Abstract

Hair cells, which detect and encode sound stimuli, have been shown to have a non-linear response and self sustaining oscillations. Here, we look at the Van der Pol oscillator, which has these properties, and numerically calculate its response to multiple driving frequencies. Any nonlinear system’s response to multiple driving frequencies will have not just the driving frequencies present, but also combination tones, or heterodyne frequencies, which have order defined by the number of driving frequency combinations. This phenomenon has been experimentally confirmed in the human auditory response [1]. In the frequency spectrum of the Van der Pol steady state response to two driving frequencies, calculated using a discrete Fourier transform, first and second order heterodyne frequencies that were significantly present were recorded. When the Van der Pol oscillator was driven at frequencies corresponding to the musical notes A₄ and C♯₅, which form a major third, no first order heterodyne frequencies, all but one second order heterodyne frequency, and at least four frequencies not associated with any first or second order heterodyne frequencies were present. When the two driving frequencies formed a perfect fifth, A₄ and E₅, all first and second order heterodyne frequencies could be observed and at least two frequencies not associated with any first or second order heterodyne frequencies were present.
Acknowledgments

I would like to thank Professor Goyal for guidance during this research experience and all the professors in the physics department for their help in my undergraduate education.
Acoustic waves incident on the human ear will drive the tympanic membrane (ear drum), which is connected to the oval window, by three tiny bones (called the ossicles). The oval window is a membrane at one end of a fluid filled canal that goes throughout a spiral shaped organ, the cochlea. The ossicles drive the oval window which will send waves propagating through the cochlear fluid. Sensory receptors in the cochlea called hair cells detect these waves and encode them into neuronal signals to be processed by the brain. Individual hair cell bundles are tuned to specific frequencies. Hair cell bundles at the start of the cochlea spiral are tuned to higher frequencies relative to bundles further down the spiral. Early experiments studying the response of dead cochleas concluded that the cochlear response is linear [2]. This model suggested the human auditory system is a passive receiver of acoustic waves. However, more advanced experiments using laser-interferometric velocimetry on a live chinchilla cochlea have shown a non-linear response [3]. A live hair cell bundle responds to a wave stimulus with self amplified oscillations, meaning the auditory system uses energy to facilitate hair cell oscillations. This model is consistent with the detection of acoustic frequencies emitted from the ear itself [4], which have been shown to be the result of limit cycle oscillations [5]. A limit cycle oscillation is when a system settles into an undamped oscillation without any external driving force.

One of the properties of non-linear differential equations is heterodyne frequency production. When forced at two different frequencies, $f_1$ and $f_2$, a nonlinear response will not only have these frequencies present but also linear combinations of integer multiples of the driving frequencies (see Table 1.1). The order of the heterodyne frequency is given by the sum of the coefficient magnitudes minus one. In the context of sound frequencies, only the absolute value of heterodyne frequencies is relevant. Generally, higher order heterodyne frequencies are present with lower intensity. In the human auditory system, usually first order and rarely second order heterodyne frequencies can be detected [1]. Since the human auditory is non-linear in its response to acoustic stimuli, the component frequencies of a sound (here, “sound” is defined as what is actually perceived in the brain of the observer, not the physical acoustic wave that caused it) will consist of the component frequencies of the acoustic wave itself and the corresponding heterodyne frequencies.
Table 1.1: Heterodyne frequencies produced from two driving frequencies: \( f_1 \) and \( f_2 \).

<table>
<thead>
<tr>
<th>Order</th>
<th>Frequencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Order</td>
<td>( 2f_1, 2f_2, f_1 + f_2,</td>
</tr>
<tr>
<td>Second Order</td>
<td>( 3f_1, 3f_2, 2f_1 + f_2,</td>
</tr>
<tr>
<td>Third Order</td>
<td>( 4f_1, 4f_2, 3f_1 + f_2,</td>
</tr>
</tbody>
</table>

A sound is heard as “musical” when its component frequencies form a harmonic series i.e. a fundamental frequency, \( f \) and its integer multiples: \( \{f, 2f, 3f, \ldots\} \). For example, the sound of a knock on a table can be broken into its component frequencies as any periodic waveform can, but they will essentially be a random collection of frequencies. Professional musicians would not agree on the musical pitch corresponding to that sound [1]. However, a sound from a guitar string or a flute will have component frequencies that very closely form a harmonic series and musicians will agree the corresponding musical pitch is the fundamental frequency of that series. It has long been known that certain frequencies played simultaneously produce pleasurable musical sounds while other combinations “clash”. It has been proposed [1] that two musical notes that go well together have the property that the component frequencies of the acoustic waves and the heterodyne frequencies between them (the component frequencies of the sound produced) form groups of closely spaced values and the values corresponding to each group approximately form a harmonic series.

Here, we study the properties of a modified Van der Pol oscillator which has been proposed as a model for the elements of the hearing organ [6], using numerical methods with Python. This is a non-linear differential equation that undergoes a limit cycle. Using a discrete Fourier transform, we analyze the frequency spectrum of the steady state response to multiple driving frequencies, and search for heterodyne frequencies.
Chapter 2

Numerical Methods

Many differential equations that arise in physics cannot be solved analytically. In this case, solutions can be analyzed with numerical methods. Given initial conditions, a finite time interval, and a differential equation in the form \( \dot{y} = f(t, y) \), numerical methods can return a solution, \( y(t) \), over the specified time interval. The price to pay for numerical solutions is discreteness. The time axis needs to be broken up into a discrete list of \( N \) time values and numerical methods will return a list of \( N \) solution values that correspond to each time value. An illustrative example of a numerical method is an integral being approximated as a Riemann sum. Ideally, like a Riemann sum, the numerical solution will converge to the exact answer as \( N \to \infty \), and the only downside to larger \( N \) is longer computation time. However, numerical solutions can sometimes be more subtle than this and errors can diverge faster than the solution converges to the real solution [7]. To analyze the error of numerical methods, they can be used on equations with known analytical solutions, and each value of the solution list can be compared to the corresponding time value plugged into the exact solution. To see the relationship between both percent error and computation time with the number of data points used in a numerical method of solving a differential equation, we used the following example:

\[
\frac{dy}{dt} + y^3 = \frac{y}{\alpha + t}, \tag{2.1}
\]

where \( \alpha \) is a constant and \( t > 0 \). This equation can be solved analytically:

\[
y(t) = \frac{\alpha + t}{\sqrt[3]{\frac{2}{3}(\alpha + t) + C}}, \tag{2.2}
\]

where \( C \) is an integration constant that can be solved with given \( \alpha \) and initial value, \( y_0 \), which were arbitrarily set to be: \( \alpha = .01, \ y_0 = 1 \). The numerical method used was the classical Runge-Kutta method (RK4) which is a component in the FORTRAN solver Isoda, which will be implemented in the next section. Given an equation for the first derivative, a list of time values, and an initial value, this method will compute \( N \) data points of the solution using the recursion relation:

\[
y_{j+1} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{2.3}
\]
where

\[
\begin{align*}
k_1 & \equiv hf(t_j, y_j), \\
k_2 & \equiv hf(t_j + \frac{h}{2}, y_j + \frac{1}{2}k_1), \\
k_3 & \equiv hf(t_j + \frac{h}{2}, y_j + \frac{1}{2}k_2), \\
k_4 & \equiv hf(t_{j+1}, y_j + k_3),
\end{align*}
\]

where \(h\) is the time step given by the time interval divided by the number of data points and \(f\) is the derivative of \(y\) in terms of \(y\) and \(t\). The continuous time interval is split into \(N\) discrete data points, \(t_j\), and used to get a list of \(y_j = y(t_j)\).

Taking the average percent error across all data points can give you an idea of how accurate your numerical method is. This analysis was performed in Python for each \(N\) from the list \([50,100,150,\ldots,2500]\) (see code below). The slope of a log-log plot of the average percent error over \(N\) (Figure 2.1) is a measurement of how error decreases as \(N\) increases.

```python
# set constants
a = .01
y0 = 1.0

# increase number of data points from 50 to 2500 in increments of 50
N = range(50, 2501, 50)
percent_error = []
for n in N:
    t = linspace(0, 3, n)
    # exact solution:
    C = (a/y0)**2 - (2.0/3)*(a**3)
y_exact = (a+t)/(((2.0/3)*(a+t)**3 + C)**.5)

    y = [y0]
    # solution using RK4:
    for j in range(len(t)-1):
        k1 = (y[j]/(a+t[j]) - y[j]**3)*h
        k2 = ((y[j]+.5*k1)/(a+(t[j]+h/2)) - (y[j]+.5*k1)**3)*h
        k3 = ((y[j]+.5*k2)/(a+(t[j]+h/2)) - (y[j]+.5*k2)**3)*h
        k4 = ((y[j]+k3)/(a+t[j+1]) - (y[j]+k3)**3)*h
        y.append(y[j]+(1./6)*(k1+2*k2+2*k3+k4))

    # calculate the percent error for each data point and average:
    err_i = []
    for j in range(len(y)):
        err_i.append(abs((y[j]-y_exact[j])/y_exact[j]))
    percent_error.append(avg(err_i))
```

8
The slope of the best fit line (dotted line), calculated with a linear regression, gives the relationship between percent error and $N$ for (2.1):

$$\% \text{ error } \propto \frac{1}{N^{4.01}}.$$  \hfill (2.4)

A similar analysis was used to determine the relationship between $N$ and the time required to compute the $y_j$ list from the RK4 method. To measure time, a variable `start` is set equal to the current time (as measured from an arbitrary reference point), then after the code to construct the $y_j$ list using the RK4 method on the following lines, a variable `end` is set equal to the current time. Taking the difference `end - start` gave a measurement of the computation time:

```python
start = time()
for j in range(len(t)-1):
    k1 = (y[j]/(a+t[j]) - y[j]**3)*h
    k2 = ((y[j]+.5*k1)/(a+(t[j]+h/2)) - (y[j]+.5*k1)**3)*h
    k3 = ((y[j]+.5*k2)/(a+(t[j]+h/2)) - (y[j]+.5*k2)**3)*h
    k4 = ((y[j]+k3)/(a+t[j+1]) - (y[j]+k3)**3)*h
    y.append(y[j]+(1./6)*(k1+2*k2+2*k3+k4))
end = time()
```

This process was repeated for every value of $N$ in the same list as before: [50,100,150,...,2500] (actually the time measurement was made 30 times and averaged for each $N$ value). A plot of computation time over $N$ shows a linear relationship (Figure 2.2).
Figure 2.2: Plot of computation time over number of data points

From the slope of the best fit line the computation time is approximately:

\[ t = (0.032 \text{ ms})N. \]  \hspace{2cm} (2.5)

The same procedure on the simpler differential equation:

\[ \frac{dy}{dt} = r y (1 - y), \]  \hspace{2cm} (2.6)

gave the following results:

\[ \% \text{ error} \propto \frac{1}{N^{3.99}}, \]  \hspace{2cm} (2.7)

\[ t = (0.019 \text{ ms})N. \]  \hspace{2cm} (2.8)

The dependance of error on N to the negative fourth power agree with what is expected from the derivation of the RK4 method [7].
Chapter 3

Numerical Analysis of Driven Oscillators

3.1 Free Oscillations

A harmonic oscillator has a spring force proportional to \( x \), the distance from equilibrium, that tends to pull towards the equilibrium point and a damping force proportional to velocity, \( \dot{x} \), that opposes motion. Newton’s second law gives a differential equation of the form:

\[
\ddot{x} + \gamma \dot{x} + \omega_0^2 x = 0,
\]

(3.1)

where \( \gamma \) and \( \omega_0 \) are positive constants. If \( \gamma < 2\omega_0 \), and the oscillator is given at least one non-zero initial position and velocity, the solution is of the form:

\[
x(t) = Ae^{-\frac{\gamma}{2} t} \cos(\omega_f t + \phi),
\]

(3.2)

where \( A \) and \( \phi \) are constants (given by the initial conditions), and

\[
\omega_f = \sqrt{\omega_0^2 - \frac{\gamma^2}{4}}.
\]

(3.3)

Usually \( \gamma \) is small enough that \( \omega_f \approx \omega_0 \).
As time increases, (3.2) goes to zero (the solutions with $\gamma \geq 2\omega_0$ also go to zero). We are interested in unforced differential equations that produce limit cycle solutions. One such equation is a modified Van der Pol equation:

$$\ddot{x} + \nu \dot{x}(x^2 - \mu) + \omega_0^2 x = 0,$$

(3.4)

where $\nu$, $\mu$, and $\omega_0$ are positive constants. The second term in (3.4) has the opposite sign of $\dot{x}$ when $|x| < \sqrt{\mu}$. When this is true, there is a force in the direction of motion i.e. energy is being input into the system. This is in contrast to the second term in the simple harmonic oscillator (3.1) which always has the sign of $\dot{x}$; energy is constantly being dissipated. There is no analytic solution to (3.4) but its properties can be observed from numerical solutions. As time increases, $x(t)$ approaches a steady oscillation. For a given set of $\nu$, $\mu$, and $\omega_0$, there is exactly one limit cycle and all initial conditions will eventually settle into this limit cycle [8]. A frequency analysis of this limit cycle (using a discrete Fourier transform) shows that much like the harmonic oscillator (3.3), the main frequency is slightly less than $\omega_0/2\pi$. This difference increases with increasing $\nu$. Odd harmonics $\{f, 3f, 5f, \ldots\}$, where $f$ is defined as the main frequency $\approx \omega_0/2\pi$, are included with decreasing intensity as the harmonics get higher (see Figure 3.3).
Figure 3.2: An example of a numerical solution to (3.4) ($\omega_0 = 5$, $\nu = 4.5$, $\mu = 1$). On the left is $x$ as function of time and on the right is the phase diagram ($v = \dot{x}$). Solutions were obtained numerically with Python (see 3.2.2).

(a) Frequency spectrum ($I$ is intensity) of the limit cycle from the numerical solution of (3.4) with $\omega_0 = 5$, $\nu = 4.5$, $\mu = 1$.

(b) Main frequency of the limit cycle from the numerical solution of (3.4) as a function of $\nu$ ($\mu = 1$, and $\omega_0 = 5$). The dashed line is $\omega_0/2\pi$. Data obtained using the methods in 3.2.2.

Figure 3.3

3.2 Forced Oscillations

3.2.1 Forced Harmonic Oscillator

Adding a sinusoidal term with angular frequency $\omega$ on the right hand side of (3.1) represents a periodic force on the system:

$$\ddot{x} + \gamma \dot{x} + \omega_0^2 x = A \sin \omega t,$$

(3.5)
where $A$ is a constant amplitude. The solution to this equation is a superposition of the homogeneous solution (3.2), and another term of the form

$$x_{ss}(t) = \frac{A}{\sqrt{(\omega_0^2 - \omega^2)^2 + \gamma^2 \omega^2}} \cos(\omega t + \phi),$$

(3.6)

where $\phi$ is a constant. Since the homogeneous solution goes to zero as time increases, the solution to (3.5) will eventually consist only of the steady state oscillation, $x_{ss}(t)$, with frequency $\omega/2\pi$ and amplitude

$$\text{Steady State Amplitude} = \frac{A}{\sqrt{(\omega_0^2 - \omega^2)^2 + \gamma^2 \omega^2}}.$$  

(3.7)

### 3.2.2 Python Code

To instill confidence in our numerical solution Python code, we solved (3.5) numerically and plotted the steady state frequency and steady state amplitude over a range of driving frequencies. Numerical solutions were obtained with the “odeint” function in Python’s scipy.integrate package which numerically solves a set of ordinary differential equations with initial conditions using the FORTRAN solver, lsoda. To solve a second order differential equation such as (3.5), define $z \equiv x$ and $g \equiv \dot{x}$ so that (3.5) becomes a set of two coupled first order differential equations:

$$\dot{g} = A \sin \omega t - \gamma g - \omega_0^2 z$$

$$\dot{z} = g.$$

To implement this solver, first define a function that gives the set of differential equations. Then “odeint” takes that function, the initial conditions, and the time axis as arguments and returns an array containing $z$ and $g$ ($x$ and $\dot{x}$) as a function of time:

```python
# define function that gives the set of differential equations (f is the # driving frequency)
def deriv(y,t):
    zprime = y[1]
    gprime = -gamma*y[1] - (omega_0**2)*y[0] + A*sin(2*pi*f*t)
    return array([zprime , gprime])

# odeint takes the function defined above, the initial conditions, and the # time axis as arguments and returns an array containing z and g (x and dx/dt) # as a function of time
y = odeint(deriv,y0,t)
x = y[:,0]
v = y[:,1]
```

The steady state frequency was computed using a function that finds the main frequency of a periodic function, $y(t)$, by taking the average value of the function, then looping through $y$ and recording every $t[j+1]$ ($j$ is the index), where $y[j] < \text{avg}(y)$ and $y[j+1] \geq \text{avg}(y)$, minus a linear interpolation term to correct for the overshoot. Then the inverse of the average difference between
adjacent $t[j]$ values gives an approximation of the main frequency of $y$. This function will work for any periodic function that passes through its average value twice in a given period (such as the steady state of a forced harmonic oscillator or limit cycle of the Van der Pol equation). The definition of this function is as follows:

```python
def getfreq(t,y):
    if len(t) == len(y): # check that time and y axes are consistent
        base = sum(y)/len(y) # get average value of function
        Q = []
        dt = t[2]-t[1] # get time step
        for j in range(len(y)-1):
            if y[j] < base and y[j+1] >= base:
                # take the first t value that is above the base value and subtract
                # a fraction of the time step that corresponds to how much greater
                # than base value y at this time point is
                Q.append(t[j+1] - ((y[j+1]-base)*dt)/(y[j+1]-y[j]))
        per = []
        for j in range(1,len(Q)):
            # list of individual periods
            per.append(Q[j]-Q[j-1])
        avg_per = sum(per)/len(per) # get average period
        return (1.0/avg_per)
    else:
        print 'Inconsistent t and y axes'
```

This function was called on the second half of the numerical solution for $x(t)$ to measure the frequency of the steady state oscillation. Steady state amplitude was computed by taking the maximum of the last sixth of the numerical solution. The Python code to calculate steady state amplitude and frequency over driving frequency was:

```python
# set constants, length of time axis, number of data points, and initial conditions
omega_0 = 2*pi*440.
gamma = 400.
t_max = 40.
N = 600000
t = linspace(0,t_max,N)
y0 = [0.,0.] # [initial position, initial velocity]

# driving frequencies range from 100 to 770 Hz with 200 linearly spaced data points
freq = linspace(100,770,200)
ss_freq = []
ss_amp = []
for f in freq:
    # solve the differential equation for each frequency, f
    def deriv(y,t):
        zprime = y[1]
```
The steady state frequency and amplitude from the numerical solution matched the expected values from the analytical solutions with very low error (see Figure 3.4). The average percent error was 0.031% and $4.8 \cdot 10^{-7}$% for steady state amplitude and frequency respectively.

3.2.3 Two Frequency Forcing

If a harmonic oscillator is forced with two different frequencies:

$$\ddot{x} + \gamma \dot{x} + \omega_0^2 x = A_1 \sin \omega_1 t + A_2 \sin \omega_2 t,$$

Figure 3.4: Steady state amplitude (top) with percent error to its right and steady state frequency (bottom) with percent error to its right from the numerical solution of (3.5) with $\omega_0 = 880\pi$, $\gamma = 400$, and $A = 1$. 

3.2.3 Two Frequency Forcing

If a harmonic oscillator is forced with two different frequencies:

$$\ddot{x} + \gamma \dot{x} + \omega_0^2 x = A_1 \sin \omega_1 t + A_2 \sin \omega_2 t,$$
the solution will be a superposition of the unforced solution and two terms like (3.6) corresponding to each driving frequency and amplitude. The steady state will consist only of the undamped terms:

$$x_{ss}(t) = \frac{A_1}{\sqrt{(\omega_0^2 - \omega_1^2)^2 + \gamma^2 \omega_1^4}} \cos(\omega_1 t + \phi_1) + \frac{A_2}{\sqrt{(\omega_0^2 - \omega_2^2)^2 + \gamma^2 \omega_2^4}} \cos(\omega_2 t + \phi_2), \quad (3.9)$$

and the only frequencies present will be $\omega_1/2\pi$ and $\omega_2/2\pi$.

When driving the Van der Pol oscillator at two different frequencies,

$$\ddot{x} + \nu \dot{x}(x^2 - \mu) + \omega_0^2 x = A_1 \sin \omega_1 t + A_2 \sin \omega_2 t, \quad (3.10)$$

we expect the frequency spectrum of the steady state to be more complex. The coupled differential equations to input into the “odeint” function become

$$\dot{g} = -\nu g(z^2 - \mu) - \omega_0^2 z + A_1 \sin \omega_1 t + A_2 \sin \omega_2 t$$

$$\dot{z} = g,$$

where $z = x$ and $g = \dot{x}$, as before. The code to generate $x(t)$ from (3.10) is

```python
def deriv(y,t):
    zprime = y[1]
    gprime = -nu*y[1]*(y[0]**2 - mu) - (omeg**2)*y[0]+A1*sin(af1*t)+A2*sin(af2*t)
    return array([zprime,gprime])
```

t_max = 4.
N = 600000
t = linspace(0,t_max,N)
y0 = [0.,0.]
y = odeint(deriv,y0,t)
x = y[:,0]
v = y[:,1]

where “af1” and “af2” are $\omega_1$ and $\omega_2$, “A1” and “A2” are $A_1$ and $A_2$, and “omeg” is $\omega_0$.

The Van der Pol oscillator parameters were set to $\omega_0 = 2\pi \cdot 440$, $\nu = 450$, and $\mu = 1$. With these parameters, the main frequency of free oscillation, calculated with the function “getfreq” described in 3.2.2, was 439.27 Hz. This system driven at 440 and 554.365 Hz (corresponding to the musical notes A_4 and C#_5 respectively which form a major third interval) is shown in Figure 3.5. The driving amplitudes, $A_1$ and $A_2$, were both $5 \cdot 10^6$. In the frequency spectrum of the steady state (Figure 3.6), which was taken to start at $t = 0.3$ s, the two most prominent frequencies are the two driving frequencies. No first order heterodyne frequencies are seen. The second order heterodyne frequencies 325.63, 1320, 1434.37 Hz are, in order from more to less intense, the next most prominent frequencies. Other frequencies, not corresponding to first or second order heterodyne frequencies can be seen, and two other second order heterodyne frequencies, 668.73 and 1548.73 Hz are present.
Figure 3.5: Van der Pol oscillator ($\omega_0 = 2\pi \cdot 440$, $\nu = 450$, and $\mu = 1$) response driven at two frequencies that form a major third: A$_4$ and C#$_5$.

Figure 3.6: Frequency spectrum of the Van der Pol oscillator ($\omega_0 = 2\pi \cdot 440$, $\nu = 450$, and $\mu = 1$) steady state response driven at two frequencies that form a major third: A$_4$ and C#$_5$. The right figure is the spectrum zoomed in on with grey dotted lines showing first and second heterodyne frequencies and red dotted lines showing the driving frequencies.

The Van der Pol oscillator response (with same parameters) driven at two frequencies that form a perfect fifth: 440 and 659.26 Hz (A$_4$ and E$_5$) is shown in Figure 3.7. Again, the driving amplitudes, $A_1$ and $A_2$, were both $5 \cdot 10^6$. In the frequency spectrum of the steady state (Figure 3.8), taken to start at $t = 0.3$ s, the two most prominent frequencies are again the two driving frequencies. All first and second order heterodyne frequencies are present though the two highest second order frequencies are barely noticeable. Two more frequencies higher than that are also present. The results from both intervals are summarized in Table 3.1.
Figure 3.7: Van der Pol oscillator ($\omega_0 = 2\pi \cdot 440$, $\nu = 450$, and $\mu = 1$) response driven at two frequencies that form a major fifth: $A_4$ and $E_5$.

Figure 3.8: Frequency spectrum of the Van der Pol oscillator ($\omega_0 = 2\pi \cdot 440$, $\nu = 450$, and $\mu = 1$) steady state response driven at two frequencies that form a major fifth: $A_4$ and $E_5$. The right figure is the spectrum zoomed in on with grey dotted lines showing first and second order heterodyne frequencies and red dotted lines showing the driving frequencies.

Table 3.1: Summary of results from two frequency driving of the Van der Pol oscillator (3.10). All first and second order heterodyne frequencies are listed and any frequency found in the frequency spectrum are in bold. The last column is any significant frequency found that isn’t associated with a heterodyne frequency. All frequencies are in Hz.

<table>
<thead>
<tr>
<th>Musical Notes</th>
<th>Driving Frequencies</th>
<th>First Order Heterodyne Frequencies</th>
<th>Second Order Heterodyne Frequencies</th>
<th>Other Frequencies Present</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_4 &amp; C#_5 (Major Third)</td>
<td>440.0 &amp; 554.37</td>
<td>114.37, 880.0, 994.37, 1108.73</td>
<td>325.63, 668.73, 1320.0, 1434.37, 1548.73</td>
<td>211.35, 1205.68</td>
</tr>
<tr>
<td>A_4 &amp; E_5 (Perfect fifth)</td>
<td>440.0 &amp; 659.26</td>
<td>219.26, 880.0, 1099.26, 1318.51</td>
<td>220.74, 878.51, 1320.0, 1539.26, 1758.51</td>
<td>2200.0, 2419.2, (1977.77)</td>
</tr>
</tbody>
</table>
Chapter 4

Conclusion

Hair cells, the sensory receptors in the inner ear that detect and encode sound stimuli, have a non-linear response and undergo self sustained oscillations. We numerically solved the Van der Pol equation, which has these properties, with the “odeint” function in Python’s scipy.integrate package. The frequency spectrum, calculated using a discrete Fourier transform, of the steady state while driven at two different frequencies was observed to record which first and second order heterodyne frequencies were present. When the Van der Pol oscillator ($\omega_0 = 2\pi \cdot 440$, $\nu = 450$, and $\mu = 1$) was driven at frequencies A$^4$ (440 Hz) and C#$^5$, which form a major third, no first order heterodyne frequencies, all but one second order heterodyne frequency, and at least four frequencies not associated with any first or second order heterodyne frequencies were present. When the two driving frequencies formed a perfect fifth, A$^4$ and E$^5$, all first and second order heterodyne frequencies could be observed and at least two frequencies not associated with any first or second order heterodyne frequencies were present.

The selection of the Van der Pol oscillator was quite arbitrary and it was not at all apparent what the parameters, $\mu$ and $\nu$, should be. In future studies, it would be better to start from a differential equation that represents a physical model of the hair cell bundle response.
Bibliography


