Master Acknowledgment

No Master Acknowledgment

C Code for the Init and Config 1



① Sample code from /mini6410/linux/examples/eeprog.c

```
int main(int argc, char** argv)
{
        struct eeprom e;
        int op;    op = 0;
        usage_if(argc != 2 || argv[1][0] != '-' || argv[1][2] != '\0');
        op = argv[1][1];
        fprintf(stderr, "Open /dev/i2c/0 with 8bit mode\n");
        die_if(eeprom_open("/dev/i2c/0", 0x50, EEPROM_TYPE_8BIT_ADDR, &e) < 0,
                "unable to open eeprom device file "
                "(check that the file exists and that it's readable)");
        switch(op)
        {
```

② For EEPROM

But, for NANO (NVDA) we have existing code in github (untested). Ref: github (CmpE244)

2021F-116a3-i2c-v2-hl-2021-11-18.pdf

Be careful the code for I2C Servo Not for LSM303. use the git Repo. code, Modify it for your Need.

I2C Programming

https://www.jetsonhacks.com/2019/07/22/jetson-nano-using-i2c/

Cross Reference (Not Tested): libi2c github

To use:
1. Installation
2. Interface
3. Data structure

```
i2c_ioctl_write (once max 16 bytes) are more efficient than i2c_write (once max 4 bytes).
/* Close i2c bus */
void i2c_close(int bus);
/* Open i2c bus, return i2c bus fd */
int i2c_open(const char *bus_name);
```

Too Small, But look for this from the github.

Now, Connect this to team presentation, 3 tiers in the implementation.

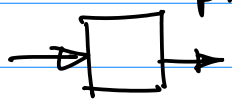
Tier1: Stepper Motor Drive (GPPT+PWM) Kernel Space + User Space.

Tier2: Sensor FeedBack Such as I2C (LSM303) K-U-Spaces

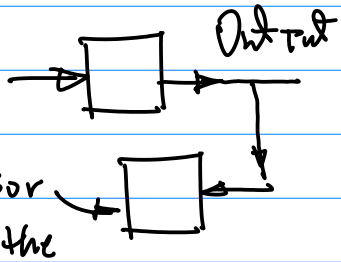
Tier3: P.I.D. Controller. (Derivative/Integral)

Example: To process Sensor Data for PID controller Design.

First, "plant" (stepper motor + motor Drive)



2nd, Add Sensor



Input to the Sensor is the Output of motor

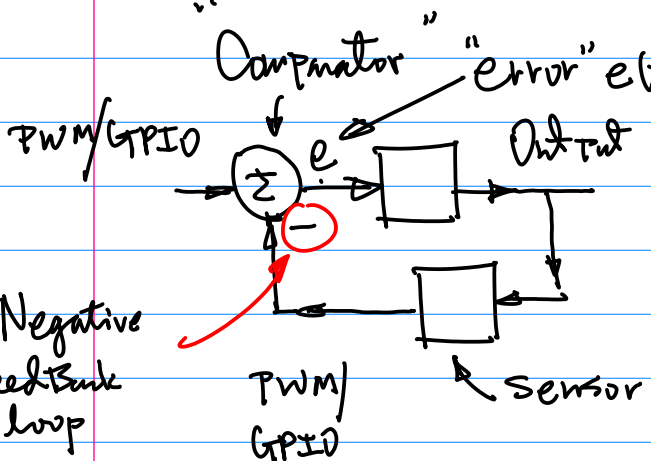
Output → Processed

Converted to Signal to drive the motor

→ PWM Signal (f_{PWM}, Duty Cycle)

↳ GPIO (Control Direction of the Spinning

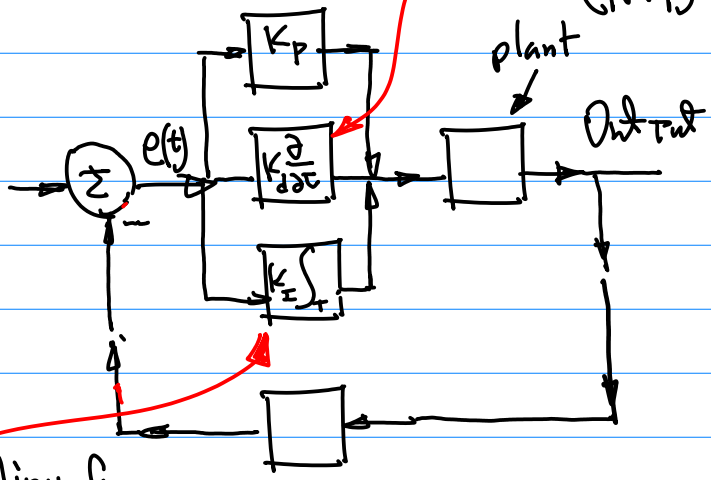
3rd



Feedback loop Control

Next

Derivate is Computed By Central Diff. (PP.9)



$K_d \frac{\partial}{\partial t} e(t)$ for Derivative ...(1)

$K_I \int e^2(\tau) d\tau$ for integral ..(2)

Example:

Sensor Data (from Output of the Sensor)

t	0	Δt	$2\Delta t$	$3\Delta t$	$4\Delta t$	$5\Delta t$
f(t)	0.25	0.3	0.42	0.15	-0.5	-0.1

↑ Sensor Data (use Sensor data as error data, In Real Application They are NOT the Same!) We use it for Simplicity to make Point for Eqn (1) & (2) Computation

Consider Derivative Computation.

$$\frac{\partial}{\partial t}e(t) \simeq \frac{d}{dt}e(t) \simeq \frac{1}{2}\left[\text{Forward Difference} + \text{Backward Difference}\right] \quad \text{(central difference)} \quad \ldots (3)$$

$$\text{Forward Difference} = \frac{d}{dt}e(t) = \lim_{\Delta t \to 0} \frac{e(t+\Delta t) - e(t)}{\Delta t} \quad \text{(forward)} \quad \simeq e(t+1) - e(t) \quad \Delta t = 1 \quad \ldots (4)$$

$$\text{Backward Difference} = \frac{d}{dt}e(t) = \lim_{\Delta t \to 0} \frac{e(t) - e(t-\Delta t)}{\Delta t} \simeq e(t) - e(t-1) \quad \ldots (5)$$

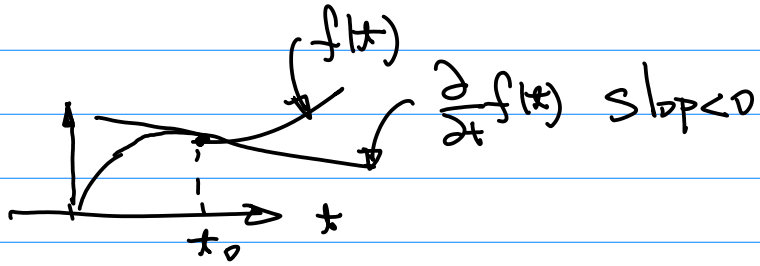
$$\frac{1}{2}\left[(4)+(5)\right] = \frac{1}{2}\left[e(t+1) - e(t) + e(t) - e(t-1)\right] = \frac{1}{2}\left[e(t+1) - e(t-1)\right] \quad \ldots (5)$$

t	0	Δt	$2\Delta t$	$3\Delta t$	$4\Delta t$	$5\Delta t$
$f(t)$	0.25	0.3	0.42	0.15	−0.5	−0.4

$$\frac{df}{dt} \quad \frac{1}{2}\left[e(2) - e(0)\right] = \frac{1}{2}(0.42 - 0.25) = \frac{1}{2} \times 0.17$$

$$\frac{1}{2}\left[e(3) - e(1)\right] = \frac{1}{2}(0.15 - 0.3) = \frac{1}{2} \times (-0.15)$$

for $t=0$, $C.D. = \frac{1}{2}\left[e(1) - e(-1)\right]$



Continue this process to find derivatives.

Consider $\int_{T} e^{2}(\tau)\, d\tau$ Integration of Error. ... (6)

Square!

How much history to count.

Make 2 steps.

$$\int_T e^2(\tau)d\tau \simeq \sum_{i=0}^{N-1} e^2(k-i) \quad \cdots (6\text{-}6)$$

t	0	Δt	$2\Delta t$	$3\Delta t$	$4\Delta t$	$5\Delta t$
f(t)	0.25	0.3	0.42	0.15	-0.5	-0.4

for $t = \Delta t (=1)$

$$\int_T e^2(\tau)d\tau \simeq \sum_{i=0}^{1} e^2(k-i) = e^2(k)+e^2(k-1)\Big|_{k=1}$$

$$= e^2(1)+e^2(0) = 0.3^2+0.25^2 = \cdots$$

for $t = 2\Delta t (=2)$

$$\int_T e^2(\tau)d\tau \simeq \sum_{i=0}^{1} e^2(k-i) = e^2(k)+e^2(k-1)\Big|_{k=2}$$

$$= e^2(2)+e^2(1) = 0.42^2+0.3^2 = \cdots$$

Exercise (Homework) Write A program in User Space to Compute (1) Derivative of error Based on Central Difference; (2) Integral of error Based on Eqn (6-6).