STAT 31410: Homework 1

Caleb Derrickson

October 8, 2023

Collaborators: The TA's of the class, as well as Kevin Hefner, Nathan Suhr, and Steven Lee.

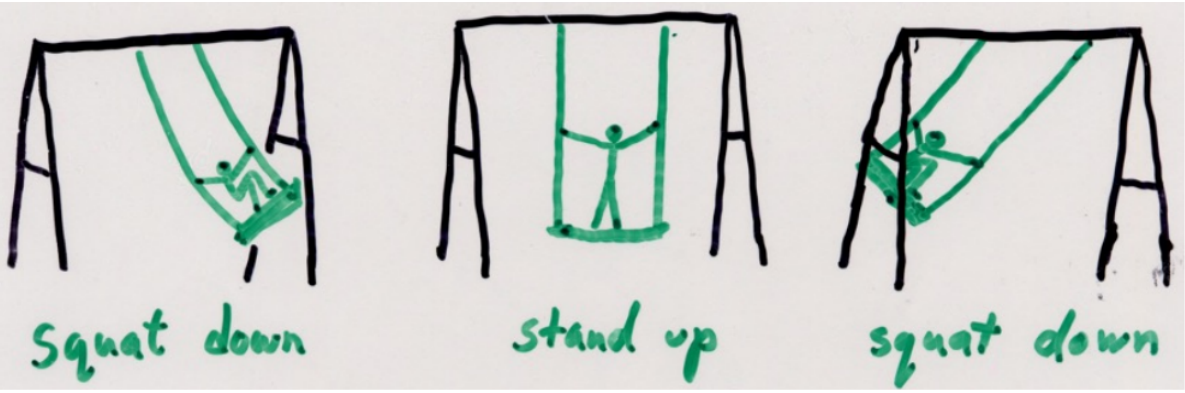
For this problem set, we consider the equation for an idealized pendulum

$$\frac{d^2\theta}{dt^2} = -\frac{g}{\ell(t)} sin(\theta), \tag{1}$$

where the length of the moment arm, l, varies periodically according to

$$\ell(t) = \ell_0(1 + \varepsilon \cos(\omega t)), \quad \varepsilon \ll 1. \tag{2}$$

This might be a good model of a child on a swing, trying to pump energy into it by repeatedly standing up and then squatting back down with some appropriate rhythm, to be determined.



1. Make an approximation where you keep only the leading order term in ε, i.e linearize about $\varepsilon = 0$. Then choose a non-dimensionalization of time that puts (1) into this form:

$$\ddot{\theta} = -(\alpha + \beta \cos(\tau)) \sin(\theta), \tag{3}$$

where you define the dimensionless parameters α and β in terms of the original parameters g, ℓ_0, ε and ω. Re-write the second order equation 3 as two first order equations for the angle θ and the angular speed

$\Omega \equiv \dot{\theta}$. For $\beta = 0$, derive a conserved quantity, the energy of the pendulum, denoted $E(\theta, \Omega)$. Write an equation for $\dot{E}$ that applies if $\beta \neq 0$.

Solution:

Since we are taking $\varepsilon \ll 1$ from 1, we can define a function $f(\varepsilon)$ on which we will expand using ordinary Taylor expansion. Here, I will define $f(\varepsilon)$ as:

$$f(\varepsilon) = \frac{1}{1 + \varepsilon \cos(\omega t)}, \tag{4}$$

where it is understood that $\cos(\omega t)$ is a constant in this sense. Note that we are *linearizing* with respect to θ, so we can drop all terms of higher order. Thus,

$$\begin{aligned} f(\varepsilon) &= \sum_{n=0}^{\infty} \varepsilon^n \frac{d^n f}{d\varepsilon^n}\bigg|_{\varepsilon=0} \\ &= f(\varepsilon = 0) + \varepsilon \frac{df}{d\varepsilon}\bigg|_{\varepsilon=0} + O(n^2) \\ &= 1 - \varepsilon \left[\frac{\cos(\omega t)}{(1 + \varepsilon \cos(\omega t))^2} \right]_{\varepsilon=0} \\ &= 1 - \varepsilon \cos(\omega t) \end{aligned} \tag{5}$$

We can then plug 5 into 1 to obtain,

$$\ddot{\theta} = -\frac{g}{\ell_0}\Big(1 - \varepsilon \cos(\omega t)\Big) \sin(\theta). \tag{6}$$

From 6, we can define three dimensionless parameters: τ, α, β. τ will take the place of the argument inside the cosine term of 6, while α and β will clean up any extraneous terms. We will first redefine the ratio of g and ℓ_0 as the resonance frequency of our state, ω_0.

$$\omega_0^2 = \frac{\ell_0}{g}.$$

Then we will take τ as $\tau = \omega t$. Note that we are taking time derivatives, so this re-parameterization affects what we are differentiating by.

$$\tau = \omega t \implies d\tau = \omega dt \iff \frac{1}{\omega}\frac{d}{d\tau} = \frac{d}{dt}.$$

Applying these to 6, we get,

$$\ddot{\theta} = -\left(\frac{\omega_0}{\omega}\right)^2 \big(1 - \varepsilon\cos(\tau)\big)\sin(\theta). \tag{7}$$

We can then define α, β as,

$$\alpha = \left(\frac{\omega_0}{\omega}\right)^2, \qquad \beta = \alpha\varepsilon \tag{8}$$

Note that, as suggested by the TA's, to take β as a positive constant. The negative can be recovered by shifting τ by π, meaning $\tau = \omega t + \pi$. This shift will change nothing else in our analysis, hopefully. Also, when taking time derivates for the rest of this homework, I am specifically referring to taking the τ derivative. Thus, our second order differential equation is,

$$\ddot{\theta} = -\big(\alpha + \beta\cos(\tau)\big)\sin(\theta), \tag{9}$$

matching 3.

Our next goal is to re-write 9 as two first order equations for θ and the angular speed $\Omega \equiv \dot{\theta}$. This can be achieved quite easily - our system is now.

$$\begin{cases} \dot{\theta} = \Omega, \\ \dot{\Omega} = -\big(\alpha + \beta\cos(\tau)\big)\sin(\theta) \end{cases} \tag{10}$$

Our next goal is to find a conserved quantity for $\beta = 0$. This quantity will be the energy of our system (or rather, an approximation of it) and will be denoted as E. From the lectures, our energy is the sum of kinetic and potential energies, in just the θ dimension. Potential can be found by integrating the (negative) right-hand-side of 9 by our spacial parameter.

$$\begin{aligned} \implies V(\theta) &= \int \alpha\sin(\theta)\, d\theta \\ &= -\big(\alpha + \beta\cos(\tau)\big)\cos(\theta) \end{aligned} \tag{11}$$

The energy is now,

$$E(\theta, \Omega) = \frac{1}{2}\dot{\theta}^2 + V(\theta),$$

$$E(\theta, \Omega) = \frac{1}{2}\Omega^2 - \alpha\cos(\theta). \tag{12}$$

Where the instantaneous change in energy (with respect to time) is,

$$\begin{aligned}
\dot{E}(\theta, \Omega) &= \frac{d}{d\tau}\left[\frac{1}{2}\Omega^2 - \alpha\cos(\theta)\right] \\
&= \Omega\dot{\Omega} - \alpha\frac{d}{d\tau}\left[\cos(\theta)\right] \\
&= -\Omega\big(\alpha + \beta\cos(\tau)\big)\sin(\theta) + \alpha\sin(\theta)\dot{\theta} \\
&= -\beta\Omega\cos(\tau)\sin(\theta)
\end{aligned} \tag{13}$$

For the child in the swing picture, is the energy E (instantaneously) increasing or decreasing or neither in each of the frames shown? Try making a rough sketch of the time course of each of these quantities: $\theta(t), \Omega(t), \ell(t)$ and $E(t)$, assuming a periodic motion of the swing as well as the swinger. What is the period of the swing motion compared to the period of $\ell(t)$? Are they the same?

Solution: As a means of faithfully showing the plots, I used matplotlib to create them. Note I am not showing the numerical values of the functions, since they are not enlightening. Note the function $\ell(t)$ is bounded by $1 \pm \varepsilon$. My code is shown below, followed by the generated plots.

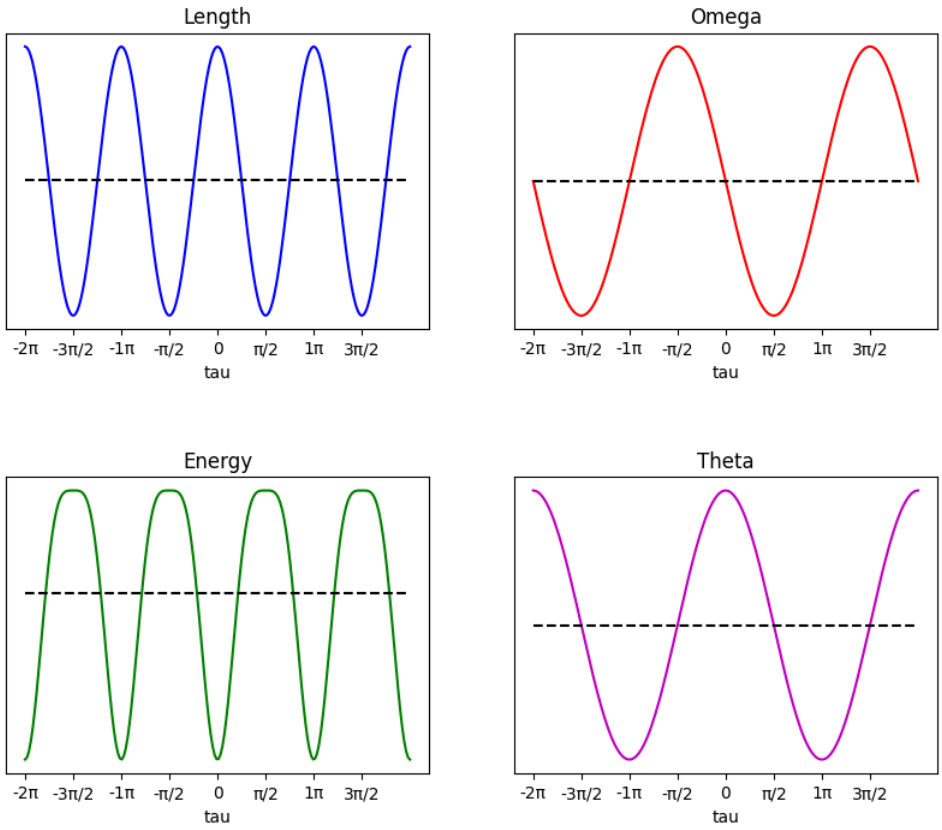
Note that these are here purely as a heuristic. As we expect the length of the pendulum arm to be the greatest while the child is squatting down, and minimal when the child to be standing up. Squatting down should correlate to maximum values of theta, and standing up should correlate to minimum values of theta. We can see this represented in the plots of length and theta. Since $\dot{\theta} = \Omega$, we should see maxed values of ω when $\theta = 0$, which is shown. Energy is slightly different. When $E = 0$, the kinetic energy is equal to its potential, which happens going to and coming from max values of θ. It's maximal values should then be when either the kinetic or potential energies are maximized, lining up with values $\theta = -\pi/2,\ 0,\ \pi/2$. Regardless, in an entire swing interval, when going from $-\pi/2$ to $\pi/2$, we should hit the zeros of E four times, indicating that the energy runs with frequency four times that of θ, twice the frequency of ℓ. Though this seems to not be represented in the plot of energy, I expect my rationality to

be a better interpretation.

From our calculation of $\dot{E}$, we can inspect the change of the energy in the given frames of the system. Note that we are to expect the system to achieve its maximum heights in the first and third frames. In the first frame, we can argue the change in energy is zero, as shown by 13, Ω should be zero. This is also seen in the third frame. The second frame shows the child at $\theta = 0$, thus $\sin(\theta) = 0$, making our $\dot{E}$ also zero.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline

def ell(tau, ep):
    return (1 + ep * np.cos(2 * tau))
def Omega(tau, ep):
    return -1*(np.sin(tau))
def energy(tau, ep):
    return 0.5 * Omega(tau, ep) **2 - np.cos(theta(tau))
def theta(tau):
    return np.cos(tau)

ep = 0.1
tau = np.linspace(-np.pi * 2, np.pi * 2, 200)
tau_ticks = np.arange(-np.pi * 2, np.pi * 2, np.pi/2)
tau_ticks_label = [f'0' if n == 0 else f'-pi/2' if n == -1 else f'pi/2' if n ==
    1 else f'{int(n/2)}pi' if n%2 == 0 else f'{int(n)}pi/2' for n in tau_ticks
    / (np.pi / 2)]

plt.figure(figsize=(10, 8))
plt.subplots_adjust(hspace=0.5)

plt.subplot(2, 2, 1)
plt.plot(tau, ell(tau, ep), 'b-', label = "Length")
plt.plot(tau, [np.mean(ell(tau, ep)) for _ in range(len(tau))], 'k--')
plt.xlabel("tau")
plt.title("Length")
plt.yticks([])
plt.xticks(tau_ticks, tau_ticks_label)

plt.subplot(2, 2, 2)
plt.plot(tau, Omega(tau, ep), 'r-', label = "Omega")
```

```
plt.plot(tau, 0 * tau, 'k--')
plt.xlabel("tau")
plt.title("Omega")
plt.yticks([])
plt.xticks(tau_ticks, tau_ticks_label)

plt.subplot(2, 2, 3)
plt.plot(tau, energy(tau, ep), 'g-', label = "Energy")
plt.plot(tau, [np.mean(energy(tau, ep)) for _ in range(len(tau))], 'k--')
plt.xlabel("tau")
plt.title("Energy")
plt.yticks([])
plt.xticks(tau_ticks, tau_ticks_label)

plt.subplot(2, 2, 4)
plt.plot(tau, theta(tau), 'm-', label = "Theta")
plt.plot(tau, 0 * tau, 'k--')
plt.xlabel("tau")
plt.title("Theta")
plt.yticks([])
plt.xticks(tau_ticks, tau_ticks_label)

plt.show()
```



2. Linearize your system of first order equations about the equilibrium position $\theta = \Omega = 0$ to obtain a linear non-autonomous system to study:

$$\frac{d\mathbf{x}}{d\tau} = A(\tau)\mathbf{x}, \tag{14}$$

where $A(\tau) = A(\tau + T)$ is a 2×2 T-periodic matrix, and $\mathbf{x} \in \mathbb{R}^2$ is your state space vector. What is T for your non-dimensional problem?

Solution:

In preparation for this, we wrote out our "pseudo system" as 10. From this, we are only interested in small perturbations of θ and Ω around $\theta = \Omega = 0$. Thus, we can replace functions dependent on θ and Ω with their Taylor Expanded equivalent, up to linear term. From 10, the only function is $\sin(\theta)$, which can be approximated by $\sin(\theta) \approx \theta$ under our assumptions. Therefore we have,

$$\begin{pmatrix} \dot{\theta} \\ \dot{\Omega} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -(\alpha + \beta\cos(\tau)) & 0 \end{pmatrix} \begin{pmatrix} \theta \\ \Omega \end{pmatrix} \tag{15}$$

It is quite easy to see that our state vector, $\mathbf{x}$ is just $(\theta, \Omega)^t \in \mathbb{R}^2$, where t denotes the transpose. It is also noted that our system $A(\tau)$ is periodic, with period $T = 2\pi$.

Briefly summarize what Floquet theory tells us about the form of the solutions of 15.

Solution:

Floquet theory tells us that there exists some periodicity in our solutions. Since our system is periodic (with period $T = 2\pi$), the general form of the solutions is given by theorem 2.36 in the course book.

$$\Phi(t, 0) = \mathscr{P}(t)e^{tB}, \tag{16}$$

where $\mathscr{P}$ is a T-periodic matrix and $B = \frac{1}{T}\ln M$, where M is the monodromy matrix: $M \equiv \Phi(T, 0)$.

3. Let $\Phi(\tau)$ be the fundamental matrix solution associated with the linear problem 15 that satisfies $\Phi(0) = \mathbb{I}$, the 2×2 identity matrix. Derive and then solve a first order ordinary differential equation for $det(\Phi)$.

Solution:

This is done in the book as the proof of Abel's formula, so I will transcribe the proof in my own words.

First we re-write det(Φ) into an equivalent form,

$$\det(\Phi) = \sum_{j=1}^{n} c_{ij}\varphi_{ij}, \tag{17}$$

where c_{ij} is the co-factor of the i, j-th entry of Φ. Note that 17 can be taken over any row of Φ, thus if we decouple the respective indices row entries of φ_{ij} and c_{ij}, we get,

$$\det(\Phi)\ \delta_{ik} = \sum_{j=1}^{n} c_{ij}\varphi_{kj}, \tag{18}$$

where δ_{ik} is the Kronnecker delta of i and k. Furthermore, since the only term in 17 which depends on φ_{ij} is exactly the term $c_{ij}\varphi_{ij}$, we can write,

$$\frac{\partial}{\partial \varphi_{ij}} \det(\Phi) = c_{ij}. \tag{19}$$

Therefore, using the chain rule and 19, we can write the time derivative of $\det(\Phi)$ as,

$$\frac{d}{dt}\det(\Phi(t)) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}(t)\frac{d}{dt}\varphi_{ij}.$$

Note that the fundamental matrix has columns of the linearly independent solutions of 15, thus

$$\frac{d}{dt}\varphi_{ij}(t) = A(t)\Phi(t) = \sum_{k=1}^{n} a_{ik}\varphi_{kj}(t)$$

This can then be substituted into our ODE, making it,

$$\frac{d}{dt}\det(\Phi(t)) = \sum_{i,j,k}^{n} c_{ij}(t)a_{ik}(t)\varphi_{kj}(t),$$

where I have compressed the triple sum for legibility. Rearranging these terms and noting that a_ik does not depend on the j summation implies,

$$\frac{d}{dt}\det(\Phi(t)) = \sum_{i,k=1}^{n} a_{ik}(t)\sum_{j=1}^{n} c_{ij}(t)\varphi_{kj}(t).$$

The inner summation is equivalent to 18, thus,

$$\frac{d}{dt}\det(\Phi(t)) = \sum_{i,k=1}^{n} a_{ik}(t)\delta_{ik}\det(\Phi).$$

$\det(\Phi)$ can be taken out of the summation, and the Kronnecker delta collapses the summation to the case where $i = k$, so,

$$\frac{d}{dt}\det(\Phi(t)) = \det(\Phi)\sum_{i=1}^{n} a_{ii}(t).$$

This summation is exactly the trace of A, then our ODE is exactly

$$\frac{d}{dt}\det(\Phi(t)) = \det(\Phi)\mathrm{tr}(A).$$

Treating this as a separable ODE, and nothing the trace of a matrix is a scalar, we get,

$$\begin{aligned}
&\frac{d(\det(\Phi))}{\det(\Phi)} = \mathrm{tr}(A(t))dt, \\
\iff &\int \frac{d(\det(\Phi))}{\det(\Phi)} = \int_{t_0}^{t} \mathrm{tr}(A(s))ds, \\
\iff &\det(\Phi) = \exp\left[\int_{t_0}^{t} \mathrm{tr}(A(s))ds\right] \qquad (20)
\end{aligned}$$

This is the result of Abel's theorem, completing the proof.

What can you say, based on this, about the eigenvalues μ of the monodromy matrix $M \equiv \Phi(T, 0)$?

Solution:

By our Linear system, $A(s) = 0$, so the determinant of our fundamental matrix, by 20, is equal to $e^0 = 1$ for any time t. Thus, for $t = 0$, the monodromy matrix has determinant 1. This then breaks down into two cases, depending if the eigenvalues are real or complex.

In the real case, this means that μ_1 and μ_2 are inverses of each other, i.e. $\mu_1 = \frac{1}{\mu_2}$. As explained briefly in the lectures, this solution will lead to instability in our solutions.

In the complex case, this means $\mu_1 = \overline{\mu_2}$, i.e. they are complex conjugates of each other. This will mean our solutions are stable, leading to some oscillation.

If we add some linear damping to the problem so that 3 becomes

$$\ddot{\theta} = -\Big(\alpha + \beta\cos(\tau)\Big)\sin(\theta) - \gamma\dot{\theta} \quad \gamma > 0, \qquad (21)$$

What can you say about the determinant of the Monodromy matrix associated with the linearized problem in that case?

Solution:

Under the approximation we employed to derive 15, we'll take $\sin(\theta) \approx \theta$. Also, we will decompose this second order ODE into two first order by the substitution $\Omega = \dot{\theta}$, $\dot{\Omega} = -(\alpha + \beta\cos(\tau))\theta - \gamma\Omega$. Thus our system is now.

$$\begin{pmatrix} \dot{\theta} \\ \dot{\Omega} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -(\alpha + \beta\cos(\tau)) & -\gamma \end{pmatrix} \begin{pmatrix} \theta \\ \Omega \end{pmatrix} \tag{22}$$

Thus, by 20,

$$\begin{aligned} \det(\mathrm{M}) = \det(\Phi(T,0)) &= \exp\left[-\int_0^T \gamma \mathrm{ds} \right] \\ &= e^{-\gamma T} \end{aligned} \tag{23}$$

Very briefly discuss what type of stability claims, about the equilibrium $\theta = \Omega = 0$, can be made from an analysis of the solutions of 14 in the case of $\gamma = 0$ and the case of $\gamma > 0$. Recall that there are different types of "stability", e.g. Lyapunov stability, asymptotic stability, linear stability, as well as instability. Later we will be looking at some results that tell us when it is enough to examine the linearized problem to determine the stability properties of an equilibrium of the nonlinear problem. We will also be interested in cases where that is not enough.

Solution:

To start off my brief discussion, I will summarize these two classes of stability. Lyapunov stability claims that there exists a neighborhood centered at x^* such that x^* is a zero of of 14 within which our solution resides for any t. Asymptotic stability takes the Lyapunov stability and adds the condition that a solution will tend to x^* as $t \to \infty$. So or the case of $\gamma = 0$, our system will be only Lyapunov stable if the Floquet multipliers are complex. If they are real, the system will only have unstable solutions. When we take $\gamma > 0$, the real case stays the same, but the complex case will graduate to asymptotic stability. This are due to our calculation above.

---

4. The remainder of this problem involves numerical computations for the linearized problem 14. On an appropriate grid of parameter values (α, β) numerically compute the Monodromy matrix and its eigenvalues in order to estimate the boundary separating parameter regions where solutions are bounded vs. unbounded. Can we assume α and/or β is positive?

   You should find "resonance tongues" in the (α, β) parameter plane that correspond to parameters where solutions of the linear problem grow exponentially. Along the way, check how well your numerical results

match the analytic ones on det(M) from part 3.

Solution:

As a quick mention. We specifically chose both α and β to be positive when we made our shift in τ. The code for generating the boundaries of bounded and unbounded solutions. I had the help of Steven Lee in generating these plots. The code to generate the plots are shown first.

We can see the "resonance tongues", boundaries of the unbounded and bounded regions in 4. We can also investigate the numerical accuracy of the products of the eigenvalues. This is shown in 4. For both unbounded and bounded solutions, we can see product of the eigenvalues to be within 1e-6 of 1.

```
#Function to model the system
def system(Y, t, a, b):
    x, y, phi11, phi12, phi21, phi22 = Y

    dxdt = y
    dydt = -(a + b * np.cos(t)) * x
    dphi11dt = phi12
    dphi12dt = -(a + b * np.cos(t)) * phi11
    dphi21dt = phi22
    dphi22dt = -(a + b * np.cos(t)) * phi21

    return [dxdt, dydt, dphi11dt, dphi12dt, dphi21dt, dphi22dt]

#Calculates the monodromy matrix from given alpha and beta
def calcualte_monodromy_matrix(alpha, beta):
    #Setting the period
    T = 2 * np.pi
    time = np.linspace(0, T, 1000)
    Y0 = [0, 1, 1, 0, 0, 1] # initial conditions: [x0, y0, phi11, phi12, phi21, phi22]

    sol = odeint(system, Y0, time, args=(alpha, beta))

    M = np.array([[sol[-1, 2], sol[-1, 3]],
                  [sol[-1, 4], sol[-1, 5]]])
    return M

#Set up the values for alpha
alpha_span = np.linspace(0, 4, 1000)
beta_span = np.linspace(0, 4, 1000)
```

```
#Arrays to categorize solutions by choice of beta, alpha
bounded_solns = []
unbounded_solns = []
unbounded_eigenvals = []
bounded_eigenvals = []

for alpha in alpha_span:
    for beta in beta_span:
        # Calculate mono matrix
        monodromy = calcualte_monodromy_matrix(alpha, beta)

        #Extracting eigenvalues
        eigenvalues = np.linalg.eigvals(monodromy)

        #saving off eigenvalues
        eigenvals.append(eigenvalues)

        #Checking magnitudes to see distance from 1.0
        magnitudes = np.abs(eigenvalues)

        if all(np.isclose(magnitudes, 1.0, atol=1e-3)):
            bounded_solns.append((alpha, beta))
            bounded_eigenvals.append(magnitudes)
        else:
            unbounded_solns.append((alpha, beta))
            unbounded_eigenvals.append(magnitudes)

bounded_solns = np.array(bounded_solns)
unbounded_solns = np.array(unbounded_solns)

plt.scatter(bounded_solns[:, 0], bounded_solns[:, 1], color='blue', label='Bounded')
plt.scatter(unbounded_solns[:, 0], unbounded_solns[:, 1], color='red',
    label='Unbounded')
plt.xlabel('alpha')
plt.ylabel('beta')
plt.legend()
plt.title('Regions of Bounded vs. Unbounded Solutions')
plt.show()
```

```
#Code to generate the numerical distance of Det(M) from 1
unbounded_mag = [e1*e2 - 1 for e1, e2 in unbounded_eigenvals]
```

```
bounded_mag = [e1*e2 - 1 for e1, e2 in bounded_eigenvals]
fig, axs = plt.subplots(2)
fig.suptitle('Eigenvalue Product Distances from 1')
axs[0].plot(np.abs(unbounded_mag), label = "unbounded")
axs[1].plot(np.abs(bounded_mag), label = "bounded")
```

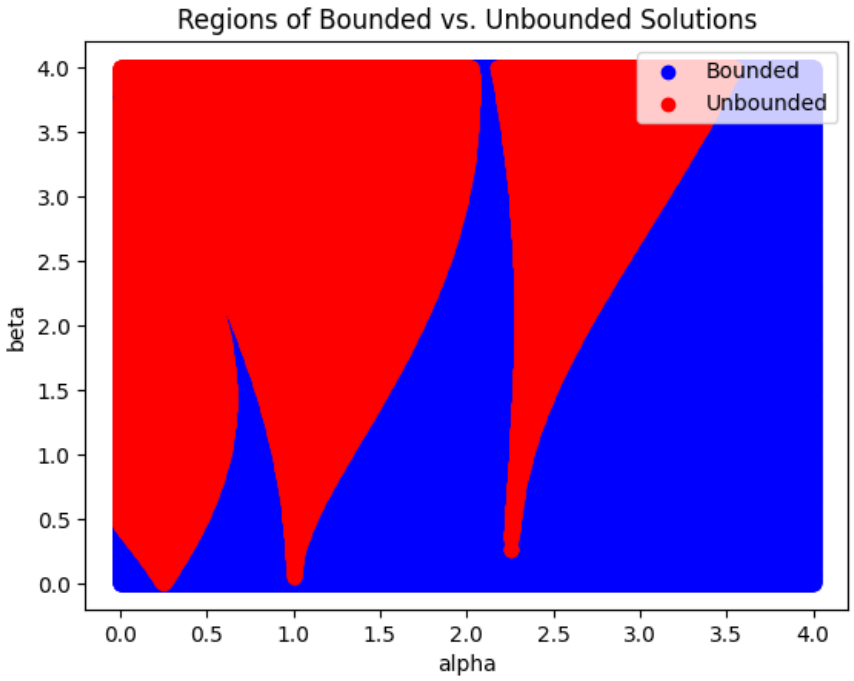


Figure 1: ”Resonance Tongues”

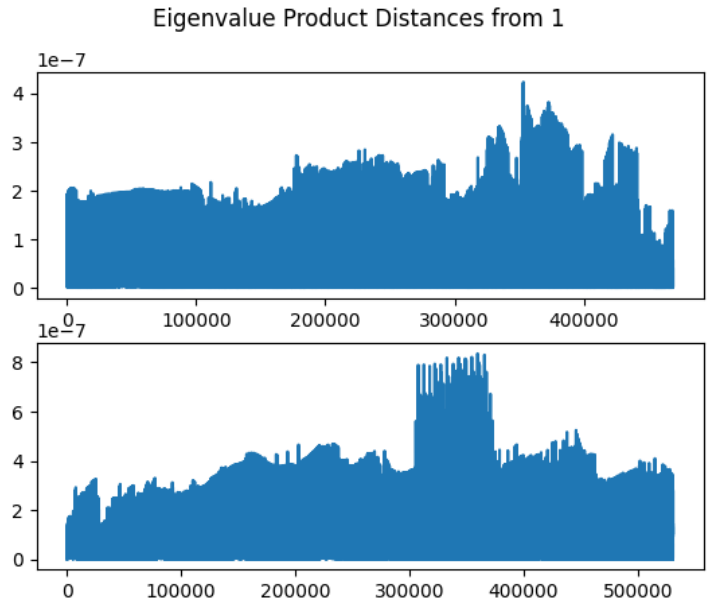


Figure 2: Above is unbounded eigenvalues, below is bounded eigenvalues.

For $\beta = 0.1$, plot the real and imaginary parts of the Floquet multipliers as a function of α; make sure to choose a large enough range of α values to capture the first 3 resonance tongues. (You should find that they are associated with particular values of α; for example the first resonance tongue is associated with $\alpha = 0.25$.) What are the Floquet multipliers on the boundary between the stable and unstable regions, for each of the tongues? What can you say about the period of the solutions on the boundaries?

Solution:

I modified the code for generating the "Resonance tongues" plot to instead just capture the real and imaginary parts of the Floquet multipliers for α up to 2. Setting $\beta = 0.4$, we see not only the first resonance tongue appear for $\alpha = 0.25$, but also the second resonance tongue at $\alpha = 1.0$. I could not capture the third resonance tongue, however, due to keeping a constant beta. The code to generate this is shown below, followed by the resulting plots.

Interestingly enough, the imaginary part of the Floquet multipliers collapse along the boundary of the bounded and unbounded solutions, while the real part take on an elliptic shape. It seems that, as we increase the value of β, we are getting increasingly higher "slices" of 4. I modified the targeted value of β to capture the second tongue. This characteristic does not hold for higher values of α, however, but we can see that the period of these elliptic shapes only increases, requiring higher values of β to capture.

```
#Function to model the system
def system(Y, t, a, b):
    x, y, phi11, phi12, phi21, phi22 = Y

    dxdt = y
    dydt = -(a + b * np.cos(t)) * x
    dphi11dt = phi12
    dphi12dt = -(a + b * np.cos(t)) * phi11
    dphi21dt = phi22
    dphi22dt = -(a + b * np.cos(t)) * phi21

    return [dxdt, dydt, dphi11dt, dphi12dt, dphi21dt, dphi22dt]


#Calculates the monodromy matrix from given alpha and beta
def calcualte_monodromy_matrix(alpha, beta):
    #Setting the period
    T = 2 * np.pi
    time = np.linspace(0, T, 100)
```

```
    Y0 = [0, 1, 1, 0, 0, 1] # initial conditions: [x0, y0, phi11, phi12, phi21, phi22]

    sol = odeint(system, Y0, time, args=(alpha, beta))

    M = np.array([[sol[-1, 2], sol[-1, 3]],
                  [sol[-1, 4], sol[-1, 5]]])
    return M


#Set up the values for alpha
alpha_span = np.linspace(0, 2, 500)
beta = 0.4
real_eigenvals = []
imag_eigenvals = []

for alpha in alpha_span:
    # Calculate mono matrix
    monodromy = calcualte_monodromy_matrix(alpha, beta)

    #Extracting eigenvalues

    eigenvalues = np.linalg.eigvals(monodromy)

    #saving off eigenvalues
    for val in eigenvalues:
        real_eigenvals.append(tuple([alpha, val.real]))
        imag_eigenvals.append(tuple([alpha, val.imag]))

fig, axs = plt.subplots(2, figsize=(12,8))
fig.suptitle('Floquet multipliers for beta = {}'.format(beta))
axs[0].scatter(*zip(*real_eigenvals))
axs[0].set_xlabel("alpha")
axs[0].set_ylabel("Real")

axs[1].scatter(*zip(*imag_eigenvals))
axs[1].set_xlabel("alpha")
axs[1].set_ylabel("Imaginary")
```

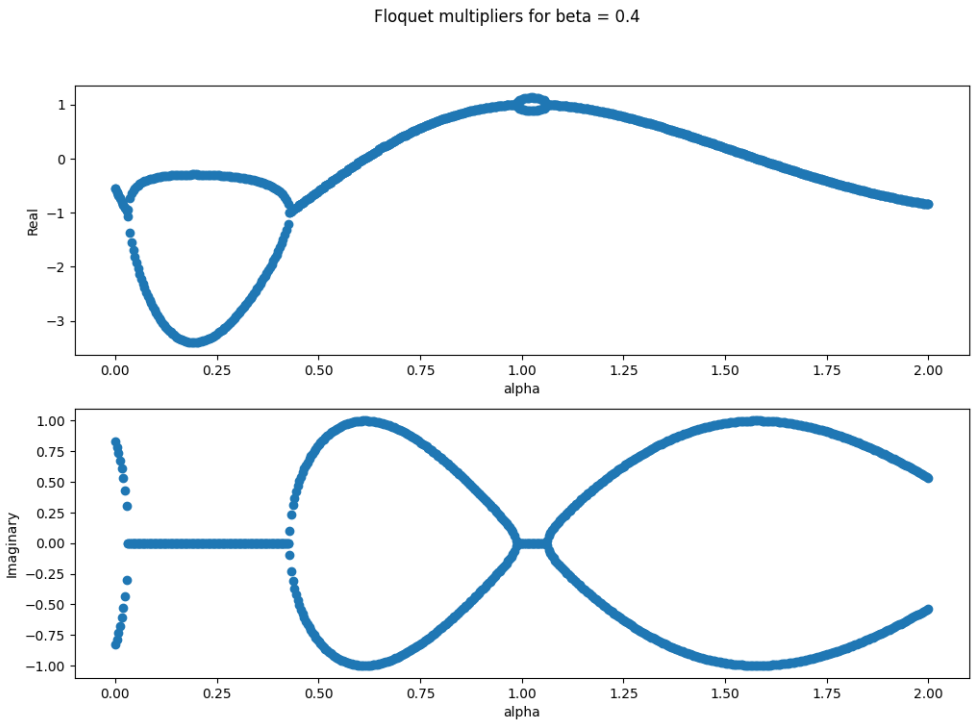


Figure 3: Real and Imaginary values for Floquet multipliers.

5. The following are examples of more open-ended questions and you are encouraged to explore at lease one of them, and/or pose and answer your own:

   (b) For this, let's return to the child on the swing modeled by the nonlinear equation 3 rather than the linearized model 14 that assumes a small angle approximation. Choose a value of α inside the first resonance tongue for $\beta = 0.1$. Choose an initial condition corresponding to a small angle for the pendulum. Solve this initial value problem numerically and plot $\theta(t)$ (and/or $E(t)$ and/or $\Omega(t)$ and/or $\ell(t)$) for some time interval and compare with the numerical solution of the linearized problem. It could be especially interesting to think about resonant forcing in the context of Problem Set #0 where you computed the period of the pendulum as a function of its amplitude, for instance.

Solution:

For this problem, I started from scratch and numerically solved both the linear and nonlinear systems for $\alpha = 0.25$, $\beta = 0.2$. I chose these numbers just to give slightly different results, which as shown in 4, should behave similarly to the requested values. I chose initial conditions $\theta_0 = \pi/10$ and $\Omega_0 = 1$. The code which solves the two systems is shown below, followed by the generated plots. Unfortunately, there is something flawed in my code, which resulted from switching to interactive plotting for various values of the initial conditions. In this transition I had to update some packages, so this might be the cause of the issue. Fortunately enough, I saved off the plots before this problem happened, so I have something legible to present. The "better" plots are shown in 4, while the plots which contain the error are shown in 5. Note that I will only comment on the correct plots, as they give interpretable solutions.

Comparing the linear and nonlinear systems, it looks as though their difference results in an increasing phase shift in θ and Ω, with a slight increase in amplitude for the nonlinear system. The Length of the pendulum arm seems to stay the same, however, which was strange when I first saw it. I confirmed that I was not plotting the same function twice. The most interesting plot, I feel, is the energy difference. We can see the nonlinear system appears to result in higher energies for each period. As we stray further from starting conditions, the two systems will steadily diverge. I would have plotted the period as a function of the amplitude for both systems, but this would result in nonsense since all I have is the code which results in an error.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import integrate

#Defining our linear and nonlinear systems
#Note they differ only by the taylor expansion of sin(x)
```

```
def nonlinear_system(x, tau, alpha, beta):
    dthetadt = x[1]
    dOmegadt = -(alpha + beta * np.cos(tau))*np.sin(x[0])

    return [dthetadt, dOmegadt]

def linear_system(x, tau, alpha, beta):
    dthetadt = x[1]
    dOmegadt = -(alpha + beta * np.cos(tau))*x[0]

    return [dthetadt, dOmegadt]

#Defining functions for length and energy
# These are just for plotting purposes
def ell(theta, eps, tau):
    return 1 + eps * np.cos(tau)

def energy(theta, Omega, alpha, tau):
    return 0.5*Omega**2 - alpha*np.cos(theta)

#setting parameters
alpha = 0.25
beta = 0.2

#Setting initial conditions
#These are random numbers
theta0 = np.pi/10
Omega0 = 1
x0 = [theta0, Omega0]

#Setting the time
tau = np.linspace(0, 10, 1000)

lin_solution = integrate.odeint(linear_system, x0, tau, args=(alpha, beta))
nonlin_solution = integrate.odeint(nonlinear_system, x0, tau, args=(alpha, beta))

ltheta = np.take(lin_solution, 0, axis=1)
lOmega = np.take(lin_solution, 1, axis=1)

nltheta = np.take(nonlin_solution, 0, axis=1)
nlOmega = np.take(nonlin_solution, 1, axis=1)
```

```
#Setting the xticks as increments of pi/2
tau_ticks = np.arange(0, 10, np.pi/2)
tau_ticks_label = [f'0' if n == 0 else f'pi/2' if n == 1 else f'{int(n/2)}pi' if
    n%2 == 0 else f'{int(n)}pi/2' for n in tau_ticks / (np.pi / 2)]

#Plotting Thetas
plt.figure(figsize=(10, 16))
plt.subplots_adjust(hspace=0.5)

plt.subplot(4, 1, 1)
plt.plot(tau, ltheta, label= "Linear", color = 'b')
plt.plot(tau, nltheta, label= "NonLinear", color= 'g')
plt.xticks(tau_ticks, tau_ticks_label)
plt.xlabel("tau")
plt.ylabel("theta")
plt.title("Theta linear vs nonlinear")
plt.legend()

#Plotting Omegas
plt.subplot(4, 1, 2)
plt.plot(tau, lOmega, label= "Linear", color= 'b')
plt.plot(tau, nlOmega, label= "NonLinear", color= 'g')
plt.xticks(tau_ticks, tau_ticks_label)
plt.xlabel("tau")
plt.ylabel("Omega")
plt.title("Omega linear vs nonlinear")
plt.legend()

#plotting Lengths
plt.subplot(4, 1, 3)
plt.plot(tau, ell(ltheta, beta/alpha, tau), label= "Linear", color = 'b')
plt.plot(tau, ell(nltheta, beta/alpha, tau), label= "NonLinear", color= 'g')
plt.xticks(tau_ticks, tau_ticks_label)
plt.xlabel("tau")
plt.ylabel("Length")
plt.title("Length linear vs nonlinear")
plt.legend()

#Plotting Energies
plt.subplot(4, 1, 4)
plt.plot(tau, energy(ltheta, lOmega, alpha, tau), label= "Linear", color= 'b')
plt.plot(tau, energy(nltheta, nlOmega, alpha, tau), label= "NonLinear", color=
```

```
    'g')
plt.xticks(tau_ticks, tau_ticks_label)
plt.xlabel("tau")
plt.ylabel("Energy")
plt.title("Energy linear vs nonlinear")
plt.legend()

plt.show()
```

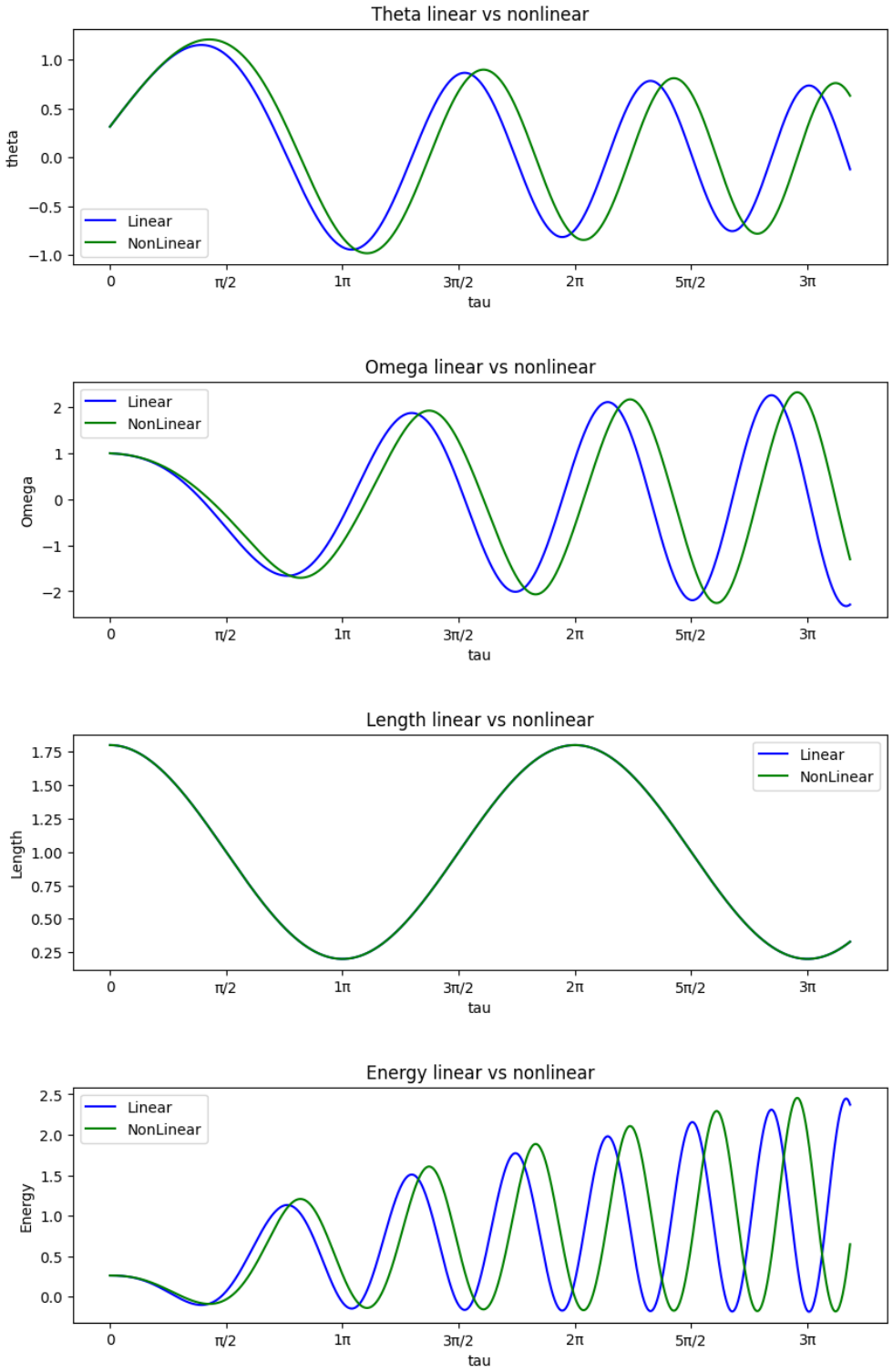


Figure 4: Plots to show the difference in the nonlinear versus linear systems.

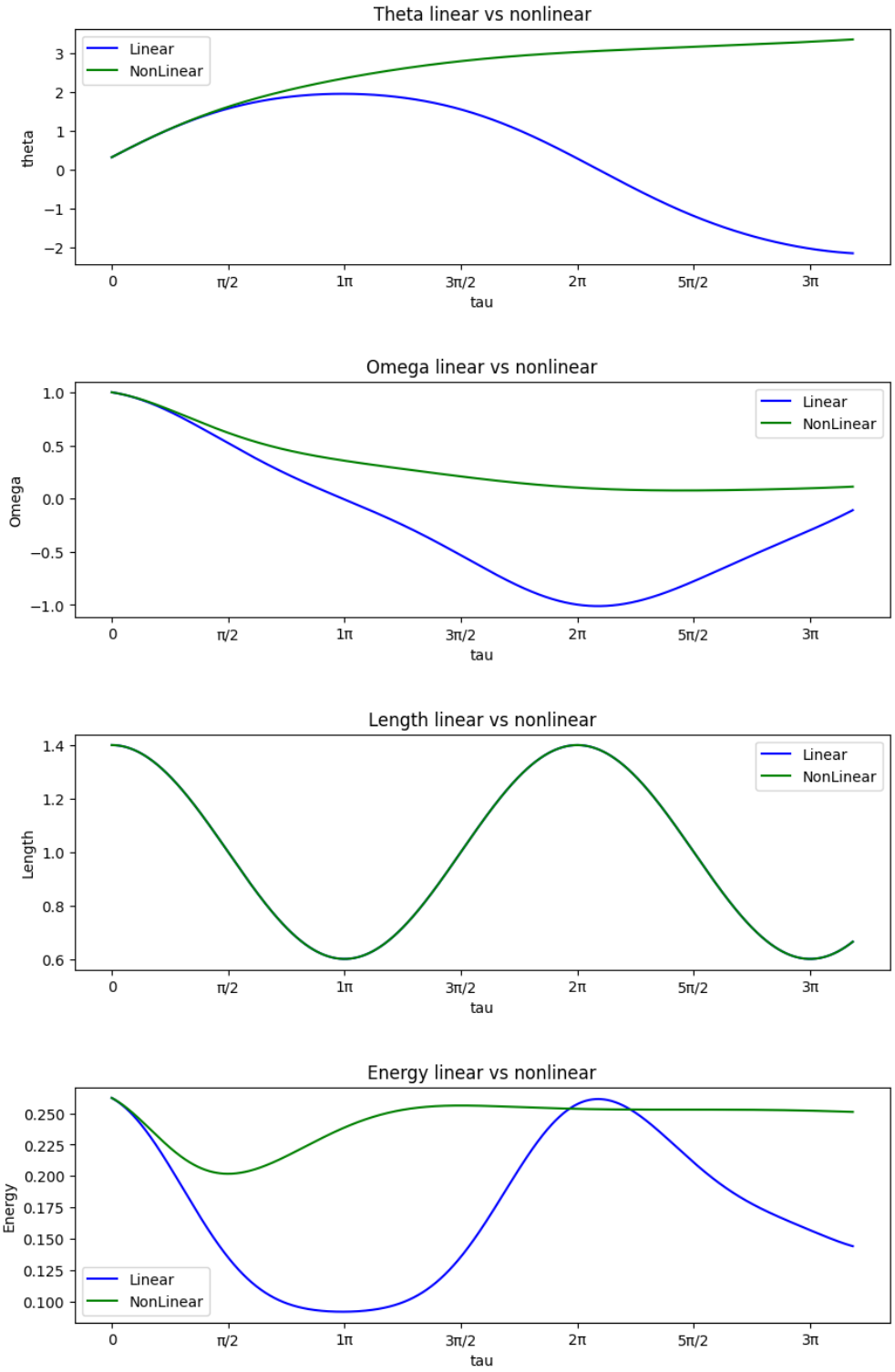


Figure 5: Error plots, not sure what the problem is.