Temperature Dependence

Josh Wilkins -- 10/04/17

```
# Imports↔
```

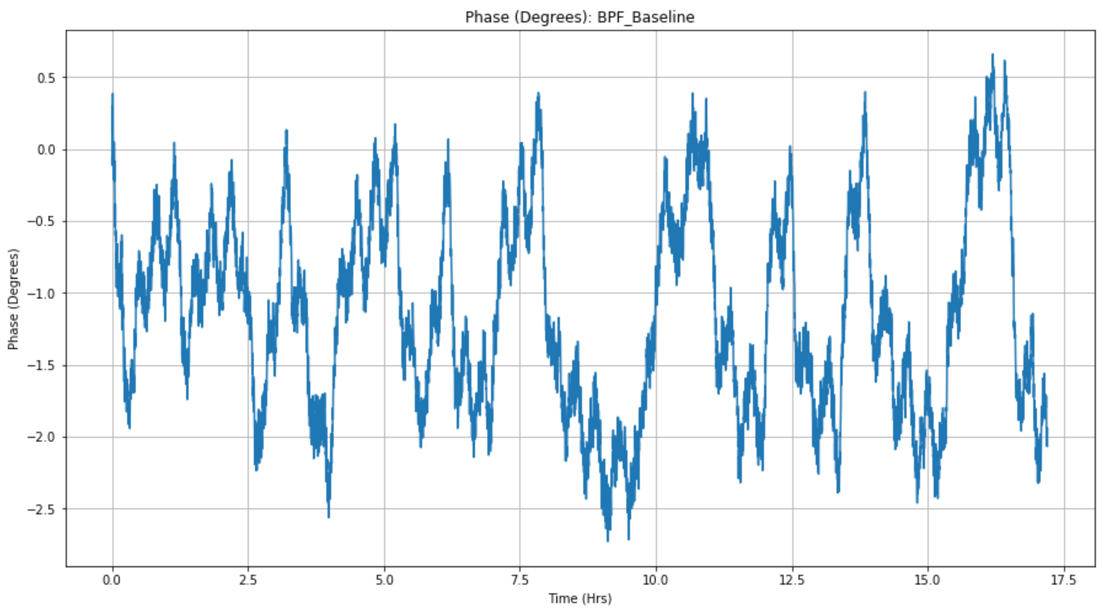
```
#CSS↔
```

```
# Custom Methods

def ADev(file_name, sample_rate, filt=False, plot_adev=False):
    # file_name: 'testfile.csv'
    # sample_rate: x Hz

    data = csvData(file_name).getData()
    time = data[:,0]      # In Seconds
    phase = data[:,2]     # In Degrees

    if filt:
        plt.plot(time/3600, phase)
        N = 100 # Moving avg filter window
        phase = np.convolve(phase, np.ones((N,))/N, mode='valid')
        time = np.linspace(time[0], time[len(time)-1], len(time)-N+1)

    print "Peak to Peak: %.2f°" % (max(phase) - min(phase))

    # Plotting the Phase in degrees vs time in hrs
    plt.plot(time/3600, phase)
    plt.xlabel('Time (Hrs)')
    plt.ylabel('Phase (Degrees)')
    plt.title('Phase (Degrees): ' + file_name[:-4])
    plt.grid(which='both')

    # Converting phase to seconds
    phase /= 360.*6.4e9

    if plot_adev:
        # Allan Deviation
        plt.figure()
        (t2, adev, adev_error, adev_n) = allantools.oadev(phase, rate=sample_rate, taus='all
        plt.loglog(t2, adev)

        plt.xlabel('Tau')
        plt.ylabel('Allan deviation')
        plt.title('Allan Deviation: ' + file_name[:-4])
        plt.grid(which='both')
```

BPF Results

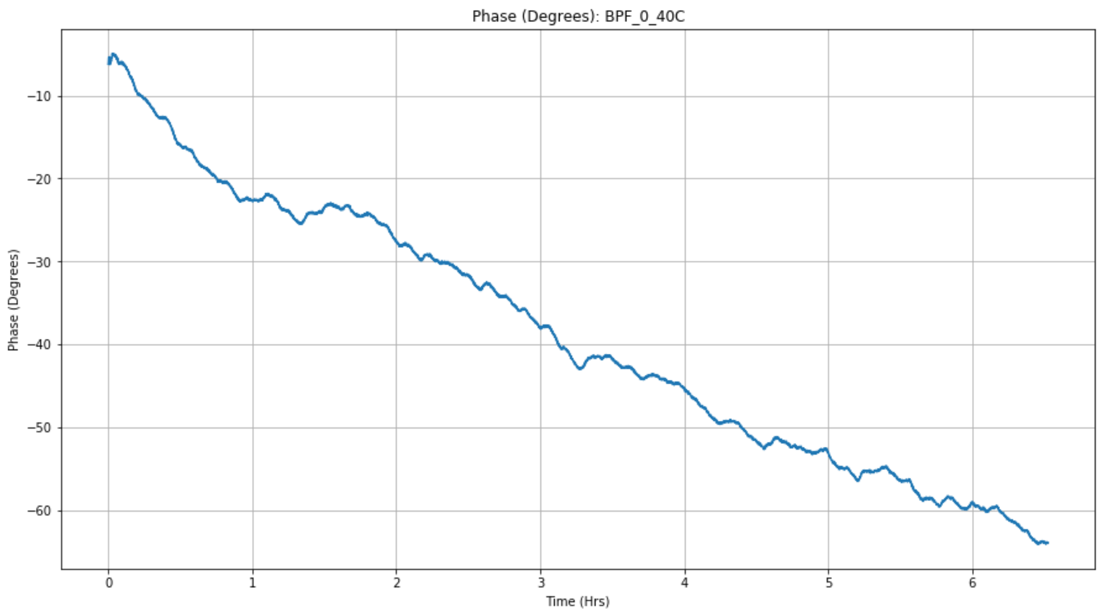
```
▸ # BPF Baseline↔
```

```
Peak to Peak: 3.39°
```



```
▸ # BPF 0 to 40.4 °C↔
```

```
Peak to Peak: 59.15°
```



From the results above, we can determine that the temperature dependence of the divider is

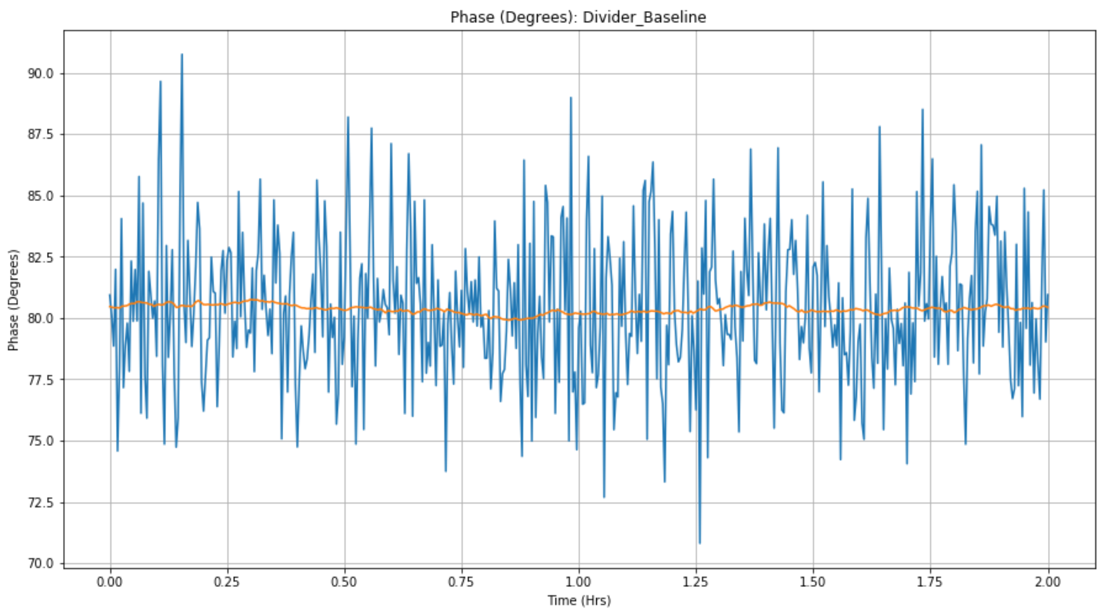
$$\left(\frac{60^\circ}{40^\circ C}\right) = \frac{1.5^\circ}{^\circ C}$$

Divider Results

The setup used to find the temperature dependence of the divider was a little different. The S1 port of the Network Analyzer (NA) was sent through the divider and the divider's 800 MHz output was sent back to the S2 port of the NA. The NA is thus observing the 8th harmonic of the 6.4 GHz signal which means the following results are scaled by a factor of 8.

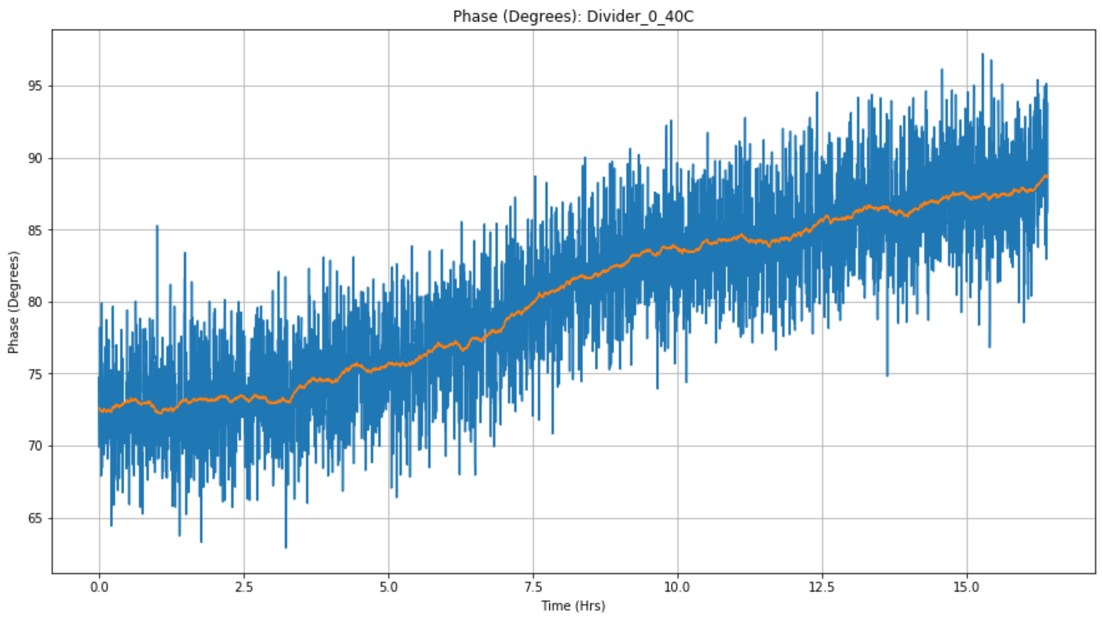
```
▸ # Divider Baseline↔
```

```
Peak to Peak: 0.84°
```



```
▸ # Divider 0 to 40°C↔
```

```
Peak to Peak: 16.56°
```



Given that these results are scaled by a factor of 8, we can determine that the temperature dependence of the divider is

$$\left(\frac{16^\circ}{8 * 40^\circ C}\right) = \frac{0.05^\circ}{^\circ C}$$

This concludes that the divider is about a 30 times less temperature dependent than the BPF!

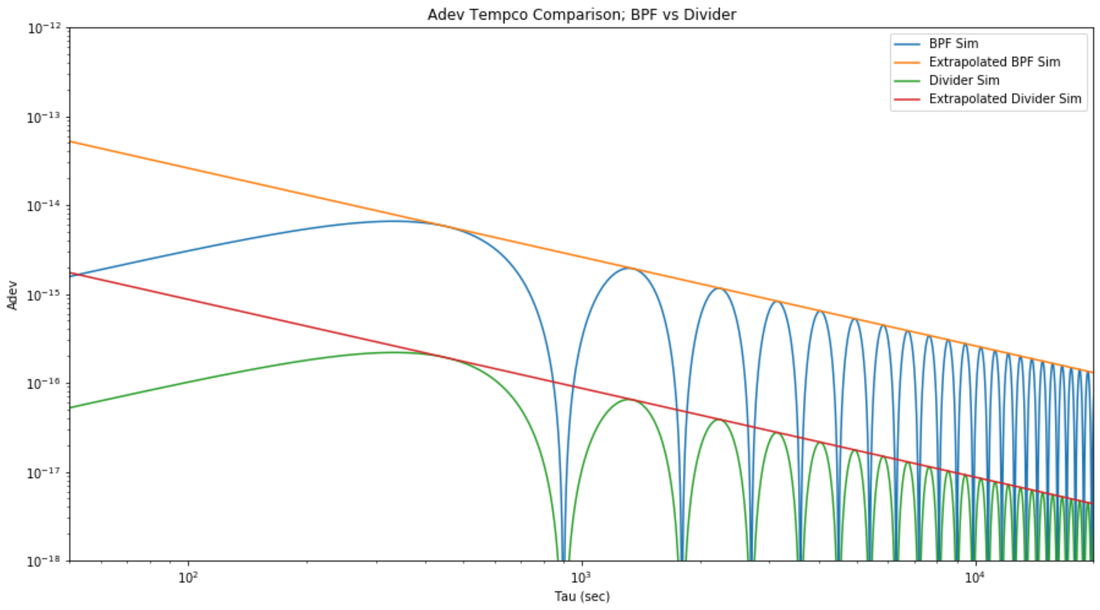
Approximate Adev Floor of Synth

A simulated allan deviation plot was created using a sine wave with an equivalent amplitude of 4°C peak to peak and a 15 minute cycle time. This is a rough approximation of the effect the A/C might have on the synthesizer.

It should be noted that the BPF was tested inside 'Hotel Guan' and it may have a thermal resistance high enough to slow the effect of ambient temperature changes.

```python
# Temperature Effect Method Definition
def temp_effect(T, dC, tempco, plot=False):
    # T - Time Interval for temperature change [minutes]
    # dC - Temperature change over interval T (Peak to Peak) [°C]
    # tempco - Temperature coefficient of device [°/°C]

    T *= 60   # Convert time interval to sec
    f = 1./T  # Convert to freq [Hz]

    dphase = float(dC)/2 * tempco # Phase change over interval T [°]

    num_samples = 100000
    num_periods = 64
    x = np.linspace(0, num_periods*T, num_samples)
    y = dphase*np.sin(2*np.pi*f*x)
    phase = y/(360*6.4e9)

    if plot:
        plt.plot(x,y)
        plt.title('Phase Change vs Time due to Temperature')
        plt.xlabel('Time')
        plt.ylabel('Phase Change')

    sample_rate = f*num_samples/num_periods
    (t2, adev, adev_error, adev_n) = allantools.oadev(phase, rate=sample_rate, taus='all')
    plt.loglog(t2, adev)
    plt.grid(which='both')
    plt.title('Adev Tempco Comparison; BPF vs Divider')
    plt.xlabel('Tau (sec)')
    plt.ylabel('Adev')
    plt.ylim(10**(-18), 10**(-12))
    plt.xlim(50, 2*10**(4))
    plt.grid(which='both')

    # TinyHawk Requirements
    #x2 = [1, 10, 100, 86400]
    #y2 = [3e-13, 9.5e-14, 3e-14, 1e-15]

    y3 = (2*dphase/(360*6.4e9))/t2

    #plt.loglog(x2,y2)
    plt.loglog(t2,y3)
```

```
# Comparison of Adev resulting from tempco measurements↔
```



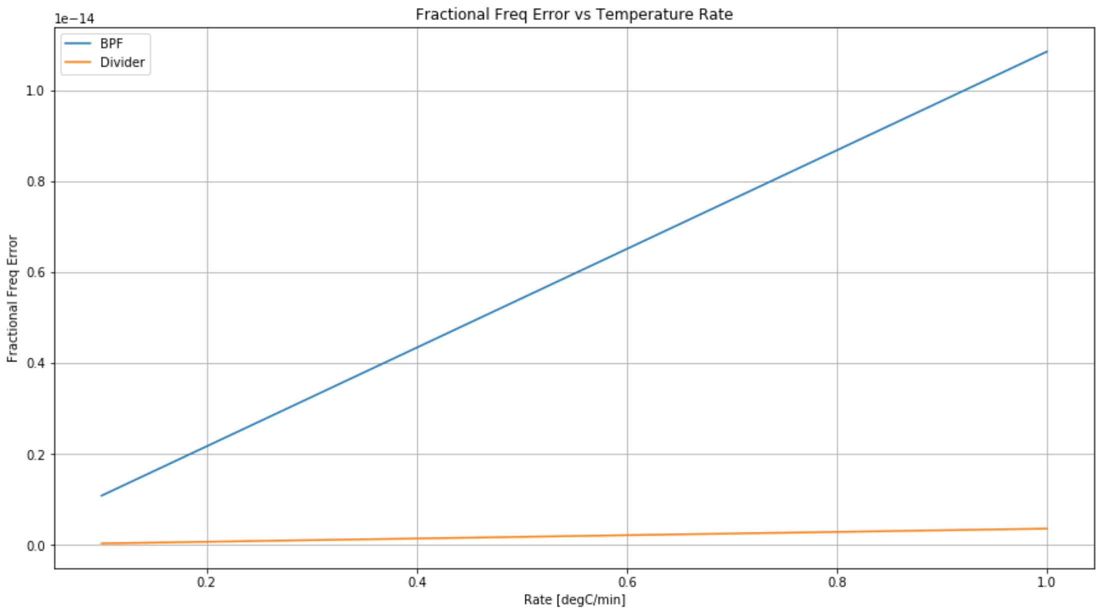
The slope of the two plots correspond to the peak to peak variation in temperature and the Tau would represent the interval at which this variation would occur.

Fractional Freq Error vs Temperature Rate

Given a temperature rate in °C/min, the corresponding fractional freq error is found.

```
# Fractional Freq Error vs Temperature Rate↔
```

```
<matplotlib.legend.Legend at 0x121e4b00>
```


Fractional Freq Error vs Temperature Rate
1e−14
BPF
Divider
Fractional Freq Error
Rate [degC/min]