Johnson Noise

Michael Mulhearn

May 15, 2019

1 Introduction

Johnson–Nyquist noise is the electronic noise that results from the thermal excitation of electrons in a conductor independent of any applied voltage. As the applied voltage is zero, there is no net potential difference due to Johnson noise ($\langle V \rangle = 0$), but generally the average power $\langle V^2 \rangle$ is non-zero. As we will show, the one-sided power spectral distribution turns out to be simply:

$$\mathcal{S}_+(f) = 4Rk_{\rm b}T, \tag{1}$$

By measuring all of the other quantities, we will use this relation to experimentally determine Boltzman's constant, which has the value

$$k_{\rm b} = 1.38064852(79) \times 10^{-23} {\rm J/K}$$

with the uncertainty in parenthesis.

2 Autocorrelation and Power Spectrum Distribution

The signal $V(t)$ associated with random noise does not have a Fourier Transform, and so the obvious mechanism for finding the frequency spectrum associated with $V(t)$ is ruled out. Instead, we define the autocorrelation by:

$$\mathcal{R}(\tau) \equiv \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} dt\; V(t)V(t-\tau) = \langle V(t)V(t-\tau) \rangle \tag{2}$$

which is part of a Fourier transform pair:

$$\begin{aligned} \mathcal{S}(f) &\equiv \int_{-\infty}^{\infty} d\tau\; \mathcal{R}(\tau) \exp(-i2\pi f\tau) && (3) \\ \mathcal{R}(\tau) &= \int_{-\infty}^{\infty} df\; \mathcal{S}(f) \exp(i2\pi f\tau) && (4) \end{aligned}$$

To interpret $\mathcal{S}(f)$ we note that Definition 2 implies that:

$$\mathcal{R}(0) = \langle V^2(t) \rangle \equiv P_{\rm avg}$$

but from Equation 4 we also have:

$$\begin{aligned} \mathcal{R}(0) &= \int_{-\infty}^{\infty} df\; \mathcal{S}(f) \exp(i2\pi f0) \\ &= \int_{-\infty}^{\infty} df\; \mathcal{S}(f) \end{aligned}$$

or in other words:

$$P_{\text{avg}} \equiv \langle V^2(t) \rangle = \int_{-\infty}^{\infty} df \; \mathcal{S}(f)$$

That is to say $\mathcal{S}(f)$ is the average power contained at frequency f. To calculate $\mathcal{S}(f)$ we simply calculate the Fourier transform of the autocorrelation function:

$$\mathcal{R}(\tau) \equiv \langle V(t)V(t-\tau) \rangle \, . \tag{5}$$

There are a few practical simplifications we can make resulting from the fact that $\mathcal{R}(\tau)$ is a real even function, and therefore need only calculate:

$$\mathcal{S}(f) = 2\int_0^{\infty} d\tau \; \mathcal{R}(\tau)\cos(2\pi f \tau).$$

But now clearly $\mathcal{S}(f)$ is also an even function, and we can therefore consider the one-sided-power spectral distribution:

$$\begin{aligned} \mathcal{S}_+(f) &\equiv 2\mathcal{S}(f) \qquad (6) \\ &= 4\int_0^{\infty} d\tau \; \mathcal{R}(\tau)\cos(2\pi f \tau) \qquad (7) \end{aligned}$$

which has the same interpretation as the power spectral distribution but simply restricted to positive frequencies:

$$P_{\text{avg}} \equiv \langle V^2(t) \rangle = \int_0^{\infty} df \; \mathcal{S}_+(f) \tag{8}$$

3 Theory of Johnson Noise

Consider a resistor of resistance R with no potential imposed across it. Using the superposition principle, we'll consider that the total potential V across the resistor then results from sum of the voltage caused by each individual electron due to it's current I_i:

$$V_i(t) = RI_i = \frac{Re}{L} u_i(t)$$

where L is length of the resistor and u_i is the velocity along the axis of the resistor. With no applied voltage and therefore no preferred direction we must have $\langle u_i \rangle = 0$. However, because each electron is in thermal equilibrium we have $m \langle u_i^2 \rangle = kT$ and so:

$$\langle V_i^2(t) \rangle = R^2 \frac{e^2}{mL^2} kT.$$

Assuming each electron is uncorrelated with the other electrons,

$$\langle V_i(t)V_j(t) \rangle = 0,$$

for $i \neq j$ and so:

$$\langle V^2(t) \rangle = \sum_{ij} \langle V_i(t)V_j(t) \rangle = \sum_i \langle V_i^2(t) \rangle = R^2 \frac{Ne^2}{mL^2} kT,$$

where N is the number of electrons.

The Fourier transform of $V(t)$ is not defined for continuous random noise. However, we can still determine the power spectral distribution from the autocorrelation function, as discussed above. In this case, the electrons are subject to collisions which alter their trajectory on a timescale τ_c, and we therefore expect the signal to become uncorrelated on that timescale:

$$\mathcal{R}(\tau) \equiv \langle V(t)V(t-\tau)\rangle = A\exp(-\tau/\tau_c)$$

for $\tau \geq 0$, which is all we will need. We note that by definition

$$\mathcal{R}(0) = \langle V^2(t)\rangle$$

and so we must have:

$$\mathcal{R}(\tau) = R^2\frac{Ne^2}{mL^2}kT\exp(-\tau/\tau_c).$$

With the autocorrelation function in hand, the one-sided power spectrum distribution is only an integral away:

$$\begin{aligned}
\mathcal{S}_+(f) &= 4R^2\frac{Ne^2}{mL^2}kT\int_0^\infty d\tau\ \cos(2\pi f\tau)\exp(-\tau/\tau_c) \\
&= 4R^2\frac{Ne^2}{mL^2}kT\frac{\tau_c}{1+(2\pi f\tau_c)^2} \\
&= 4RkT\frac{1}{1+(2\pi f\tau_c)^2}
\end{aligned}$$

Where in the last step we have used the fact the resistance is given by:

$$R = \frac{mL^2}{Ne^2\tau_c}.$$

Noting that $\tau_c \sim 10^{-14}$ s for typical conductors, then at any frequency we are likely to encounter in an electric circuit $f\tau_c << 1$ so that the one-sided power spectral distribution is simply

$$\mathcal{S}_+(f) = 4RkT. \tag{9}$$

4 Experimental Expectations

We saw in the previous sections that the one-sided power spectral distribution is defined by

$$P_{\text{avg}} \equiv \langle V^2(t)\rangle = \int_0^\infty df\ \mathcal{S}_+(f) \tag{10}$$

The average power is something we can easily measure. When we measure the RMS of a signal we are measuring

$$V_{\text{rms}} = \sqrt{\langle V^2(t)\rangle} = \sqrt{P_{\text{avg}}}$$

To make a quantitative prediction for the result of this measurement, we integrate the power spectral distribution across the entire bandwidth for the circuit (all electronic circuits cut off at some high-frequency).

In the case of Johnson noise we found that:

$$\mathcal{S}_+(f) = 4Rk_bT. \tag{11}$$

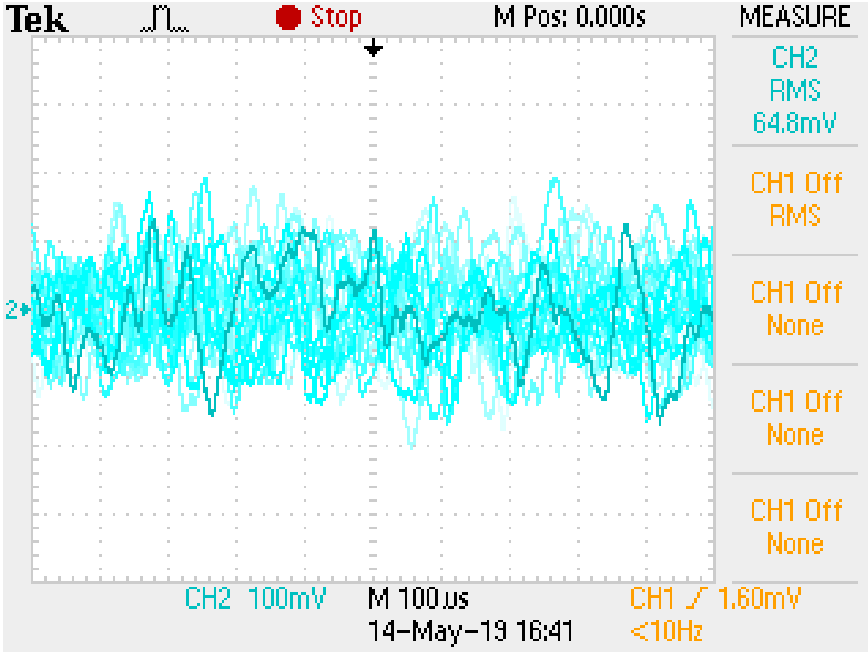


Figure 1: The amplified Johnson noise in a resistor with $R = 1$ MΩ.

which is flat in frequency space. Note that at room temperature:

$$4k_{\mathrm{b}}T = 16.4\frac{\mu\mathrm{V}^2}{\mathrm{kHz\,M\Omega}}$$

so even with a 1 MΩ and a 10 kHz bandwidth, the RMS voltage due to Johnson Noise will be much smaller than 1 mV. We will need to amplify this signal with very high gain in order to see the effect of Johnson Noise. Our gain will have a frequency dependence, even though our power spectral distribution does not, so to calculate the RMS voltage in circuit with gain we must integrate:

$$\begin{aligned} V_{\mathrm{rms}}^2 &= \int_0^\infty df\ g^2(f)\mathcal{S}_+(f) \\ &= 4k_{\mathrm{b}}TR\int_0^\infty df\ g^2(f) \end{aligned}$$

The factor:

$$F \equiv \int_0^\infty df\ g^2(f)$$

is the integral of the gain squared of our circuit across all frequencies. We will determine $g(f)$ for our circuit experimentally using our function generator to input sine waves at fixed frequencies f and measuring the ratio of the output amplitude to the input amplitude. Typical gains for our circuit are from 2000 to 3000 up to a frequency of about 10 kHz, where the gain starts to fall off. The gain integral F is typically in the range 50×10^6 to 300×10^6 kHz. For an $R = 1$ MΩ resistor, theory therefore predicts that our setup would measure an RMS voltage of V_{rms} in the range from 28 to 70 mV. In example is shown in Fig. 1.

For a particular resistance R, we can measure the average spectral density from the gain integral F and the RMS voltage V_{rms} as:

$$S_{\mathrm{meas}} = \frac{V_{\mathrm{rms}}^2}{F}$$

which we expect to follow:

$$S_{\text{meas}} = 4kTR.$$

In the first week of this experiment, you will measure the gain of the Johnson Noise circuit as a function of frequency using the attenuated output from your function generator. This will allow you to calculate the gain integral F. You will then measure the RMS voltage resulting from amplified Johnson noise for a series of different resistors to determine the average spectral density as function of resistance, as in Fig. **??**. By fitting your data to a straight line, you will make your own measurement of Boltzman's constant k_{b}.

5 Johnson Noise Device Gain Measurement

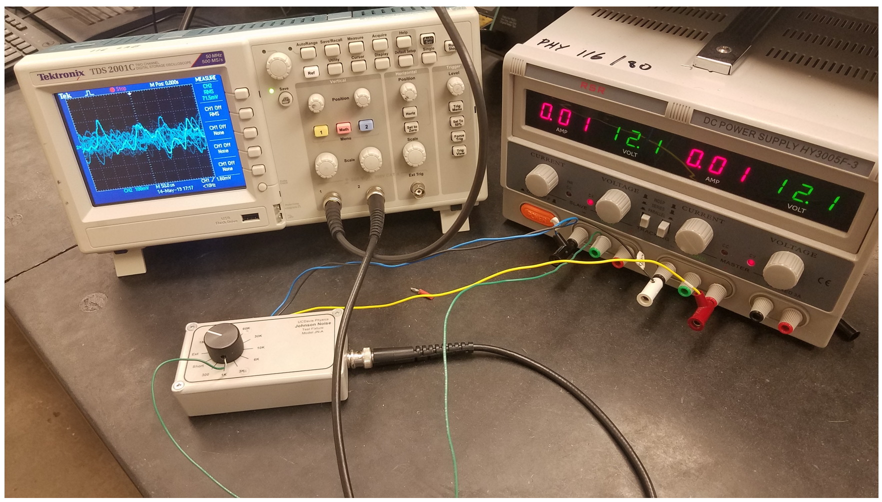


Figure 2: Connections to the Johnson Noise device. Note the green grounding wire used to reduce noise from the knob acting as an antenna!

In this section, we will measure the gain of the Johnson Noise device as function of frequency, which will allow you to determine the gain integral F. Obtain a Johnson Noise (JN) measurement device (Model JN-A or JN-B) and note the serial number (e.g. #4). These are high gain devices (factors of several thousand) that are susceptible to damage. When driven externally, for calibration, an internal 500:1 attenuator is applied to external signal.

The JN devices are very sensitive, easily pickup noise, and ring at a characteristic frequency of about 27 kHz. The tuning knob in particular picks up noise. A segment of wire, grounded at one end, can be held against the set screw to ground the knob, or wrapped around the post under the knob, eliminating this source of noise. The effect can be quite dramatic, particularly later in the experiment, when you are measuring Johnson Noise.

Adjust the supply levels on your bench-top supply to $+12$ V and -12 V. Then, with the supply turned off, connect the JN device to power and ground. Each device should have three banana plugs. The red plug is positive, the black is negative, and the green or white is neutral. Ignore the color of the wires themselves (which vary from device to device). Once connected, power the device on and adjust the current thresholds to just slightly above where the supplies become current limited. Under normal operation, the JN device should draw about $10 - 20$ mA from each supply... keep an eye on it. If one of the supplies is inadvertently disconnected, the circuit latches in a high current state, which should be avoided if possible.

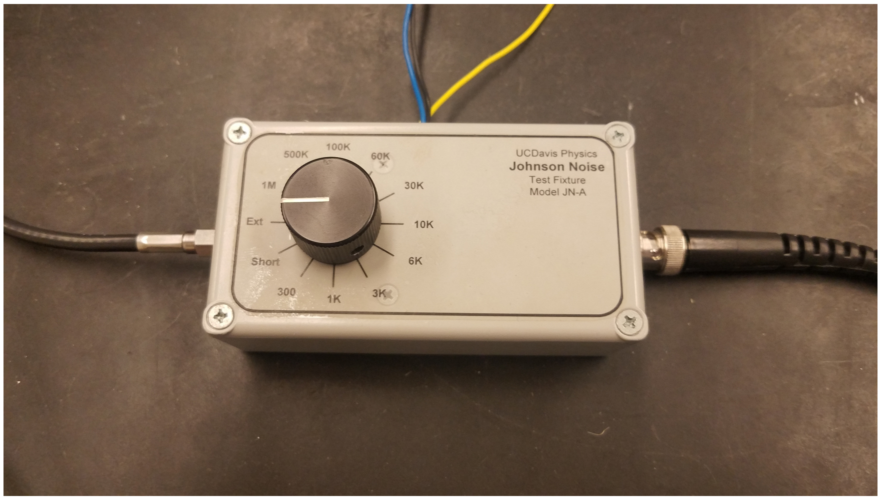


Figure 3: When measuring the gain of the amplifier, you apply an external signal from your function generator (left cable) and turn the dial to external. The external signal is attenuated in the device by a factor of 500.

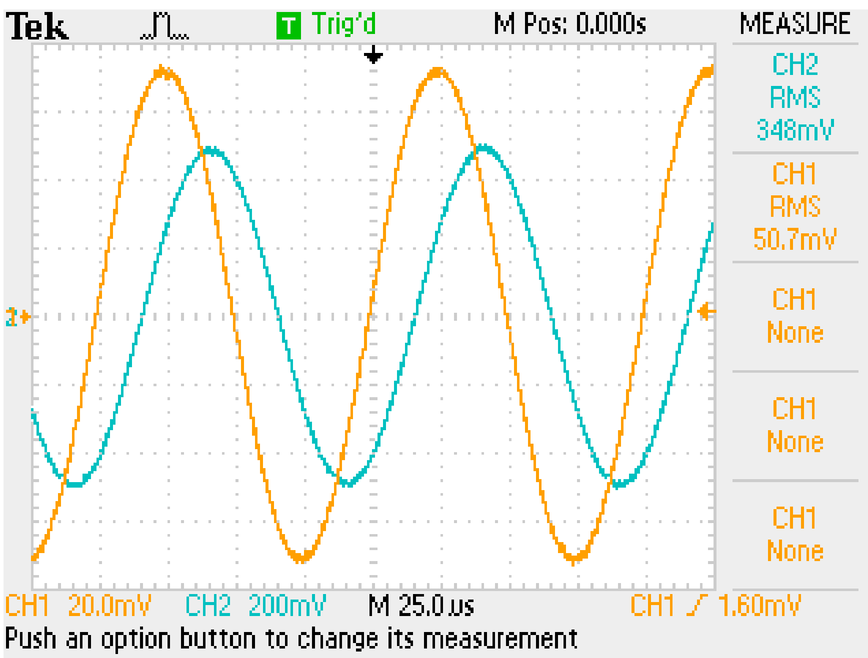


Figure 4: Example gain measurement at 10 kHz.

Set your function generator to produce a sine wave with $V_{\text{rms}} = 50$ mV and $f = 10$ kHz. Connect the function generator to channel 1 of your scope and to the external input of your JN device using a BNC tee connector. Make certain that the JN device dial is set to the "External" mode. The output of the JN device should then be plugged directly into the scope channel 2. Use your scope to measure the RMS voltage of the input and output voltage. When calculating the gain, remember that the external signal has been attenuated by a factor of 500. So for instance, an apparent gain of 4 would in fact be a gain of 2000. Measure the gain as a function of frequency for at least the values $f = 1, 5, 10, 15, 20, 25, 30, 35, 40$ kHz. You might be tempted to try and use your DMM to measure the RMS voltage of your signal, but this is a mistake. But your DMM is designed to measure 60 Hz AC. Use your scope's built-in RMS voltage measurement.

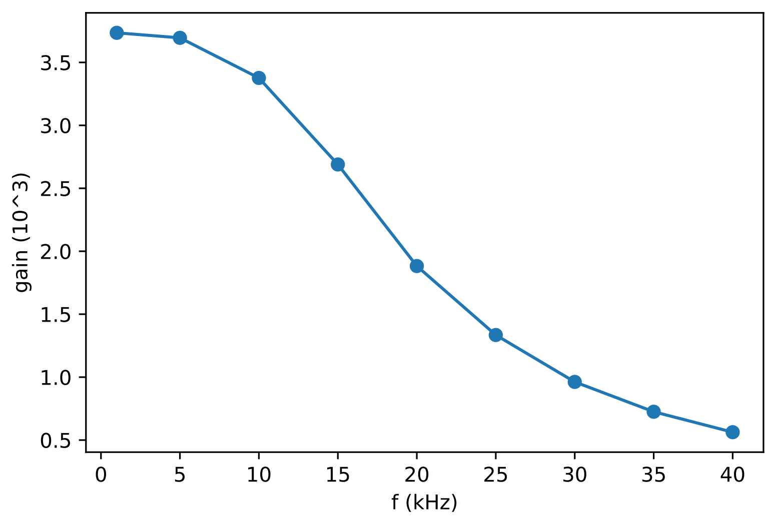


Figure 5: Example gain measurement.

Make a series of gain measurements at different frequencies which will allow to accurately determine the integral

$$F = \int_{-\infty}^{\infty} df\, g(f)^2.$$

A reasonable choice for the frequencies to measure are $f = 1, 5, 10, 15, 20, 25, 30, 35, 40$ kHz. An example gain measurement is shown in Fig. 5. The gain measurement is introduces a major systematic uncertainty in your measurement, so spend some time thinking about how to quantify it. For example, at one frequency, you should measure the gain factor for input signals larger and smaller than the nominal 50 mV input.

6 Johnson Noise Measurement

You are now ready to make your first-pass Johnson Noise measurement. Turn down the function generator and disconnect it from the JN device. Turn the knob to select a resistor, and note the RMS voltage from the scope. Repeat this measurement, for every resistor in the range 1 $M\Omega$ to 30 kΩ. For each measurement, observe the amount that the RMS values fluctuates to estimate a statistical uncertainty.

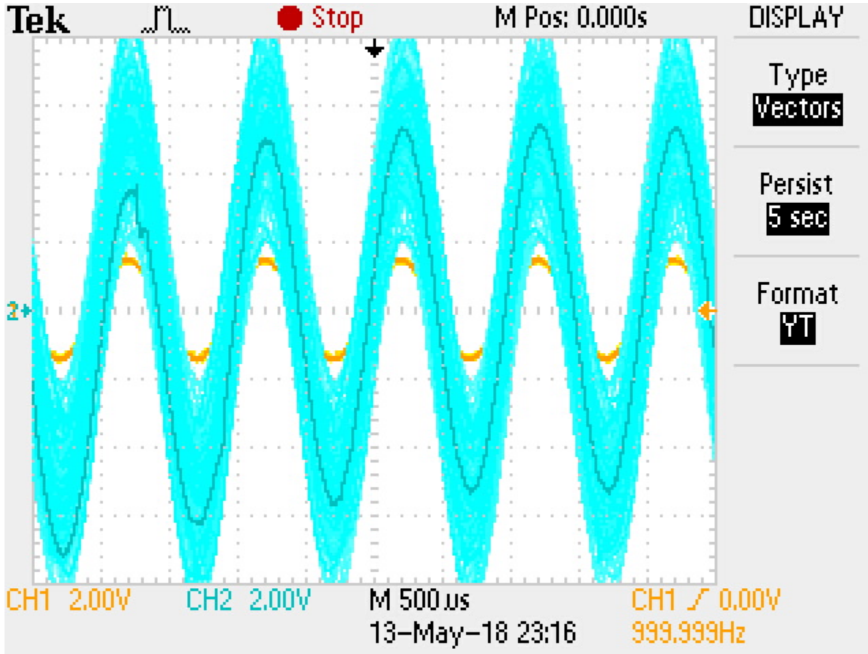


Figure 6: In the past, UCD students using these devices wrestled with horrific ground loop noise, as shown here. Fortunately for you, your instructor has dispatched this unwanted source of noise by attenuating the external input inside the device.

These measurements, combined with the gain measurement, will allow you to calculate k_b. From your gain measurements, you can calculate F as described above. For each RMS voltage measurement V_rms at a particular value of R you can calculate your measured value of the average power spectral density:

$$S_\mathrm{meas} = \frac{V_\mathrm{rms}^2}{F}$$

because this is related to the resistance R by:

$$S_\mathrm{meas} = 4k_bTR$$

you can experimentally determine $4k_bT$ from the slope of S_meas versus R, as shown in Fig. 7. **This is the end point for week one.**

7 The Arduino Power Spectral Analyzer

To further study the features of Johnson Noise, in particular, the flat frequency dependence, you will use an Arduino microprocessor to build a spectrum analyzer. The Arduino will work as a fast digitizer, providing discrete time series data to the PC for further analysis. The PC analysis will use the periodogram feature of Scientific Python to calculate the power spectral density from the discrete time series data.

Suppose $V(t)$ is continuous with some period T so that it is represented by a Fourier Series:

$$V(t) = \sum_{n=-\infty}^{\infty} c_n \exp(i2\pi f_n t)$$

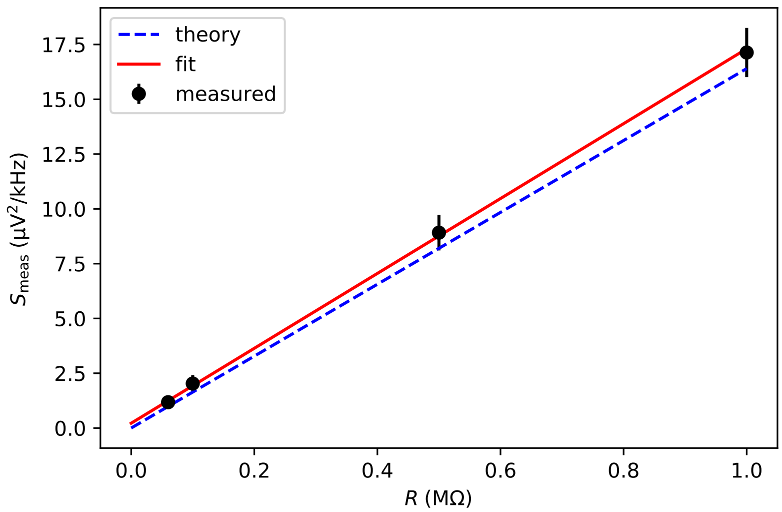


Figure 7: Example fit to determine $4k_bT$.

where $f_n = n/T$. The Fourier coefficients are determined from:

$$c_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} dt\, V(t)\exp(-i2\pi f_n t), \tag{12}$$

and in this context Parseval's theorem is:

$$\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} |V(t)|^2 = \sum_n |c_n|^2$$

or equivalently:

$$P_{\text{avg}} \equiv \langle |V(t)|^2 \rangle = \sum_n |c_n|^2$$

which we can interpret to mean that $|c_n|^2$ represents the average power $P_{\text{avg}}^{(n)}$ at frequency f_n. Since the frequencies associated with each coefficient are discrete values separated by $\Delta f = f_{n+1} - f_n = 1/T$, the power spectral distribution for the discrete Fourier Series is

$$\mathcal{S}(f_n) = \frac{P_{\text{avg}}^{(n)}}{\Delta f} = T|c_n|^2.$$

Furthermore, since we often do not care about the *phase* information in the two-sided power spectral distribution, that is, we don't care to distinguish between power at f_n and power at $-f_n$, the one-sided power spectral distribution is:

$$\mathcal{S}_+(f_n) = \frac{P_{\text{avg}}^{(n)}}{\Delta f} = T(|c_n|^2 + |c_{-n}|^2) \tag{13}$$

It is left as an exercise to show that for

$$f(t) = A\cos(2\pi f_n t)$$

the one-sided power spectral distribution is:

$$\mathcal{S}_+(f_n) = \frac{T}{2}|A|^2 = T A_{\rm rms}^2. \tag{14}$$

We have noted that $A_{\rm rms} = |A|/\sqrt{2}$ This is essential for calibrating a PSD using a function generator set to a specific RMS voltage.

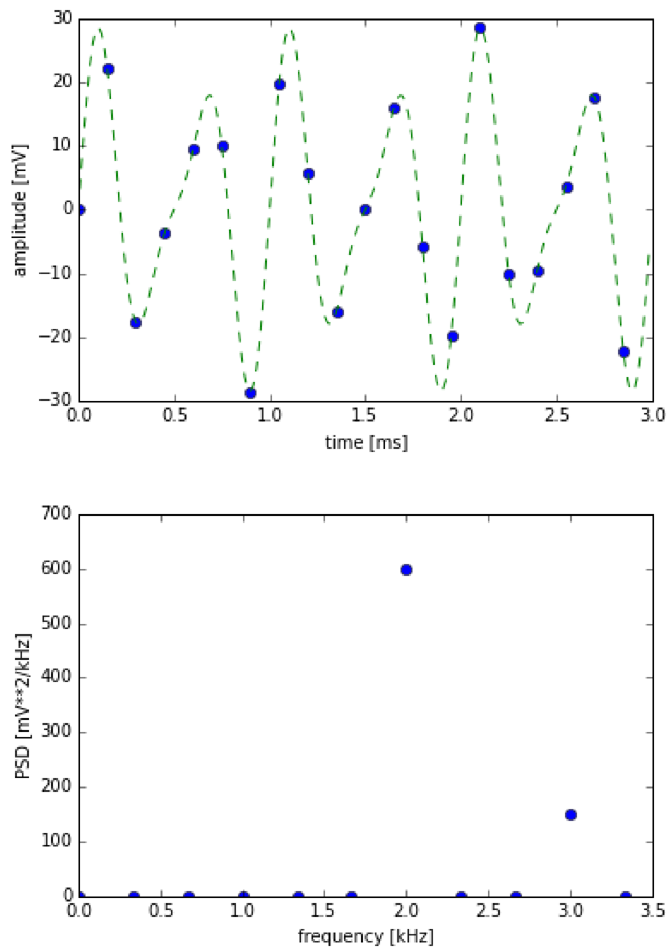


Figure 8: An example periodogram calculated from 20 samples over 3 ms from a signal composed of two sine waves with amplitudes 10 mV and 20 mV. There are two peaks in the periodogram with PSD values $\mathcal{S}_+(f_n) = 150$ and 600 mV^2/kHz as expected from Equation 14.

In many cases, we wish to estimate the PSD from a discrete data set. Instead of a continuous function of time $f(t)$ on the interval T we instead have N samples of f each separated by time $\tau = T/N$, which we record as x_i. We can replace $f(t)$ with its discrete version:

$$f(t) \rightarrow \sum_{m=0}^{N-1} x_m \, \delta(t - i\tau) \, \tau \tag{15}$$

so chosen because any integral over the discrete version

$$\int_{t_1}^{t_2} dt \sum_{m=0}^{N-1} x_m \, \delta(t - i\tau) \, \tau = \sum_{m=m_1}^{m_2} x_m \tau \rightarrow \int_{t_1}^{t_2} dt f(t)$$

approaches the integral over the continuous function in the limit $N \to \infty$. Note that m in the middle sum runs over the indices of the samples in the time interval t_1 to t_2.

Simply inserting the discrete version of $f(t)$ from Equation 15 into the formula for determining the coefficients of the Fourier series Equation 12:

$$\begin{aligned} c_n &= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} dt\, V(t)\exp(-i2\pi f_n t) \\ &= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} dt\; \exp\left(\frac{-i2\pi nt}{T}\right)\sum_{m=0}^{N-1} x_m\, \delta(t-i\tau)\,\tau \\ &= \frac{\tau}{T}\sum_{m=0}^{N-1}\exp\left(\frac{-i2\pi nm\tau}{T}\right) x_m \\ &= \frac{1}{N}\sum_{m=0}^{N-1}\exp\left(\frac{-i2\pi nm}{N}\right) x_m \end{aligned} \tag{16}$$

This is an example of a discrete Fourier Transform (DFT). There are more computationally efficient methods to calculate the coefficients c_n, but they all reproduce this calculation.

The Nyquist-Shannon sampling theorem states that if the highest frequency component of a signal is f_0, and the signal is sampled at a rate $1/\tau \geq 2f_0$ then no information is lost. The Fourier coefficients determined from the DFT can be used to exactly reproduce the original signal at any time. This may seem surprising at first glance. But consider that we are already familiar with a continous function being exactly represented by a discrete set of Fourier coefficients. If a signal has a maximum frequency component f_0, we can think of it's Fourier transform as a periodic function on the interval $(-f_0, f_0)$. It's inverse Fourier transform will then be discrete!

Given N samples x_i taken at sample rate $1/\tau$, we calculate the DFT as the coefficients in Equation 16, then we calculate the one-sided power spectrum distribution as in Equation 13. If we have satisfied the Nyquist-Shannon sampling theorem, only the coefficients corresponding to $1/2\tau$ will be non-zero, so we need only report $\mathcal{S}_+(f_n)$ at the $N/2$ values from $f = 1/T$ to $f = N/2T = 1/2\tau$. The plot of $\mathcal{S}_+(f_n)$ versus f_n for these $N/2$ values is known as periodogram, and is a staple of digital signal processing. An example periodogram resulting from a sine wave is shown in Fig. 8.

Our initial assumption was that the function was periodic with time T, but often we apply periodograms to non-periodic signals, or signals with unknown periodicity. In this case, there is a resolution loss of $1/T$. This effect is mitigated by making T as large as is feasible.

8 Analog Level Shifter and Dynamic Range Adjustment

The Arduino's analog inputs have an input voltage range from 0 to 5 V, but our signals are ground referenced (symmetric about 0 V). A DC level shifter is therefore needed to map the ground referenced output of the Johnson Noise device to the 0 to 5 V range expected by the Arduino analog inputs.

Of course we'll be running the ADC as fast as possible which comes at the price of reduced resolution to 8 bits. That would result in a voltage precision of 5 V$/2^8 \sim 20$ mV. Considering we will be looking at ~ 100 mV signals and smaller, this is simply not enough precision. Fortunately, we have one last trick up our sleeve! The ADC has an (optional) external voltage reference, which sets the scale used by the ADC. Experimentation showed that the ADC works reliably down to a voltage reference of ~ 780 mV, which, at 8-bit precision, would give us a 3 mv voltage precision.

The low-pass filter in Fig. 9 is used to convert PWM output (on pin 5 of the Arduino) into a DC reference voltage $V_{\rm ref}$. This reference voltage is used as (1) the reference for the ADC, at pin AREF, setting the dynamic range of the ADC, and (2) the reference to the DC level shifter circuit in Fig. 10 so that input to the ADC is referenced to $V_{\rm ref}/2$. In this way, we can adjust $V_{\rm ref}$ to match the dynamic range of our signal, and the DC offset will be set to the middle of this range. This will ensure that even for small signals, the full 8-bit precision of the ADC is being put to use.

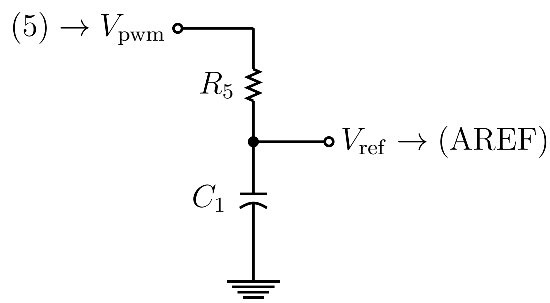


Figure 9: A low pass filter used to convert the Arduino PWM output (on pin 5) to an analog DC reference voltage (sent to pin AREF and also used by the DC level shifter.)

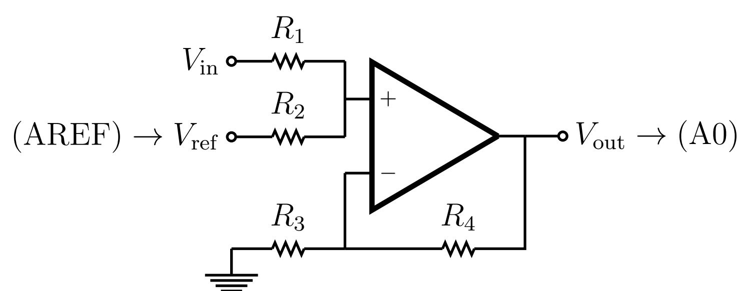


Figure 10: A DC level-shifter with unit gain. The points (AREF) and (A0) refer to the Arduino pins, which should be connected as shown.

For our op-amp we will use a LM358 low-power dual op-amp IC with pinout in Fig. 11. The LM358 is designed to run off a single supply down to 5 V, so we can power this circuit entirely from the Arduino. This is not just convenient, it also ensures that it's output cannot exceed the 0 to 5 V range expected by the Arduino.

You will build the circuits in Fig. 9 and 10 using $R_1 = R_4 = 15$ kΩ, $R_2 = R_3 = 33$ kΩ, $R_5 = 1\ k\Omega$. For C_1 use a large polarized capacitor ($C_1 = 330\ \mu$F) and **make certain that the negative terminal is connected (as shown in the circuit) to ground.** Connect the positive supply of the Op Amp to the 5 V supply of the Arduino and the 0 V supply to the Arduino ground.

A light-weight sketch CircuitTester is provided which is useful while developing your circuit. This sketch implements an RMS voltmeter (significantly better, if less durable, than the one provided by your DMM.) based on the circuit you are are building. It provides control of the PWM on pin 5, reads the analog input on A0, and regularly reports the RMS voltage (and other useful quantities) on the serial output. The level of the PWM is specified in the parameter RUN_SCALE. A value of 0 results in 0% duty-cycle for the PWM, a value of 255 corresponds to 100% duty-cycle.

While testing your circuit, you should use your function generator to drive the input $V_{\rm in}$ directly. A 5 kHz sine function with an RMS amplitude of 100 mV is a good starting point. Make sure the ground of the function generator is connected to the ground of your Arduino.

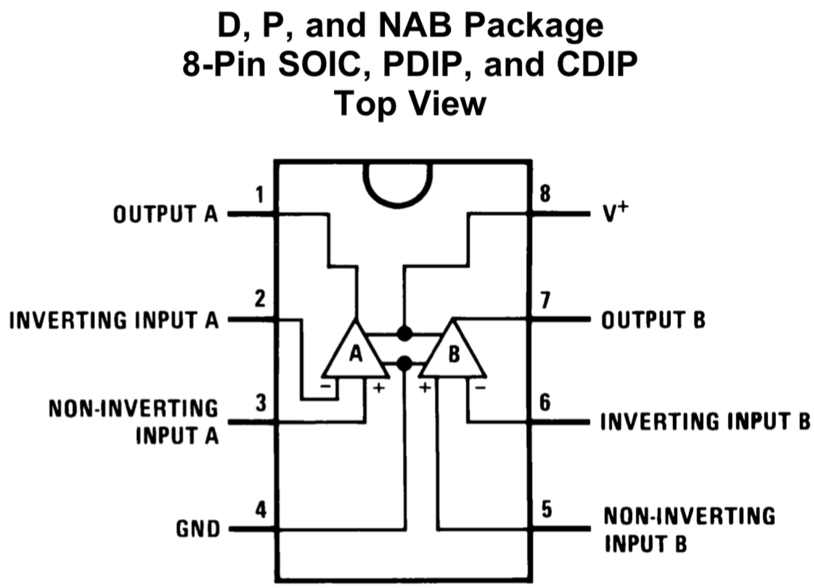
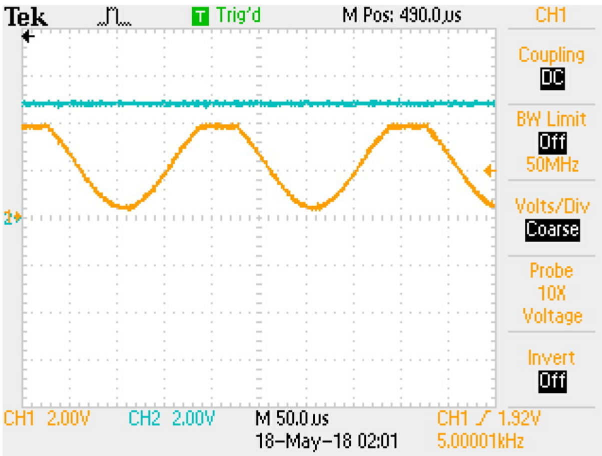


Figure 11: Pinout for the LM358 dual-op-amp with single-ended supply. We'll only need one of the op-amps.

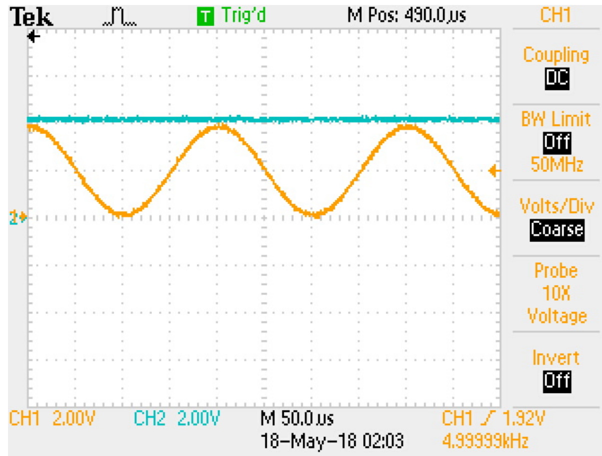
CircuitTester §

```
//
//  CircuitTester:  This is a stripped down sketch useful for testing
//      your circuit during construction... it continuously samples
//      the analog input and displays the mean and rms on the serial port.
//

// You should only have to edit these parameters, if at all...

const float CALIB_MV = 5000;   // Calibrated MV scale (nominal 5000 mV)
const int MAX_SCALE     = 220;  // Calibrated full scale: minimal clipping
const int MIN_SCALE     = 40;   // Calibrated min scale: ADC fails below
const int RUN_SCALE     = 80;

```

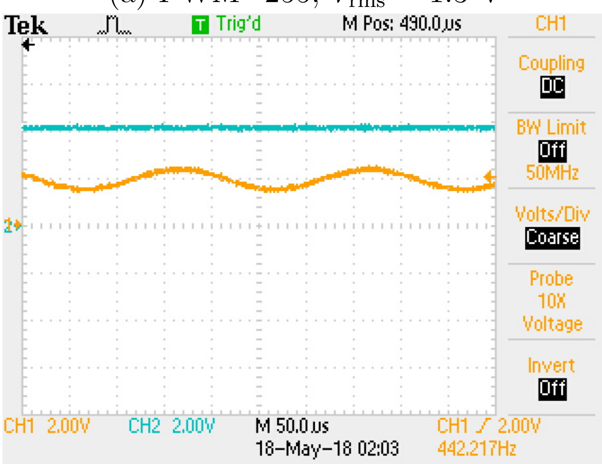
Figure 12: The CircuitTester sketch is useful for developing, testing, and calibrating your circuit. The sketch implements an RMS voltmeter that outputs directly to the Serial Monitor. In the calibration section below you will calibrate CALIB_MV which is set initially to the nominal 5 V scale to match the actual full scale on your Arduino. The scale refers to the PWM output which determines the reference voltage for the ADC. A good all-around working point for this experiment is 80.



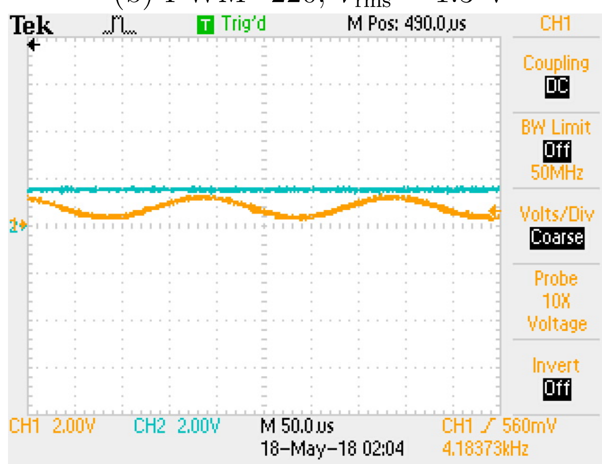
(a) PWM=255, $V_{\mathrm{rms}} = 1.3$ V



(b) PWM=220, $V_{\mathrm{rms}} = 1.3$ V



(c) PWM=220, $V_{\mathrm{rms}} = 300$ mV



(d) PWM=80, $V_{\mathrm{rms}} = 300$ V

Figure 13: Output of DC level-shifter compared to V_{ref} for various settings of the PWM analog output level and the function generator input RMS voltage. In (a) the LM358 output does not reach the rail at 5 V, so the ADC dynamic range above this cutoff is wasted. In (b) the PWM setting at 220 adjusts V_{ref} to the full output range of the LM358, but (c) for low amplitude signals, there is still wasted dynamic range. In (d) the PWM setting at 80 matches the dynamic range of the signal. Notice how the DC offset is automatically adjusted as well!

By now we've learned to divide and conquer technical challenges! I'd propose something like:

- Leave the Arduino unpowered.
- Build the level shifter as shown but at first (1) set the reference voltage to 5 V and (2) connect the output to channel one on your scope instead of pin A0.
- Power up the Arduino.
- On your scope you should see the function generator referenced to approximately 2.5 V.
- Power down the Arduino.
- Build the lowpass filter as shown, including driving the circuit from the PWM pin (pin 5) of the Arduino, but connect V_{ref} to channel two on your scope instead of to pin `AREF`.
- Power up the Arduino and download the CircuitTester sketch, setting the parameter `RUN_SCALE` to 124 (one-half duty cycle)
- On your scope, you should observe V_{ref}, it should be at about 2.5 V with very little ripple (use AC line trigger and DC offset).
- Adjust the `RUN_SCALE` parameter and ensure you have the expected behavior.
- Power down the Arduino.
- Connect the lowpass filter output V_{ref} to both (AREF) and the DC level-shifter input V_{ref}.
- Power up the Arduino.
- Check that level-shifter output has the correct DC level when you adjust `RUN_SCALE`.
- Connect the output of the DC level-shifter to the Arduino analog input A0.
- Start the Serial Monitor.

Once the whole circuit is built, the Arduino is running the `CircuitTester` sketch, and the Serial Monitor is open, you should observe periodic RMS voltage readings that should correspond approximately with the output of the function generator. If it appears to be generally working but a bit inaccurate, you are ready for calibration.

9 Calibration of Arduino Circuit

An example of uncalibrated output from the CircuitTester sketch when connected to a $V_{\mathrm{rms}} =$ 300 mV input signal is shown in Fig. 14. By design, the mean in raw ADC counts should be near 255/2 within the accuracy of out 10% resistors. But the calibrated output in millivolts will be systematically off from the correct value to start. To calibrate, simply note the discrepancy and scale the `CALIB_MV` parameter accordingly. Once calibrated, you should get accurate measurements across a range of frequencies, amplitudes, and even `RUN_SCALE` settings, as shown in Fig. 15. Take note of the value of CALIB_MV as you will need to use it in the PSD python driver as well.

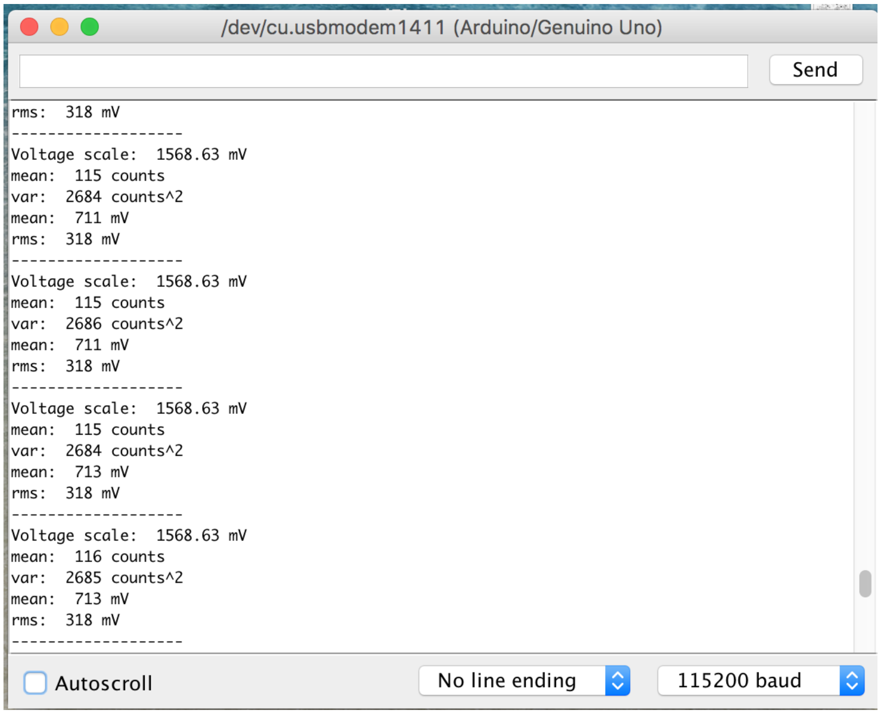


Figure 14: Uncalibrated Serial Monitor output for an input signal with $V_{\rm rms} = 300$ mV. It is systematically off at 318 mV due to full scale being different than nominal 5 V. The calibration is made by changing the value of `CALIB_MV` from 5000 to something more appropriate. The mean, in RAW counts, is right where it should be: near 255/2.

```
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  2687 counts^2
mean:  671 mV
rms:  300 mV
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  2683 counts^2
mean:  670 mV
rms:  300 mV
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  2688 counts^2
mean:  670 mV
rms:  300 mV
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  2685 counts^2
mean:  672 mV
rms:  300 mV
-------------------
```

(a)

```
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  295 counts^2
mean:  672 mV
rms:  996/10 mV
-------------------
Voltage scale:  1479.53 mV
mean:  116 counts
var:  295 counts^2
mean:  673 mV
rms:  996/10 mV
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  294 counts^2
mean:  673 mV
rms:  995/10 mV
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  294 counts^2
mean:  672 mV
rms:  996/10 mV
-------------------
```

(b)

```
-------------------
Voltage scale:  1479.53 mV
mean:  115 counts
var:  11 counts^2
mean:  673 mV
rms:  193/10 mV
-------------------
Voltage scale:  1479.53 mV
mean:  116 counts
var:  10 counts^2
mean:  673 mV
rms:  186/10 mV
-------------------
Voltage scale:  1479.53 mV
mean:  116 counts
var:  10 counts^2
mean:  673 mV
rms:  186/10 mV
-------------------
Voltage scale:  1479.53 mV
mean:  116 counts
var:  10 counts^2
mean:  673 mV
rms:  186/10 mV
-------------------
```

(c)

```
-------------------
Voltage scale:  739.76 mV
mean:  115 counts
var:  45 counts^2
mean:  336 mV
rms:  195/10 mV
-------------------
Voltage scale:  739.76 mV
mean:  115 counts
var:  45 counts^2
mean:  336 mV
rms:  195/10 mV
-------------------
Voltage scale:  739.76 mV
mean:  115 counts
var:  45 counts^2
mean:  336 mV
rms:  195/10 mV
-------------------
Voltage scale:  739.76 mV
mean:  116 counts
var:  45 counts^2
mean:  336 mV
rms:  194/10 mV
-------------------
```

(d)

Figure 15: Post-calibration output for an input signal with (a) $V_{\rm rms} = 300$ mV (b) $V_{\rm rms} = 100$ mV (c) $V_{\rm rms} = 20$ mV (d) $V_{\rm rms} = 20$ mV. In (a)-(c), the PWM analog output level is set to 80, a good choice for the gain calibration. In (d) the range has been reduced to 40, resulting in better precision. Also note that, for example, 186/10 mV should be read as 18.6 mV... it seems `sprintf` with float is not supported by the Arduino IDE.

10 Using the Arduino Power Spectral Analyzer

When your circuit is working and calibrated, you are ready to update the software to provide the Power Spectral Analyzer capability. There is a new sketch (FastAdc) and a command line python driver (psd.py) to provide this capability (see Fig. 16). The FastAdc sketch has the `RUN_SCALE` parameter used to set the analog voltage scale, but the conversion to millivolts is done in the python driver, so you should set `calib_mv` in the file `psd.py` to the calibrated value you determined in the previous section. As in the Fourier lab, if you tire of entering your COM port each time, you can hard code it in the python driver. Example output from running the python driver is shown in Fig. 17. As shown in Fig. 18 the PSD performance is quite respectable, which you can see by scanning the frequency at a constant amplitude.

You should examine the output when the frequency goes above the Nyquist frequency (38.45 kHz in our case). You should see clear aliasing of the high-frequency signal at a lower frequency. Fortunately, the low-pass filter in the Johnson Noise device eliminates problematic high-frequency aliasing during our experiment.

FastAdc §

```
//
// Fast ADC with PWM controlled ADC dynamic range adjustment.
//

// You should only have to edit these parameters, if at all...

const int MAX_SCALE    = 220;  // Calibrated full scale: minimal clipping
const int MIN_SCALE    = 40;   // Calibrated min scale: ADC fails below
const int RUN_SCALE    = 80;   // User selected value for PWM
const bool TEST_PATTERN = false; // Send test pattern instead of ADC data
//
// You shouldn't have to change anything below here...
//
```

Figure 16: The FastAdc sketch is used together with the psd.py python driver on your PC. The sketch only uses raw adc counts, but the calibration calib_mv in psd.py should match what you determine with CircuitTester. The PWM output which controls the analog voltage reference is controlled by the RUN_SCALE parameter. A good all-around working point for this experiment is 80. Note that you can optionally send test pattern data instead of the analog input, for example to test the PC interface, by setting the `TEST_PATTERN` flag to true.

11 Improved Johnson Noise Device Gain Measurement

In this section, you will use your spectrum analyzer to determine the gain curve more precisely than in the previous week. Setup the Johnson Noise device (ideally the same one you used the previous week) for an external input, taken from the function generator at 10 kHz and an RMS voltage of 50 mV. Connect the output of the Johnson Noise device to the input of your Arduino spectrum analyzer.

Using your Arduino spectrum analyzer , as shown in Fig. 19. In the `psd.py` file, you should set the `nruns` parameter to 10 to average 10 runs per measurement (default is one run only), reducing statistical fluctuations in your power spectrum. You will use these samples to determine the gain of the JN device as function of frequency.

```
connecting to the Arduino...
Arduino Initialized
Press Enter to Acquire...
1
1
sending acquire command:  a 1 1
receiving payload from Arduion of length  1538  x  1

voltage scale is  80
run  0
x:  0 y:  124
x:  1 y:  114
x:  2 y:  104
x:  3 y:  97
x:  4 y:  92
sample rms is  99.6950387241
peak at f= 5.0  with Vrms= 99.5284272594
100
saturated count is  0
total integrated rms is  99.6950387241
in narrow region of peak is  99.6515233356
in wide region of peak is  99.6598663342
mean of PSD:   258.15846094 mV^2/kHz
mean of PSD in region:   1985.58230966 mV^2/kHz
```

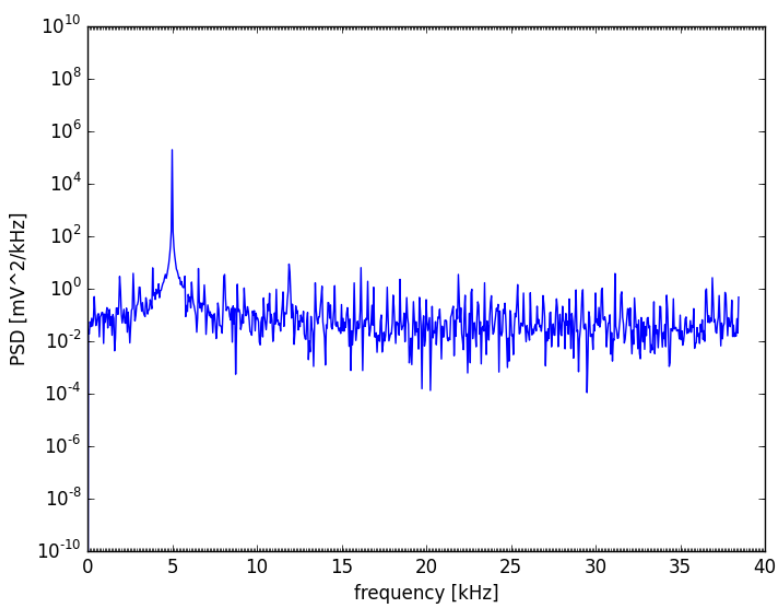


Figure 17: The python serial driver psd.py as seen at the command line (top) and the (bottom) power spectral distribution plot it produces. This output is for an input with $V_{\rm rms} = 100$ mV and $f = 5$ kHz.

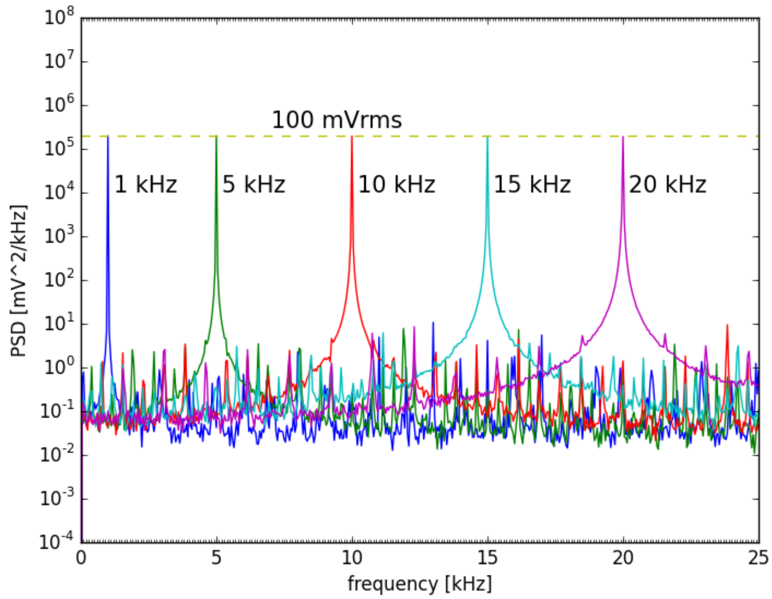


Figure 18: The response of the Arduino PSD.

By default, the python driver saves the raw data it collected from the Arduino to the txt file `raw_waveforms.txt`. If you enter a tag like `g10khz` or `r100k` instead of just pressing enter when prompted prior to data acquisition, the output file will have your tag appended in the name (to avoid confusion, don't include any whitespace in your tag). This is convenient for the gain calibration where you will be collecting lots of samples at different frequencies. An example python notebook for reproducing the PSD and time series plots from the data saved in such a text file is included on the course website (`plottxt.ipynb`).

Also make certain that your function generator is set to high impedance output... otherwise you gain will be off by a factor of two, which will be hard to plausibly reconcile with the benchmarks.

12 Johnson Noise Measurement

With your instrument calibrated, you are ready to collect Johnson noise data with your spectrum analyzer for a number of different resistor values. Example data is shown in Fig. 20. You should attempt to demonstrate at least the following three things:

- The flat frequency spectrum of Johnson Noise, as far out as you can, as in Fig. 21
- The linear relation between Johnson Noise and the resistance.
- An experimental measurement of k_b, including an uncertainty.

Remember to average across multiple runs to reduce statistical fluctuations, and use the tagging feature to keep the raw data file organized.

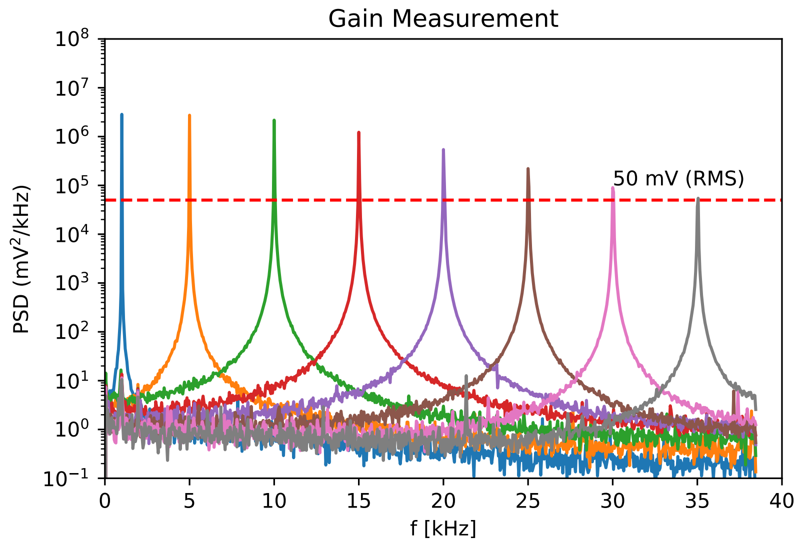


Figure 19: Data collected for gain versus frequency scan.

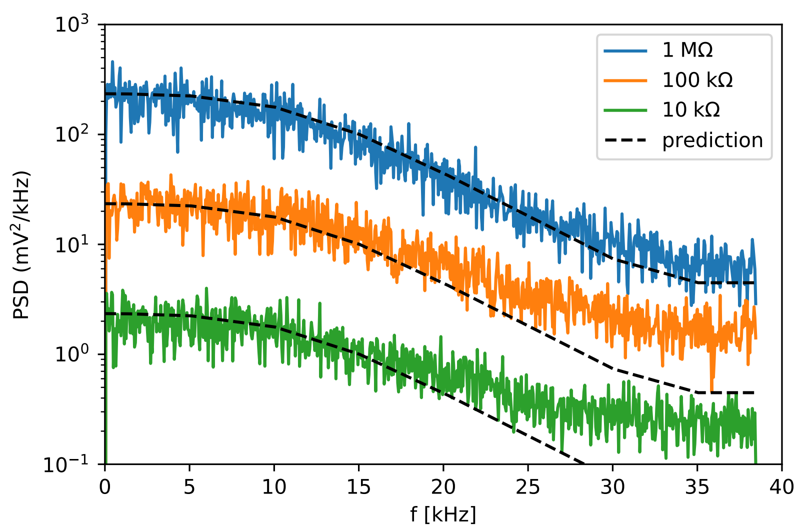


Figure 20: Example data collected for Johnson Noise measurement, compared to the expectation including the measured gain.

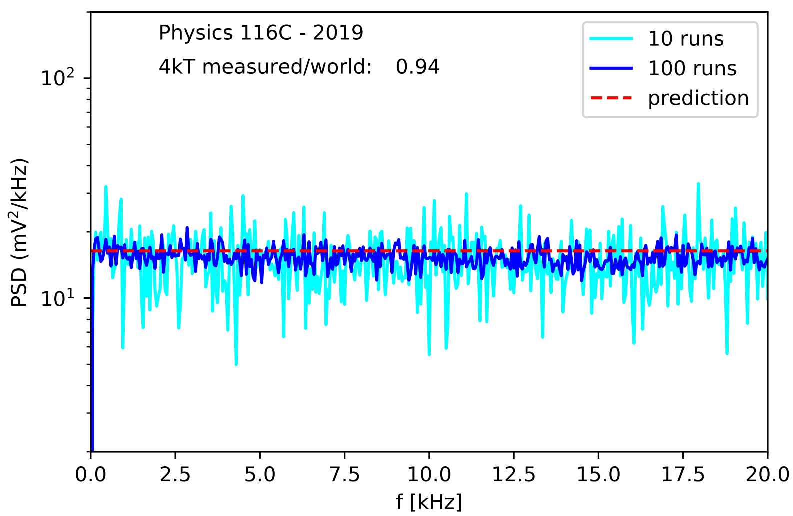


Figure 21: The flat response of Johnson Noise and measured value of $4kT$.