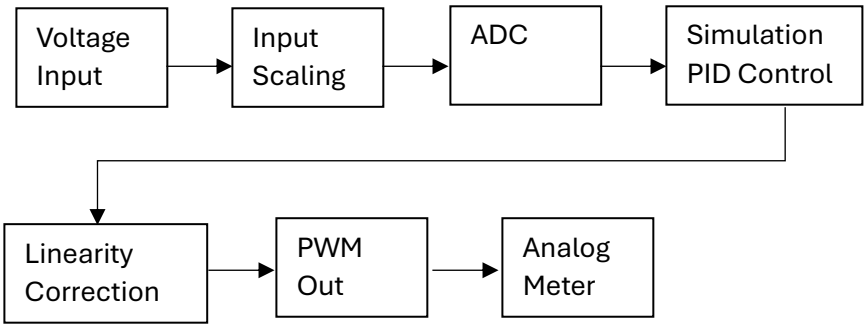
Improving Analog Meter Performance

By Simple-Circuit 2025

Improving the accuracy and dynamic response of an inexpensive -50uA 0 +50uA Galvanometer will be explored. The ammeter will be scaled with a series resistor to read -10 to +10 volts. The meter has an initial accuracy of around 2.5% of full scale. For our 20V range, the reading could be as much as 0.5V in error. We lower the typical error to around 0.1V.

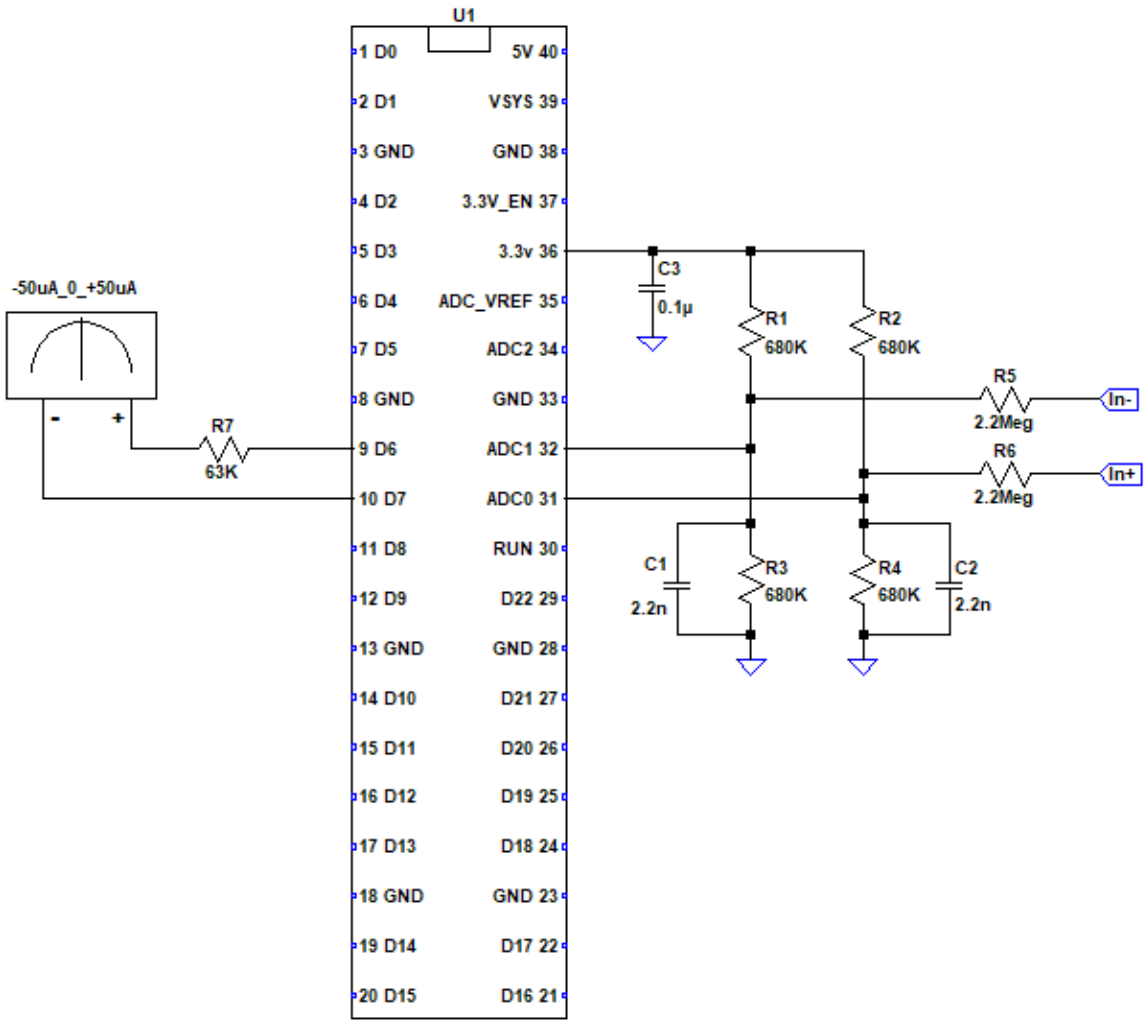
A Raspberry Pi Pico processor will read the analog level, perform a simulation in the loop PID control, correct linearity then output the result to a differential PWM output.



What you need:

- ±50uA Galvanometer (ammeter)
- 4 – 680K 1% resistors
- 2 – 2.2Meg 1% resistors
- 1 – 47K resistor
- 1 – 20K trimmer potentiometer
- 2 – 0.0022uF capacitors
- 1 – 0.1uF capacitor
- 1 – Pico Processor
- Arduino IDE
- Earle Phihower’s Arduino-Pico

The schematic diagram for the meter system.



The first step to improving meter performance is to characterize the following properties:

- The ADC scaling circuit gain and offset
- The Meter linearity
- The Meter dynamics

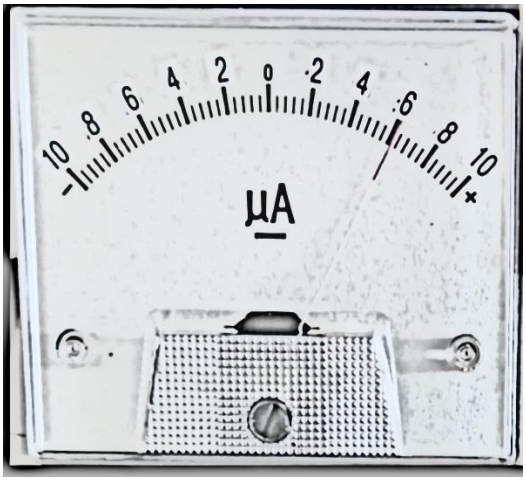
The ADC sampling will act like a resistive input connected to a voltage source. Refer to: https://github.com/simple-circuit/ADC-Input-Scaling for analysis of the ADC input. In our case, the input resistance is about 28Megohms. This will reduce the input gain by about 1%. Resistor tolerance also affects the gain and offset.

To reduce noise, we will average the differential-input signals 100x at about 44Khz. The gain and offset are corrected then the input value is printed for calibration purposes. The ±10 value is then changed to two 10-bit (0 to 1023) values. Both values have zero at 512. One increases with a positive value while the other decreases. These values are sent to PWM outputs on I/Os 6 and 7. The meter filters the 50KHz signal and no additional circuitry is used.

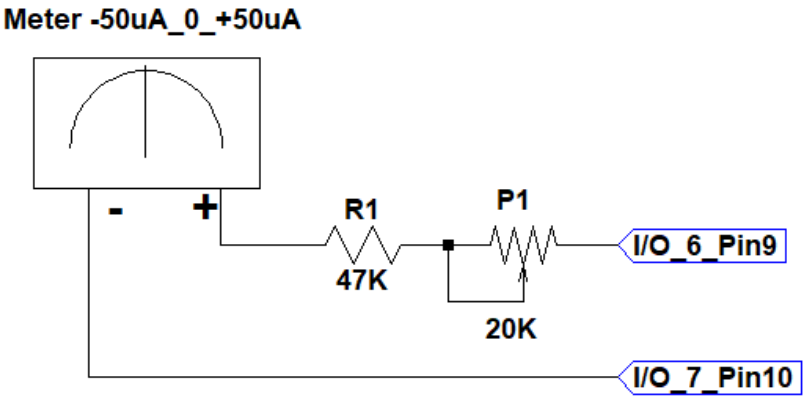
We will start with a program to calibrate the system. The input voltage will be printed through the serial port and the meter +voltage is from I/O6 and the -voltage from I/O7.

```
void setup(void) {
 Serial.begin(115200);
 pinMode(6,OUTPUT);        //+PWM Out
 pinMode(7,OUTPUT);        //-PWM Out
 pinMode(8,OUTPUT);        //Timing test pin
 analogWriteResolution(10);
 analogWriteFreq(50000);    //133MHz/1024=129.8KHz so PWM counter is good for 10-bits
}

void loop() {
  uint32_t dt = 0;
  float voltage;
  int i;

 while(true){
   if (millis() >= dt) {
     dt = millis() + 5; //loop step time = 5ms
     voltage = 0.0;
     for (i=0;i<100;i++){   //Average 100 differential readings at 43.5KHz
      digitalWrite(8,1);
      voltage = float(analogRead(A0)-analogRead(A1)) + voltage;
      delayMicroseconds(12);
      digitalWrite(8,0);
     }
     //Convert 10-bit values to +-10V range
     voltage = voltage*0.0002479+0.0;  //gain=1.03*24.65/1024/100 and offset=0.0

     Serial.println(voltage,2);          //print the voltage for calibration
     analogWrite(6,int(voltage*51.2)+512); //PWM +out 512 +- 512
     analogWrite(7,512-int(voltage*51.2)); //PWM -out 512 -+ 512
  }
 }
}
```

A ±50uA DC meter movement is used. The cover was removed and labels placed on the meter face to indicate -10 through +10V. The meter input will see a -3.3V to +3.3V signal and a resistance of 3.3V/50uA = 66K is needed to move the needle full scale.



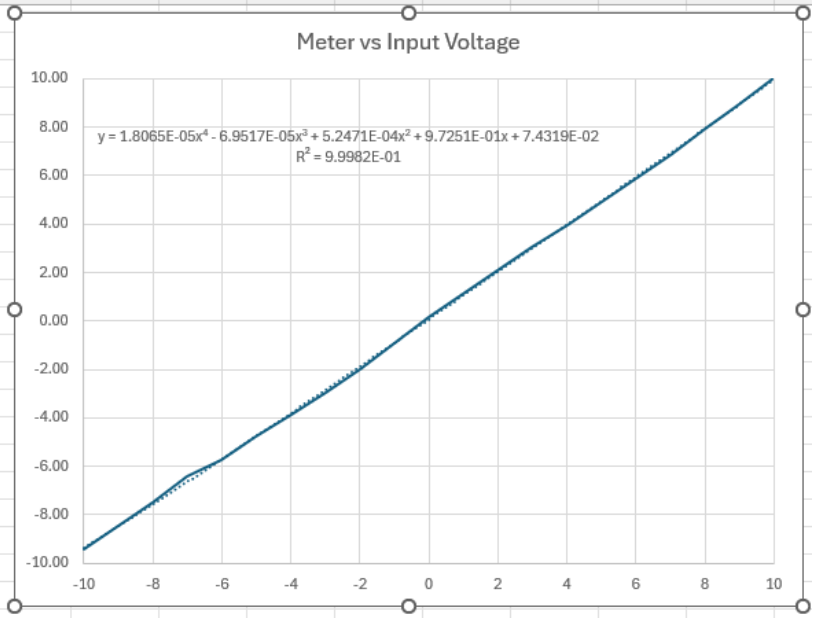
Upload the program into the Pico. Alternately apply -10 and +10V between the 2.2Meg inputs (the minus input can be grounded). Adjust P1 until the needle is at or slightly exceeds ±10V. One side may be less sensitive. Make sure both sides reach 10.



The next step is to correct non-linear indication by the meter movement. Apply a voltage between the 2.2Meg resistor differential inputs. Adjust voltage to cause the meter needle to read from -10 to +10 in steps of one volt. Record the required input voltage at each step then enter the data into a spreadsheet.

Plot the meter value versus the input voltage and generate a trend line polynomial. Try different order polynomials until you get a result that is less than 1% full scale error when compared to the input voltage. For our 20V full scale, 1% is 0.2V. My meter was generally less than 0.1V except for negative 7V where it was 0.2 in error.

	Meter Value	Input Volts	Polynomial
2	-10	-9.44	-9.35
3	-9	-8.44	-8.47
4	-8	-7.51	-7.56
5	-7	-6.44	-6.64
6	-6	-5.74	-5.70
7	-5	-4.78	-4.76
8	-4	-3.89	-3.80
9	-3	-2.94	-2.84
10	-2	-2.00	-1.87
11	-1	-0.90	-0.90
12	0	0.16	0.07
13	1	1.15	1.05
14	2	2.10	2.02
15	3	3.05	3.00
16	4	3.97	3.97
17	5	4.91	4.95
18	6	5.91	5.94
19	7	6.85	6.93
20	8	7.91	7.93
21	9	8.95	8.94
22	10	10.00	9.96



The spread sheet for the calibration values. A fourth order polynomial correction was used.

The meter dynamic movement can be approximated using a second order damped system. First, we need to measure the time to peak. This is the time it takes the needle to move to maximum after a step voltage is applied. Use a -5V to +5V step. Timing with a stopwatch will get you a starting value. I measured an average value of 0.69 seconds. The damped radian frequency is then calculated.

$$\omega_d = \frac{\pi}{T_p}$$

The damped frequency is 4.6 radians per second.

Next, measure the overshoot to calculate the damping factor. The -5 to +5V step is 10V and overshoot is the voltage the needle moves above +5V divided by 10. I measured an average of 6.3 volts peak or (6.3-5)/10=0.13.

$$\zeta = \frac{-\ln(overshoot)}{\sqrt{\pi^2 + ln^2(overshoot)}} = \frac{-\ln(0.13)}{\sqrt{\pi^2 + ln^2(0.13)}} = 0.55$$

$$\omega_n = \frac{\omega_d}{\sqrt{1-\zeta^2}} = 5.5$$

To model the second order damped response, we use a state variable filter. Learn more about this in the video "**Low Pass and Band Pass with 2 Lines of Code"**
https://www.youtube.com/watch?v=ocQqfkSZdyk

Two gains are required: g1 = $2*\omega_n*\zeta = 2*0.55*5.5 = 6.05$; $g2 = \omega^2/g1 = 5.5*5.5/6.05 = 5.0$

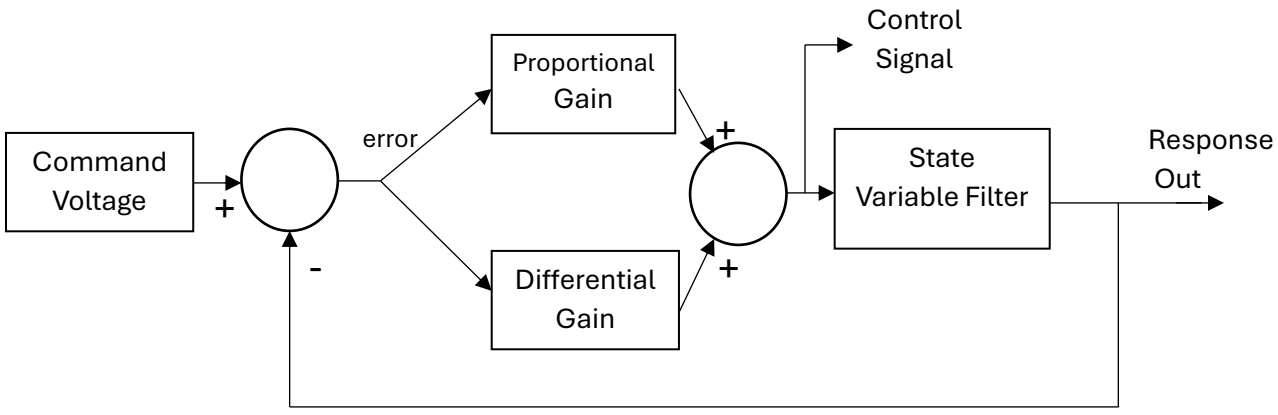
The state variable filter (SVF) requires two equations embedded in a loop with a 5ms period. The x variable is the low pass filter output, and it will be used as the meter control signal.

```
y = y + ((voltage - x - y)  * 0.005 * 6.05);    //dt*g1=5ms*6.05
x = x + (y * 0.005 * 5.0);                      //dt*g2=5ms*5.0
```

To verify the model, you can print the x value and watch a plot compared to the meter movement. I used a TFT display to show the modeled needle position and compared it to the meter.

Once the meter model is verified, and perhaps adjusted for ω_n, a PID control loop can be built around the SVF model. Simulation in the loop is used as it is too complicated to measure the meter needle position for feedback.

For my system, only Proportional and Differential control was required.



Here is the complete program listing. You will need to change the ADC gain and offset values for your system. Also, the coefficients for the correction polynomial. The g1 and g2 gains for the SVF. Finally, the P gain and D gain values for the PID control.

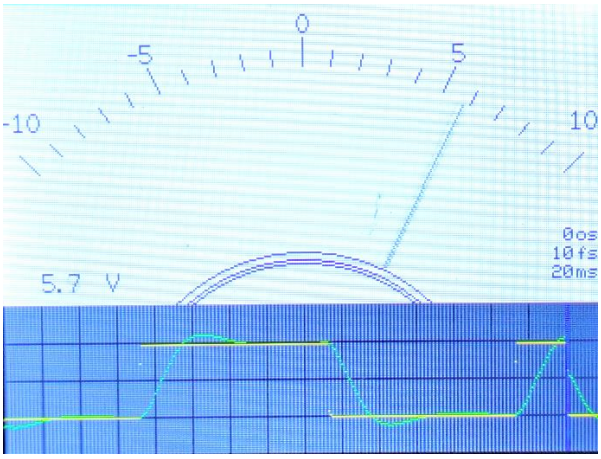
Note: The P gain value affects the overall gain of the system. The formula I P/(P+1) for the negative feedback loop. I use P=60 so the gain drops to 60/61 or 0.984. So, the ADC gain must be increased by 1.017.

```
void setup(void)
{
 Serial.begin(115200);
 pinMode(6,OUTPUT);
 pinMode(7,OUTPUT);
 pinMode(8,OUTPUT);
 pinMode(9,INPUT_PULLUP);
 analogWriteResolution(10);
 analogWriteFreq(50000);
}


void loop()
{
  uint32_t dt = 0;
  float voltage;
  int i;
  float x=0.0;
  float y=0.0;
  float v2,v3,v4,sum;
  volatile float err = 0.0;          //PID Command error
  volatile float perr = 0.0;        //err times P gain
  volatile float derr = 0.0;        //differential error times D gain
  volatile float dgerr = 0.0;      //derr with low pass filter
  volatile float eold = 0.0;       //previous value of err (needed to calculate derr)
```

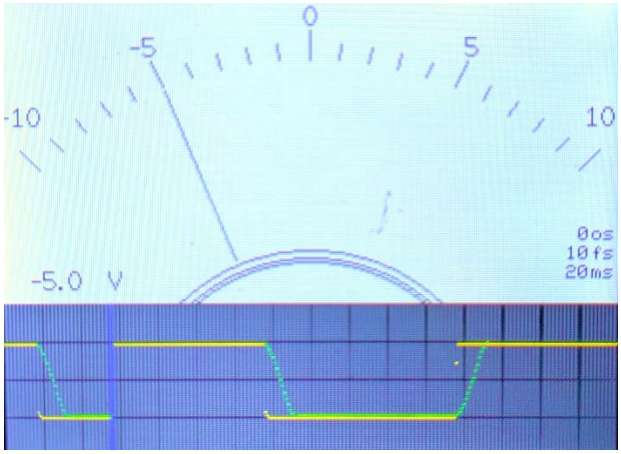
```
while(true){

 digitalWrite(8,0);
 if (millis() >= dt) {
  dt = millis() + 5;              //step time = 5ms
  digitalWrite(8,1);
  voltage = 0.0;                  //Sum differential ADC inputs 100x at 43.5KHz
  for (i=0;i<100;i++){
   voltage = float(analogRead(A0)-analogRead(A1)) + voltage;
   delayMicroseconds(12);
  }
  voltage = voltage*0.00025+0.0;  //1.0385*24.65/1024/100 gain=0.00025 offset = 0.0

  if (digitalRead(9)==1) {                //Turn on PID control if Pin-9 is ungrounded
   err = voltage-x;
   perr = err*60.0;                       //Proportional Control
   derr = 2.0*(err-eold)/(0.005);     //Differential Control
   eold = err;                                //Save current err
   dgerr = (0.5*dgerr) + (derr*0.5);     //LP filter derr
   voltage = perr+dgerr;                  //P+D for control signal
   if (voltage > 10.0) voltage = 10.0;   //Limit control to +-10V
   if (voltage < -10.0) voltage = -10.0;
  }

  //State Variable Filter Meter Simulation: w = sqrt(6.05*5)=5.5, df = 6.05/(2*w)=0.55
  y = y + ((voltage - x - y)  * 0.005 * 6.05); //dt*g1=5ms*6.05
  x = x + (y * 0.005 * 5.0); //dt*g2=5ms*5.0

  Serial.println(x); //print command voltage before linearity correction

  v2 = x * x;     //Polynomial to correct for meter linearity error
  v3 = x * v2;
  v4 = x * v3;
  sum = 0.074319+(voltage*0.97251)+(v2*0.00052471)-(v3*0.000069517)+(v4*0.000018065);

  analogWrite(6,int(sum*51.2)+512); //PWM positive output 512 +- 512
  analogWrite(7,512-int(sum*51.2)); //PWM negative output 515 -+ 512

 }
}
}
```

Performance:

Stimulation with a ±5V square wave. The graphs are for the simulated meter but the real meter responds almost identically.



Simulation step response settles in about 1.1 sec.



Simulation with PD control settles in 0.3 sec.

Sine wave input 10Vpp:

Meter Simulation
Cutoff frequency = 1.0Hz
10% Peak at 0.44H
Settle to 5V in 1.1sec

Meter Simulation with PD control
Fc = 1.75Hz
Peak less than 1%
Settle to 5V in 0.3sec

The response is amplitude sensitive as the slew rate for the meter is fixed by the maximum current drive of ±50uA. Lower amplitude signals have better frequency response. Try experimenting with a drive of ±100uA. You will need to recalibrate the system for the higher drive level and change the PID gains.

Sine, triangle and square wave videos of the meter and simulated meter can be observed on the Simple-Circuit Youtube Video: Improving Analog Meter Performance https://youtu.be/8a4jhd0KbsU