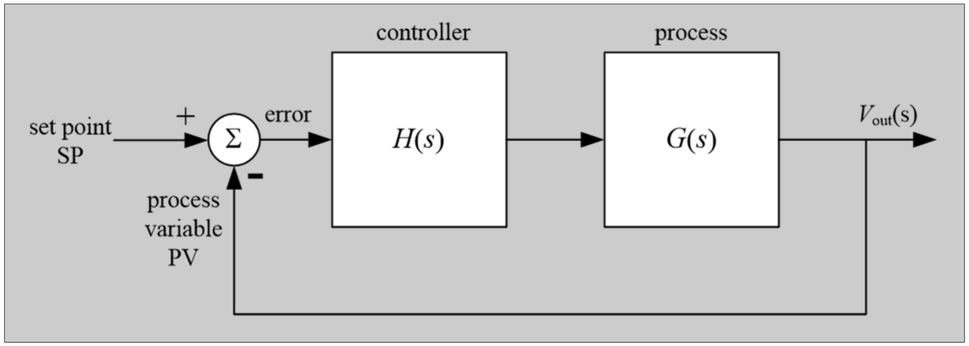
Motor with Proportional-Integral Controller

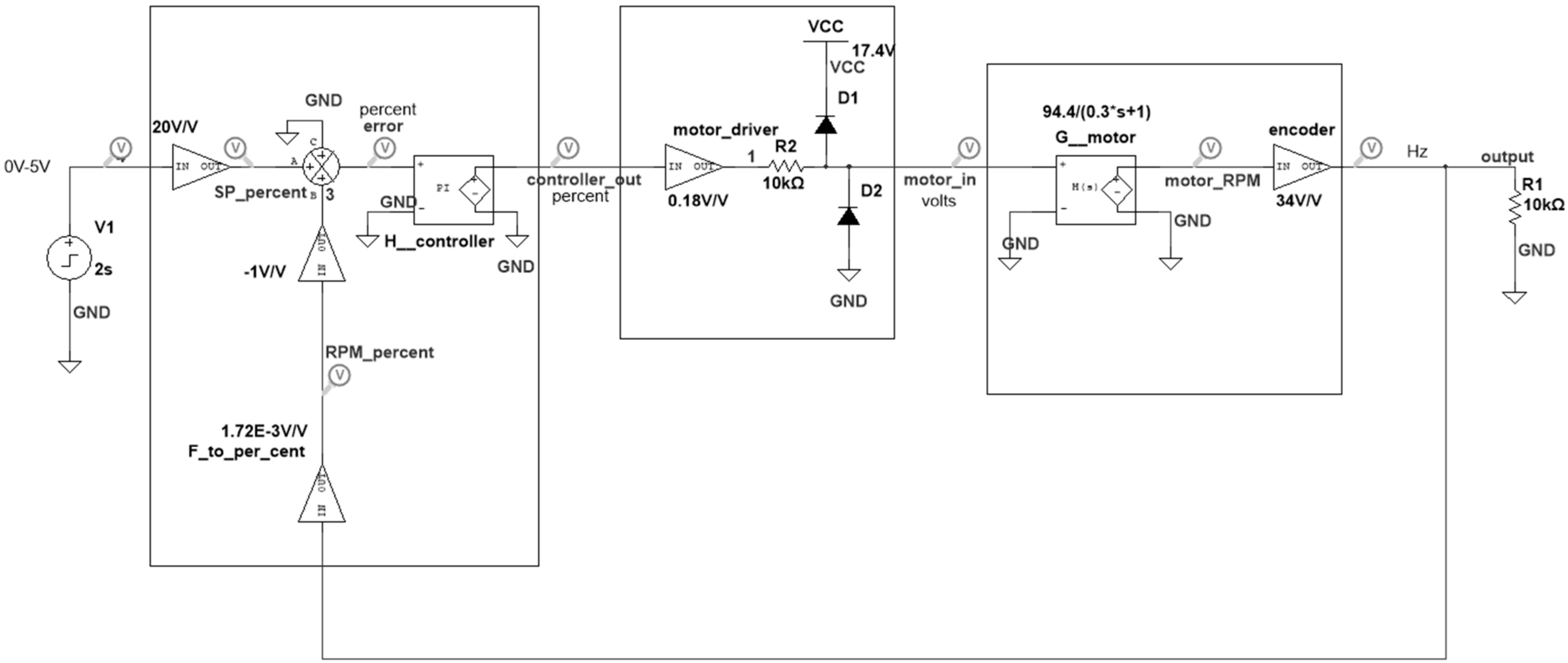


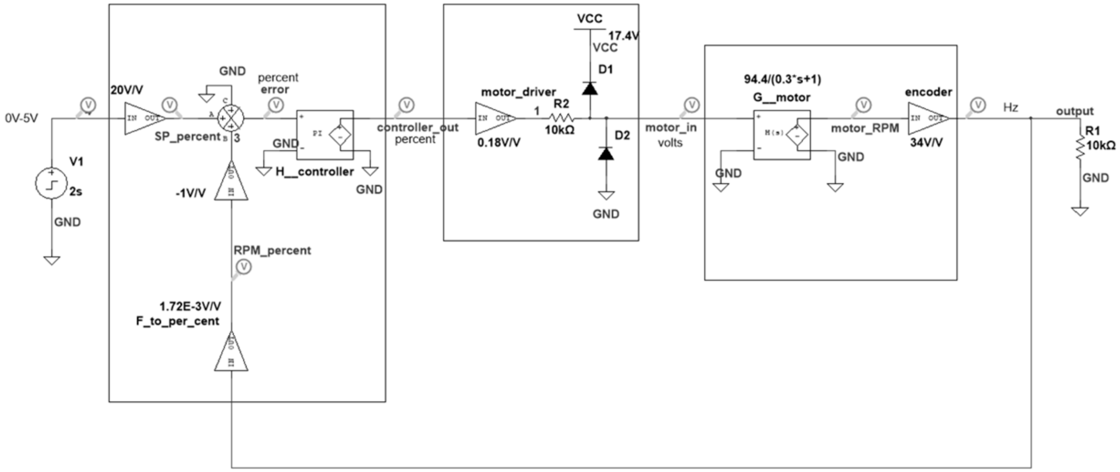
$$\frac{V_{out}}{SP} = \frac{GH}{1 + GH}$$

Integral output increases until error = 0

$$G = \frac{m_{\text{motor}}}{\tau_{\text{motor}} s + 1}$$

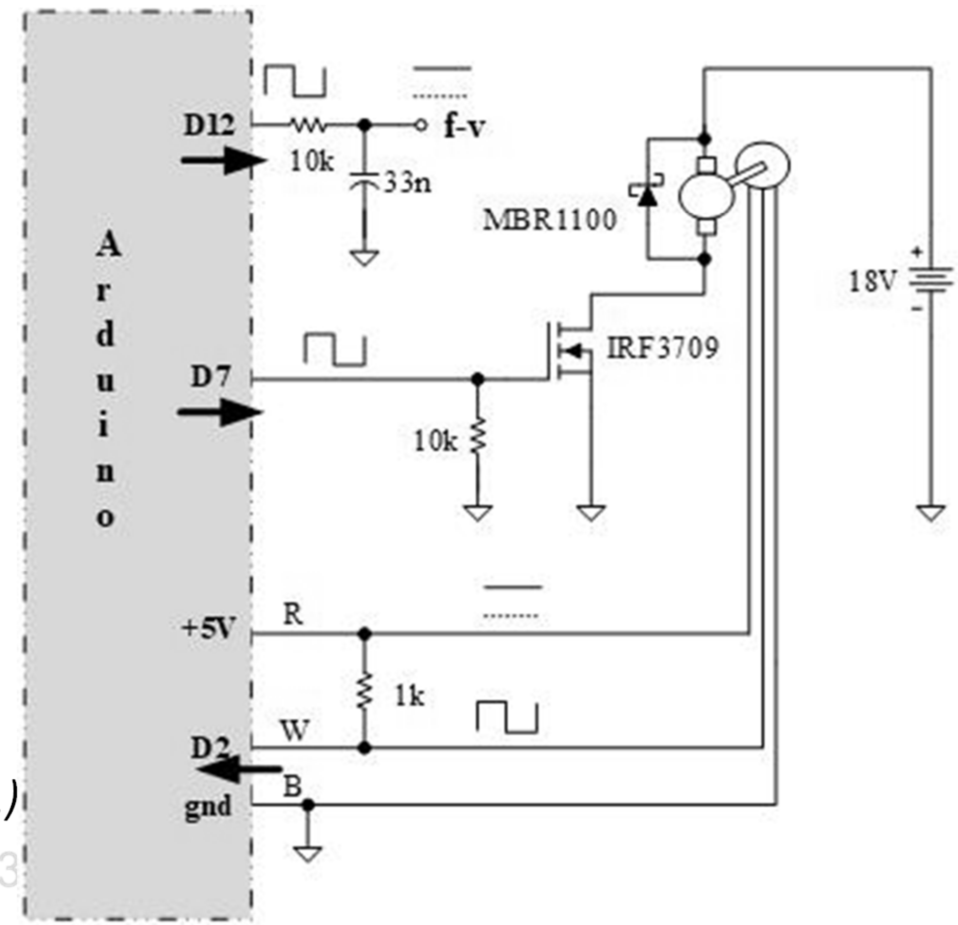
$$H = k_p + \frac{k_i}{s}$$



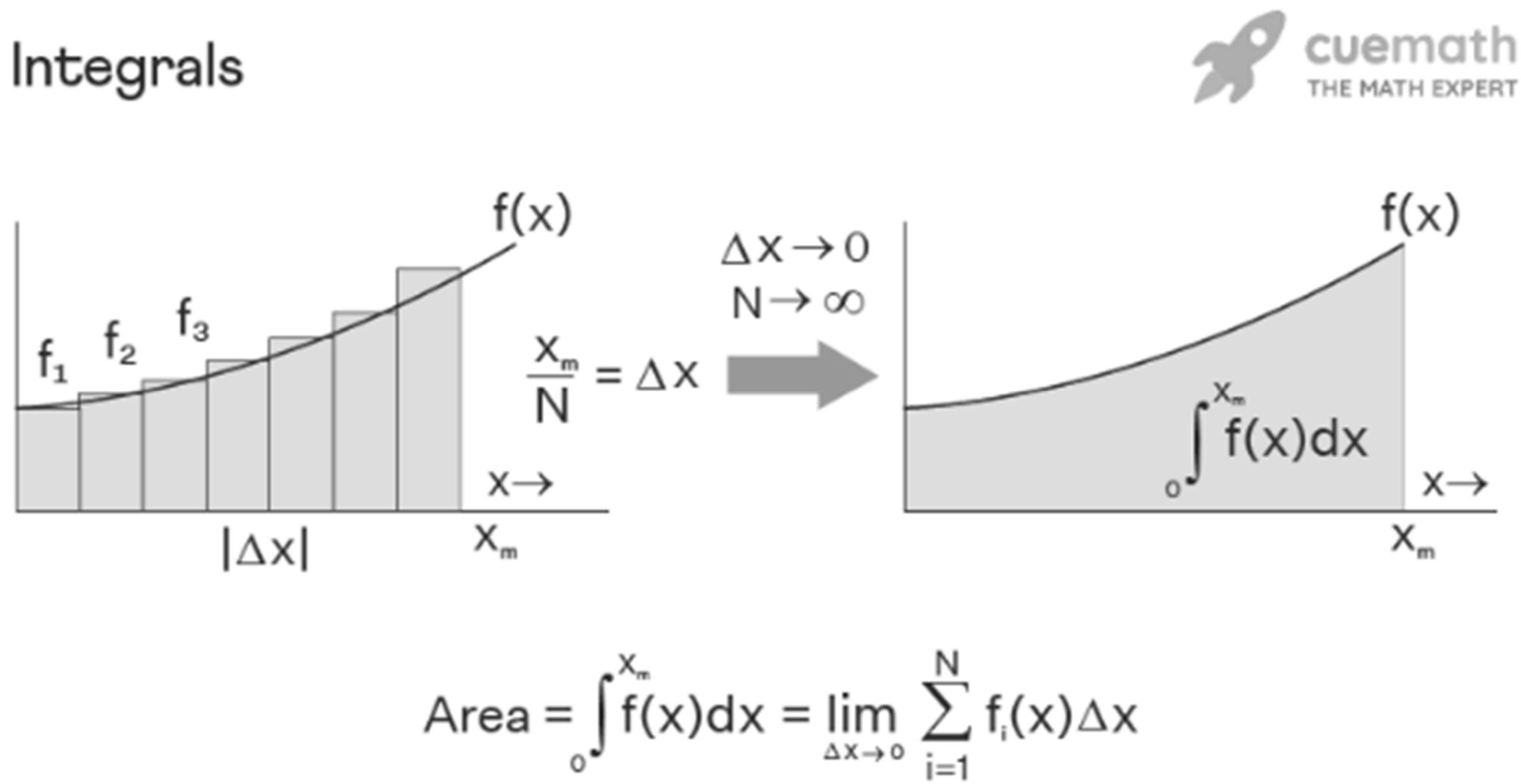


- Process under control
- Sensor
- PV
- SP
- Controller
- Power interface
- Controller Output

10/(10s+1)

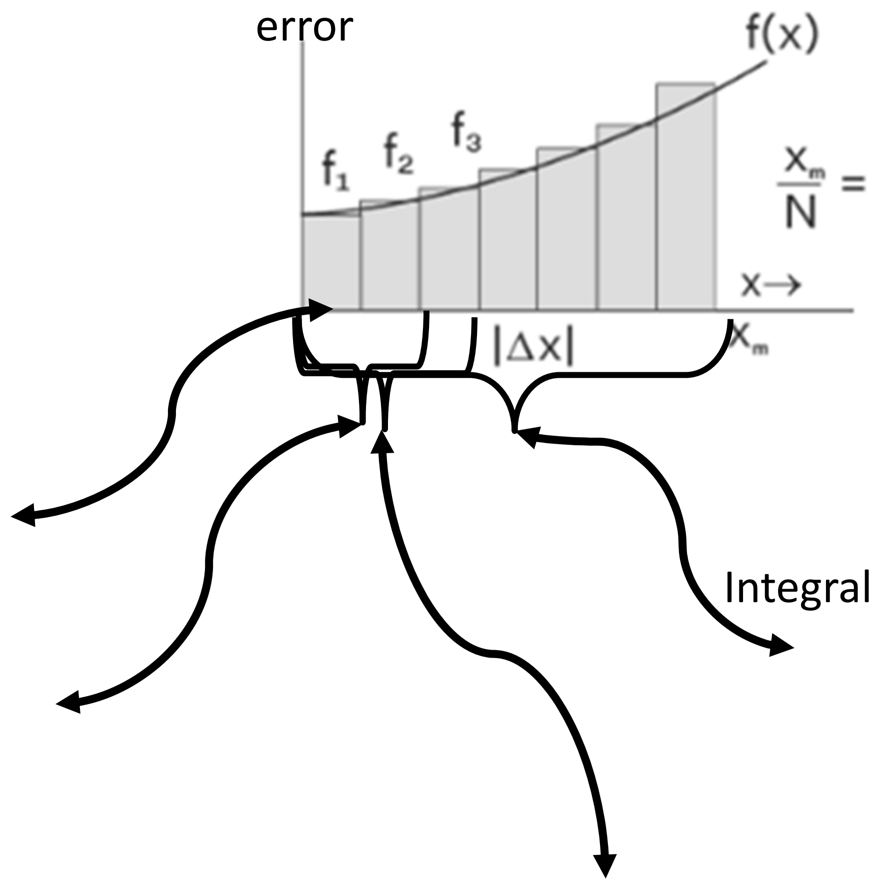


INTEGRAL: Area under the curve



https://www.cuemath.com/calculus/integral/

Discrete Integral$_m$ = $\int_0^t error \ \ dt$ => $\sum_0^{t_m} error_m \times T$



Integral$_m$=error$_1$*T+error$_2$*T+...+error$_m$*T
=Integral$_{m-1}$+error$_m$*T

Program Elements

- /**

  define variables with initial values

  integers

  floating (decimal values)

/***

setup hardware

digital out (CO) and digital in (analog SP from pot)

interrupt to time 10 ms - freq of encoder – T for integral

printer (9600)

timer for 50 kHz f-V => analog value of PV for scope

```
/**********************************************************
        loop – this is where the work is done
                wait 10 ms as counting encoder freq pulses      (0-580)

                calculate (scale)
                        RPM     (580=>1700)
                        fV      (580=>100%)
                read STP from analog potentiometer  (1024=>100%)  (SP reserved)
                PV                      (1700=>100%)
                error                   (STP-PV)
                integral                (integral+error*0.01)
                CO                      (kp*error+ki*integral)
                        0<CO<100

                scale CO (100%=>255) and drive pwmout  (to MOSFET)

                report to screen
        loop back

/**************************************************************
        support procedures
                count encoder pulses
                drive f-V output
                write to the seven-segment display
```

Set up

```
*******************************************************************/

//// set this value to desired high frequency (50kHz) pwm of fV frequency
#define Frequency 50000  /// must be defined as:50000

int encoder=0;        //encoder period
int RPM=0;
int count;           //used in the 50 kHz f-V routine
int fV = 50;         //sets the pw of the 50 kHz for f-v
int Duty_cycle;      //desired dutycycle to motor from A0 potread
int Perdisplay;      // 0 to 99  %  value to 7 segment displays

int PV=0;
int STP=0;
float error=0;
float integral=0;
float kp=1;
float ki=0;
float CO=0;
int pwmout=0;

const int LBL = 44; //left !blank
const int LLT = 43; //left !lamp test
const int LLE = 45; //left latch enable
const int RBL = 41; //right !blank
const int RLT = 40; //right !lamp test
const int RLE = 42; //right latch enable
```

10/(10s+1)


```
void setup()
{
 pinMode(7,OUTPUT);   //pwm to drive motor (1kHz)
 pinMode(2,INPUT);    //Encoder Channel A  500 per rev

 pinMode(LBL,OUTPUT);
 pinMode(LLT,OUTPUT);
 pinMode(LLE,OUTPUT);
 pinMode(RBL,OUTPUT);
 pinMode(RLT,OUTPUT);
 pinMode(RLE,OUTPUT);

 attachInterrupt(0,freq,RISING);   //interrupt process every rise on enc
 Serial.begin(9600);


timer_setup();
 DDRC = 0xFF; //pins 30 - 33 is MSB 7seg pins 34-37 is LSB 7seg
 digitalWrite(RLT, HIGH);  //setting !BL and !LT high for both ICs
 digitalWrite(RBL,HIGH);
 digitalWrite(LLT,HIGH);
 digitalWrite(LBL,HIGH);
 digitalWrite(LLE,LOW);  //setting LE LOW for both
 digitalWrite(RLE,LOW);
 PORTC = 0xFF; //displays a blank on 7seg
}
```

```
void loop()
{
//get and report the motor's speed
  delay(10);          //wait for 10 msec while routine below counts pulses from the encoder
  RPM=encoder*2.93;     //scale 570 pulses/10msec to RPM  (2048 ppr encoder)
  fV=encoder*0.175;     //1700RPM=>570pulse/10msec=>100%  to set 50kHz pulse width to speed
  encoder=0;
  OCR1B = (uint32_t)count * fV / 100; // Use this line to set 50kHz pulse width to speed

//read pot, report to 7seg display & serial monitor, drive motor

  STP=(analogRead(0)/10.24);
  display7seg(STP);
  PV=RPM*0.0588;           //scale 1700 RPM => 100%
  error=STP-PV;
  integral=integral+0.01*error;

  CO=kp*error+ki*integral;
  if (CO>100)
    CO=99;
  if (CO<0)
     CO=1;

  pwmout=CO*2.55;
  analogWrite(7,pwmout);

  Serial.print(STP);
  Serial.print("\t");
  Serial.print(PV);
  Serial.print("\t");
  Serial.println(CO);
}
```

Motor Characteristics

18 V = 100% = 1700 RPM = 28.3 rev/sec

$$\frac{1700\text{ rev}}{\text{min}} \times \frac{1\text{ min}}{60\text{ sec}} = 28.3\ \frac{\text{rev}}{\text{sec}}$$

Encoder

$$\frac{2048\text{ pulse}}{\text{rev}}$$

Motor + Encoder

$$\frac{2048\text{ pulses}}{\text{rev}} \times \frac{28.3\text{ rev}}{\text{sec}} = \frac{58\text{ k pulses}}{\text{sec}} = 58\text{ kHz}$$

$$\frac{58\text{ k pulses}}{\text{sec}} \times \frac{0.01\text{ sec}}{\textit{counting period}} = 580\text{ pulses max for one counting period}$$

Code Conversions

RPM = encoder × a

$$a = \frac{RPM}{encoder} = \frac{1700\text{ RPM}}{580\text{ pulses}} = 2.93$$

fV = encoder × b

$$b = \frac{fV}{encoder} = \frac{100\%}{580\text{ pulses}} = 0.172$$

PV = RPM × c

$$c = \frac{PV}{RPM} = \frac{100\%}{1700\text{ RPM}} = 0.0588$$

pwmout = CO × d

$$d = \frac{pwmout}{CO} = \frac{255}{100\%} = 2.55$$

Support Procedures

```
// increment every encoder CH A rise to measure motor speed
void freq()
{
  encoder=encoder+1;
}


//set 50kHz pulse width as indication of motor speed
void timer_setup(void)
{
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _BV(WGM10) | _BV(WGM11);
  TCCR1B = _BV(CS11) | _BV(WGM12) | _BV(WGM13);
  count = (16000000UL / (Frequency)) - 1;
  TCCR1B = _BV(CS10) | _BV(WGM12) | _BV(WGM13);
  OCR1A = count;
}


//write % to the 7 segment display
void display7seg(uint8_t data)//load BCD and toggle !BL    Places % on 7 seg displays
{
  uint8_t rightnum = 0;
  uint8_t leftnum = 0;

  if(data > 9)
  {
    leftnum = data/10;
    rightnum = data%10;
  }
  else
  {
    leftnum = 0;
    rightnum = data;
  }
  leftnum = leftnum<<4; //sliding leftnum to most sig 4 bits
  PORTC &=0x00;
  PORTC |= leftnum|rightnum; //outputs to PORTC (7 Segs)

  digitalWrite(RLE, LOW);
  digitalWrite(LLE,LOW);  //toggle control pins to try to load value to 7seg
}
```