Pseudo-spectra

November 9, 2018

1 Pseudo-Spectra and Non-Modal Instabilities

Kieran Ricardo

1.1 Introduction and Non-Normal Operators

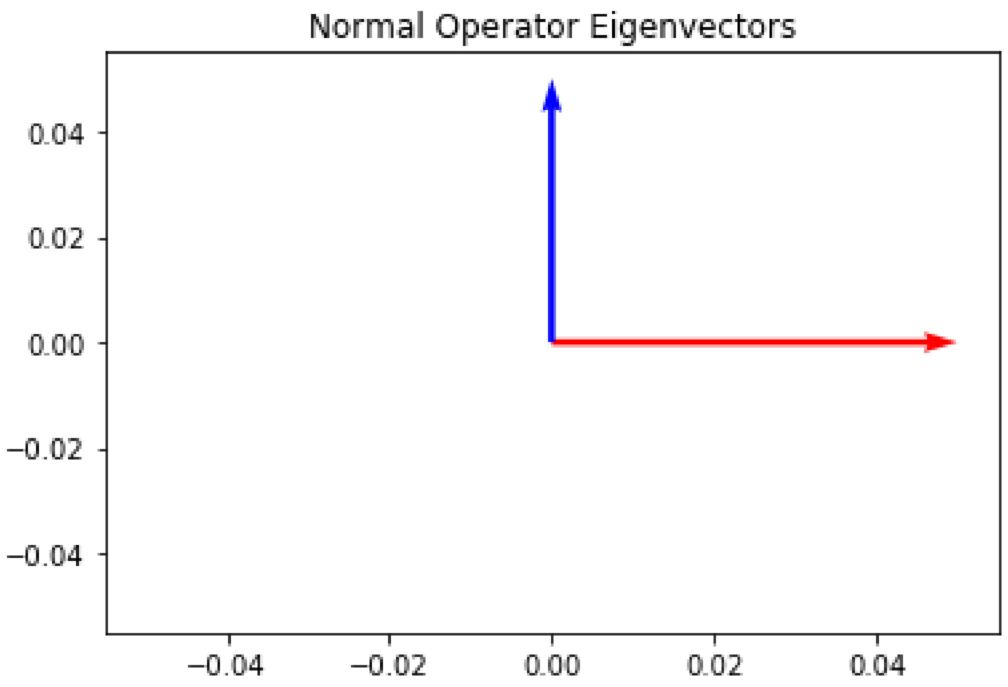
Historically, the hydrodynamic stability of flows was analyzed by utilizing the following procedure: linearlise the flow about the steady state, invoke squire's theorem that 2-dimensional pertubation will ultimately grow the fatest, and solve the resulting 2-dimensional linear PDE for the eigen-modes. If any unstable modes existed (eigenvalues with positive real or imaginary parts depending on convention) then the flow was said to be unstable to infintismal pertubations. This classic methodology has been utilized in countless papers and research projects however the results did not always agree with experiments. The classic example is Couette flow; eigen-analysis claims that this flow is stable at all Reynolds numbers but instability has been observed for Reynolds numbers as low as 350. The question then arises, where did we go wrong? Some suggested that the linearisation of Navier-Sokes equation (NSE) was to blame, others, more provactively, suggested that the NSE itself was wrong. Recent results have shown that neither of these are the case, in fact the issue is the eigen-analysis itself. Eigen-analysis specifies growth rates in the direction of the eigen-modes, and in fact in the limit as time goes to infinity all initial conditions will converge on the eigen-modes - this nice fact means that eigen-analysis can determine the end result of the pertubations, whether they will grow eponentially of decay to zero. So, what's wrong with this?

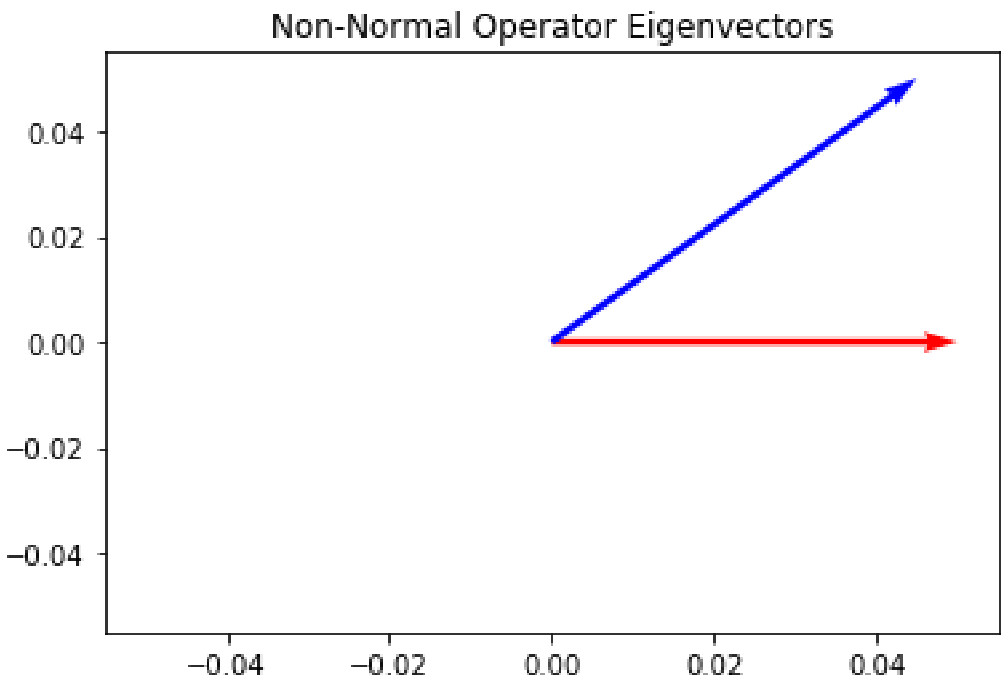
With the previous results in mind eiegen-analysis sounds like the natural method to determine hydro-dynamic stability but a crucial part of picture of the picture is missing: eigen-analysis can determine the ultimate behaviour of a system but can it determine behaviour at intermediate times? The answer, as we will see, is not always; if a system is "non-normal" transient growth can indeed occur in stable systems and this transient growth can be sufficient to trigger non-linear effects and transition away from the initial steady state.

To understand what a non-normal operator is we will have to first cover what a normal operator is. Briefly, a normal operator is an operator with orthogonal eigenvectors. This allows any initial condition to be neatly decomposed by projecting the initial state onto each eigenvector. The resulting components (projected onto the eigenvectors) will grow at the rate of the respective eigenvalues - this means that if a normal operator has all stable eigenvalues each component will monotonically decay to zero and thus the system will never experience growth for any initial condition. A non-normal operator is an operator that is not normal, therefore it's eigenvectors are not orthogonal and decomposition onto its eigenspace is not a straight forward - infact there are even initial condition that will allow some transient growth despite stable eigenvalues. To understand this slightly more intuitively consider the follow plots of eigen-vectors - the normal eigenvectors specify the growth in all directions however the non-normal eigenvalue do not! And so it is possible that transient growth can occur despite the eigenvalue, in fact for eigen-stable operators the states which will instantenously growth the fastest are perpendicular to the eigenvectors.

To understand these results this report will explore the mathematical theory of non-normal operators and transient growth rates with a focus on simple 2x2 systems. In the last section a toy model of a possible mechanism for a transition to turbulence is explored where initial non-normal transient growth can be sufficient to trigger non-linear effects transitioning the system to a higher-energy steady state.

```
In [8]: plotVecs()
```





1.2 Pseudo-spectra

The mathematical theory that underpins these results is the theory of pseudo-spectra. Pseudo-spectra are a generalisation of eigenvalues. For the purposes of explanation this section will present results in terms of a square matrix A, however there results can be trivially extending to arbitrary linear operators by substiting the relevant operator norms. For a given linear operator A, an eigenvector of A satisfies the following:

$$A\mathbf{u} = \lambda\mathbf{u}$$

(1)

$$\|A - \lambda I\| = 0$$

(2)

$$\left\|(A - \lambda I)^{-1}\right\| = \infty$$

(3)

Can these results be generalised? Consister a λ where equation (3) is large but finite, to make this idea precise:

$$\left\|(A - \lambda I)^{-1}\right\| > \epsilon^{-1}$$

A vector - a pseudo-eigenvector - in this direction will almost resonate - pseudo-resonate. The set of all λ for a given ϵ is formally defined as:

$$\Lambda_\epsilon = \left\{ \lambda : \left\| \left(A - \lambda I \right)^{-1} \right\| > \epsilon^{-1} \right\}$$

Where $\epsilon > 0$ is typically small. A natural question arises, for a given ϵ what structure does λ have? Is it possible to have stable eigenvalues (i.e. eigenvalues in the lower complex plane) with unstable pseudo-eigenvalues with small ϵ?

For normal operators, defined in equation (4), Λ_ϵ is the set of all points within ϵ distance of its egeinvalues. However, for non-normal opertors Λ_ϵ can be much, much larger as we will see.

1.3 Transient Growth Rates

1.3.1 Theory

This section aims to theoretically connect the pseudo-spectra of a dynamical system to its transient growth rates.

For a linear dynamical system:

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u}$$

There is a theorem that states that the maximum transient growth of a system is greater than or equal to the maximum product of $\lambda_\epsilon \epsilon^{-1}$. Formally:

$$\sup \|\exp(-iAt)\| \geq \sup \epsilon^{-1} \lambda_\epsilon$$

Where λ_ϵ is the most unstable pseudo-eigenvalue for ϵ, formally:

$$\lambda_\epsilon = sup_{\omega \in \Lambda_\epsilon(A)} \Im \omega$$

This can be intuitively understood as the system growing faster than linear at rate λ_ϵ until time ϵ^{-1}. An informal justification of this result is as follows:

Consider the above dynamical system, where there unit vector $\mathbf{u_0}$ is a pseudo-mode of A.

$$\frac{d\mathbf{u}}{dt}(t = 0) = A\mathbf{u_0} = \lambda \mathbf{u_0} + \epsilon \mathbf{v}$$

After one time step the system becomes:

$$\mathbf{u}(\delta t) \approx (1 + \lambda \delta t)\mathbf{u_0} + \epsilon \delta t \mathbf{v} \approx (1 + \lambda \delta t)\mathbf{u_0}$$

$$\frac{d\mathbf{u}}{dt}(t = \delta t) \approx \lambda(1 + \lambda \delta t)\mathbf{u_0} + \epsilon(1 + \lambda \delta t)\mathbf{v}$$

Therefore:

$$\mathbf{u}(2\delta t) \approx (1 + \lambda \delta t)^2 \mathbf{u_0} + (1 + \lambda \delta t)\epsilon \delta t \mathbf{v} \approx (1 + 2\lambda \delta t)\mathbf{u_0} + (1 + \lambda \delta t)\epsilon \delta t \mathbf{v}$$

And hence:

$$\mathbf{u}(n\delta t) \approx (1 + n\lambda \delta t)\mathbf{u_0} + (1 + (n-1)\lambda \delta t)\epsilon \delta t \mathbf{v}$$

Therefore growth in this system will occur approximately linearly until the second term becomes: $n\delta t \epsilon \approx 1$ or $t = \epsilon^{-1}$ and from this:

$$max(\,\|\mathbf{u}\|\,) > \lambda\epsilon^{-1}$$

The key idea here is that if the the pseudo-spectra protrude into the upper half complex plane, i.e. are "unstable", and are almost resonant, i.e. ϵ is small, there can be large transient growth despite stable eigenvalues.

1.3.2 Example

As an example consider the following linear dynamical system:

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u}$$

Where:

$$A = \begin{bmatrix} -R^{-1} & 1 \\ 0 & -2R^{-1} \end{bmatrix}$$

The normalized eigenvectors of this system are:

$$\mathbf{u_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mathbf{u_2} = \begin{bmatrix} \sqrt{\dfrac{4R^4}{4R^4+1}} \\ \sqrt{\dfrac{1}{4R^4+1}} \end{bmatrix}$$

With eigenvectors $-R^{-1}$ and $-2R^{-1}$ respectively. From this it is clear that this system is modally stable (negative eigenvalues) and that:

$$\lim_{R\to\infty} \mathbf{u_2} = \mathbf{u_1}$$

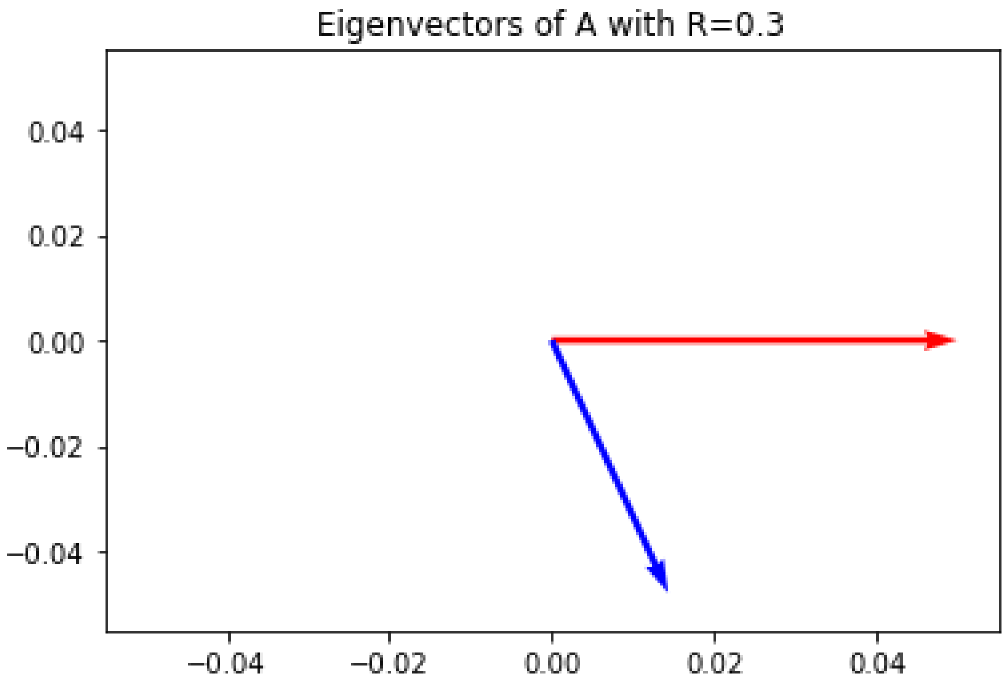
Therefore, the larger R is the closer the eiegenvectors are to parallel and hence A is more non-normal. As was discussed above, non-normal operators can have large regions of pseudo-spectra. It also worth noting that as $\lim_{R\to\infty}$ both eigenvalues move towards zero and the system becomes approximately marginally stable - i.e. sustained oscillations can occur. The psuedo spectra of A for various values of R are plotted below.

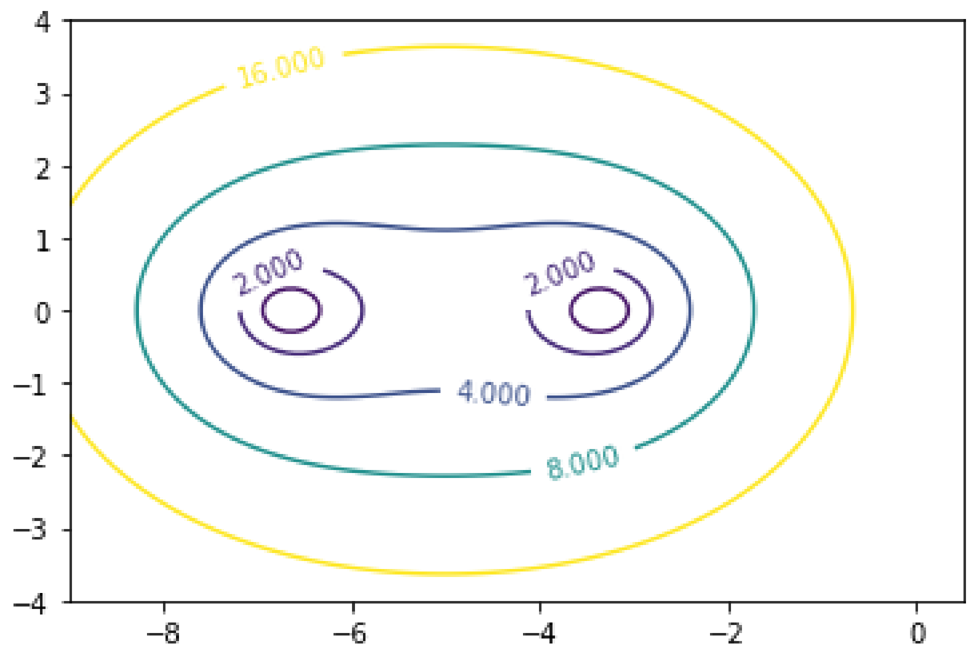
R=0.3 For R=0.3 the eigenvectors are almost perpendicular and therefore A is almost normal. This is reflected in the contour plot of epsilon values for different pseudo-eigenvalues, the contours start out as concentric circles around the two eigenvalues which merge to form as an eliptical shape. For no small value of epsilon (at least for $\epsilon < 16$) do the pseudo-eigenvalues protude onto the right half of the complex plane. The maximum value of $\epsilon^{-1}\lambda_\epsilon$ is 0.05 and therefore no significant amount of transient growth would be expected for this system.

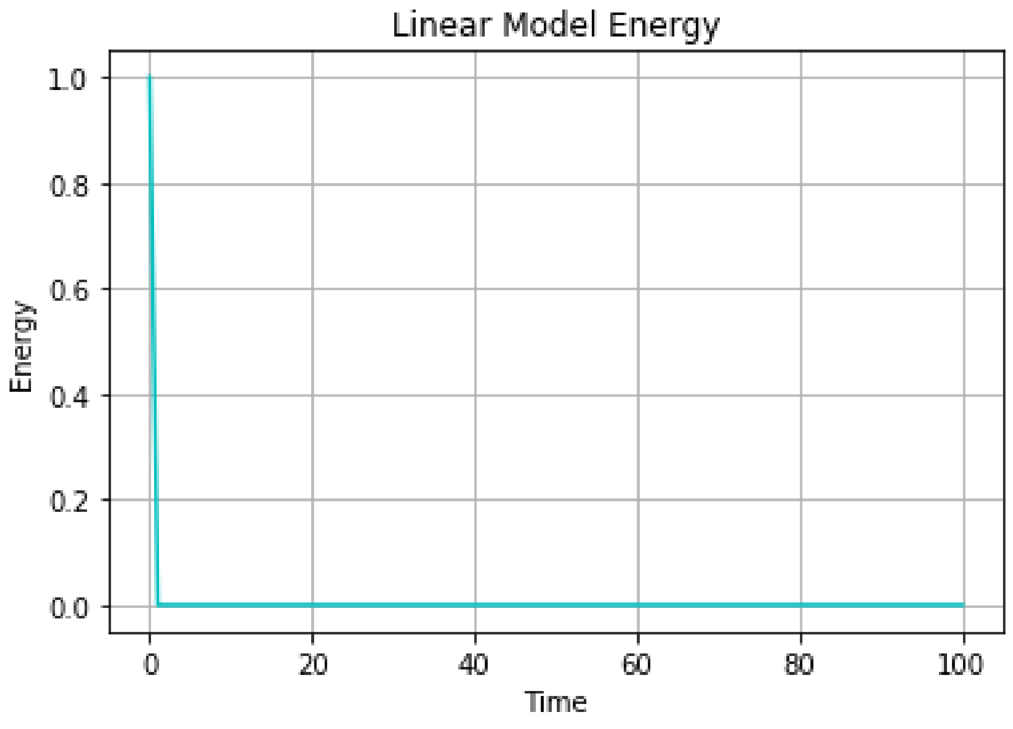
For the simulation the system was initially perpendicular to the red eigen-vector, it can be seen that the energy of the system decays to zero almost instaneously while simultaneously rotating toward the red eigenvector, as expected.

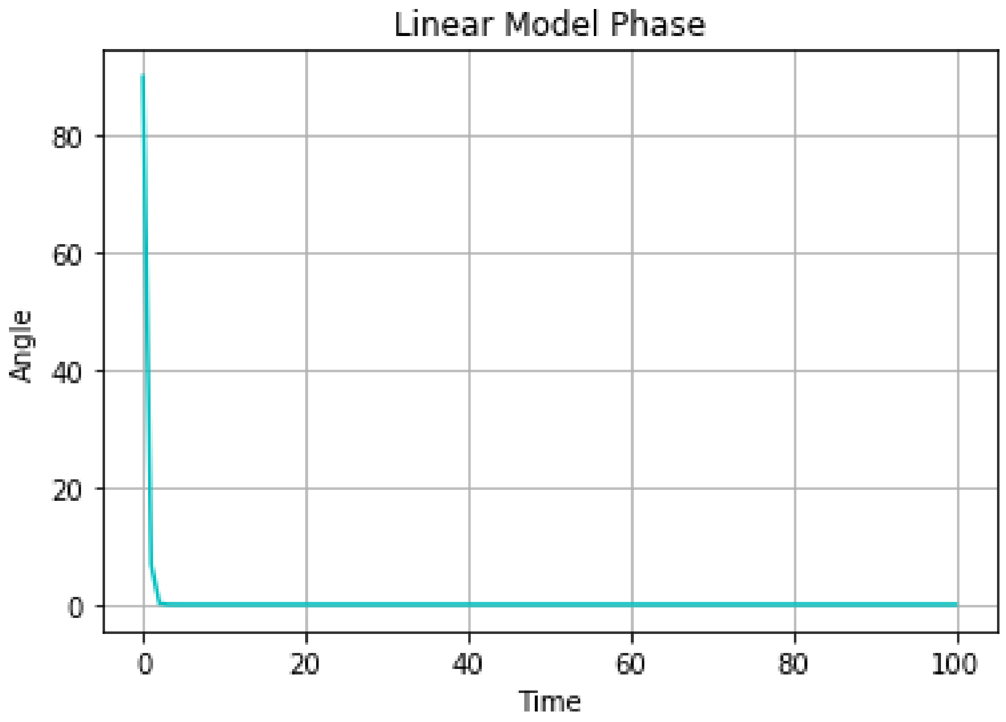
```
In [130]: R = 0.3
          A = np.array([[-1/R,1],[0,-2/R]])
          print(max(map(lambda r: r*1/epsilonCalc(A, r), np.linspace(-3,10,1000))))
          eigenvecPlot(R=0.3)
          contourEpsilonPlot(R=0.3)
          linearGrowthPlots(R=0.3)
```

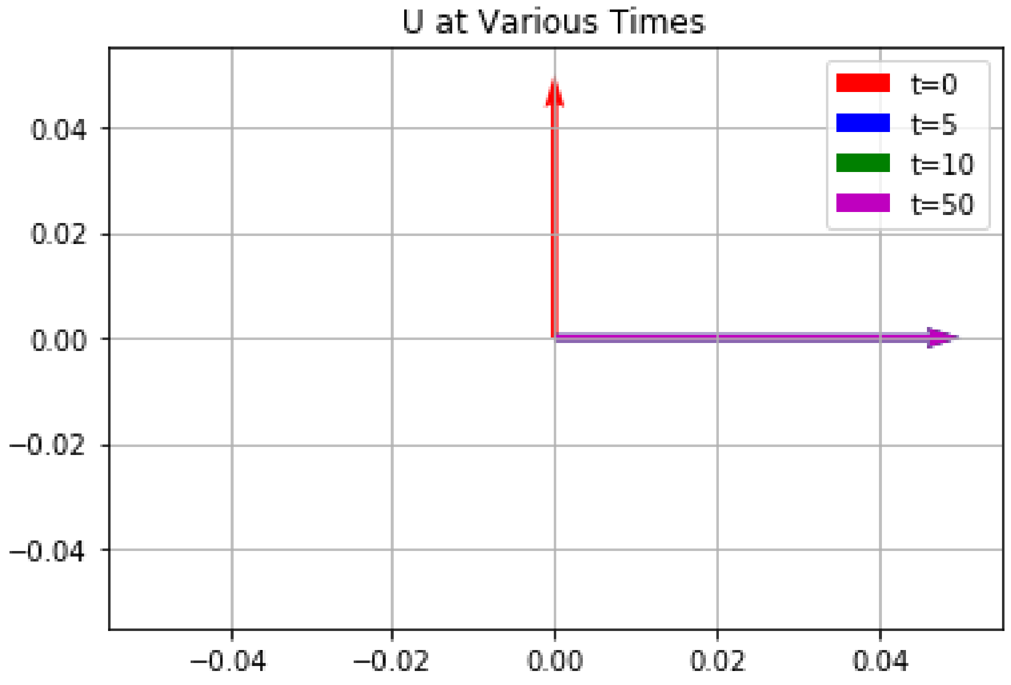
```
0.05147185856735432б
```




16.000
8.000
4.000
2.000
2.000
4
3
2
1
0
−1
−2
−3
−4
−8
−6
−4
−2
0


Linear Model Energy
Energy
Time
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

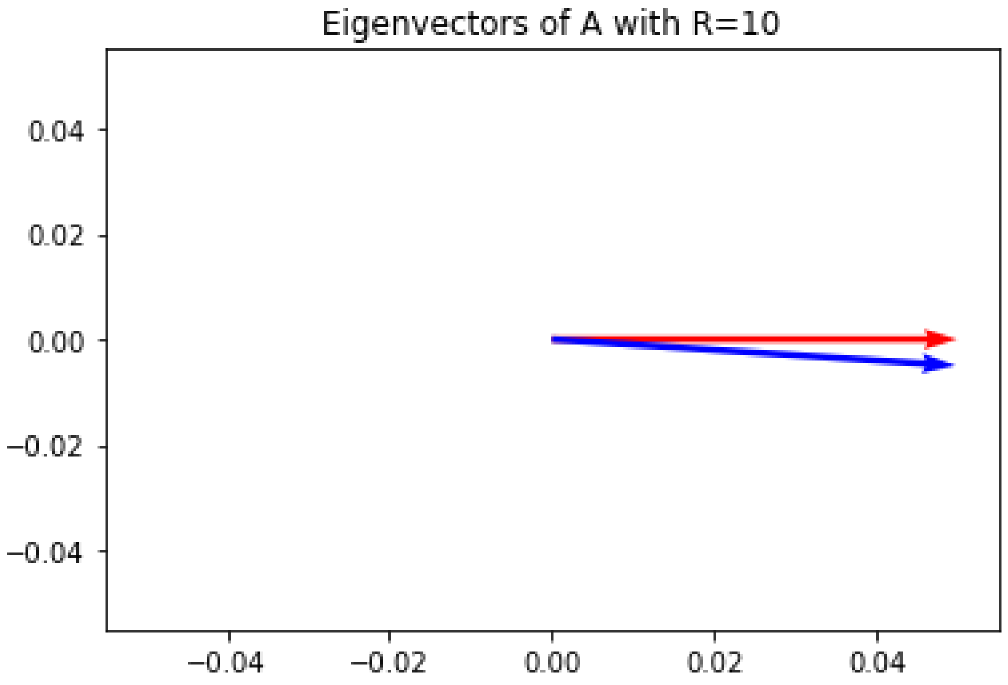

Linear Model Phase
Angle
Time
0
20
40
60
80
100

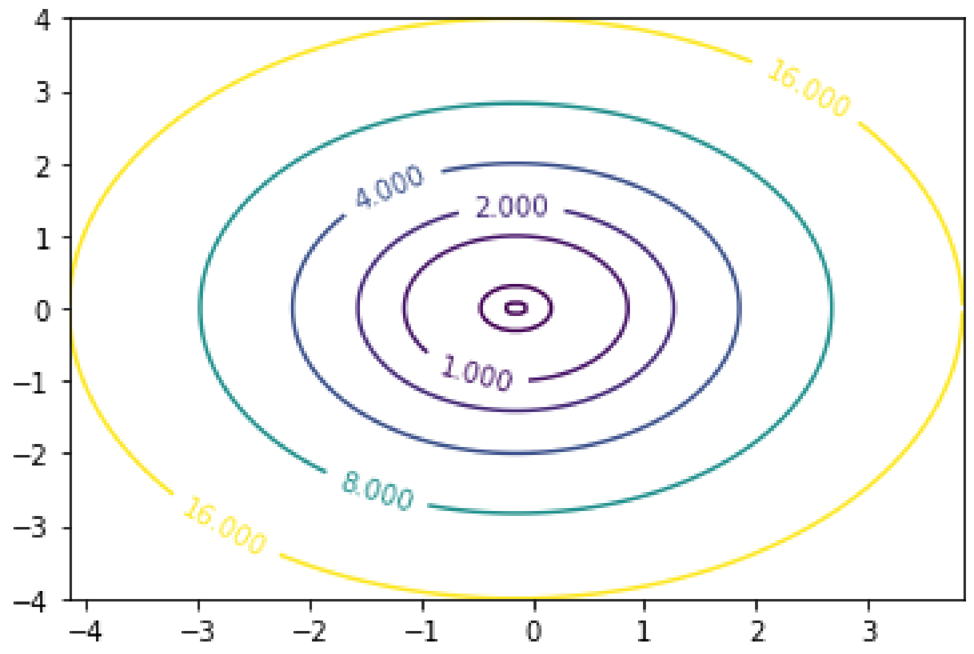

U at Various Times
t=0
t=5
t=10
t=50
0.04
0.02
0.00
−0.02
−0.04

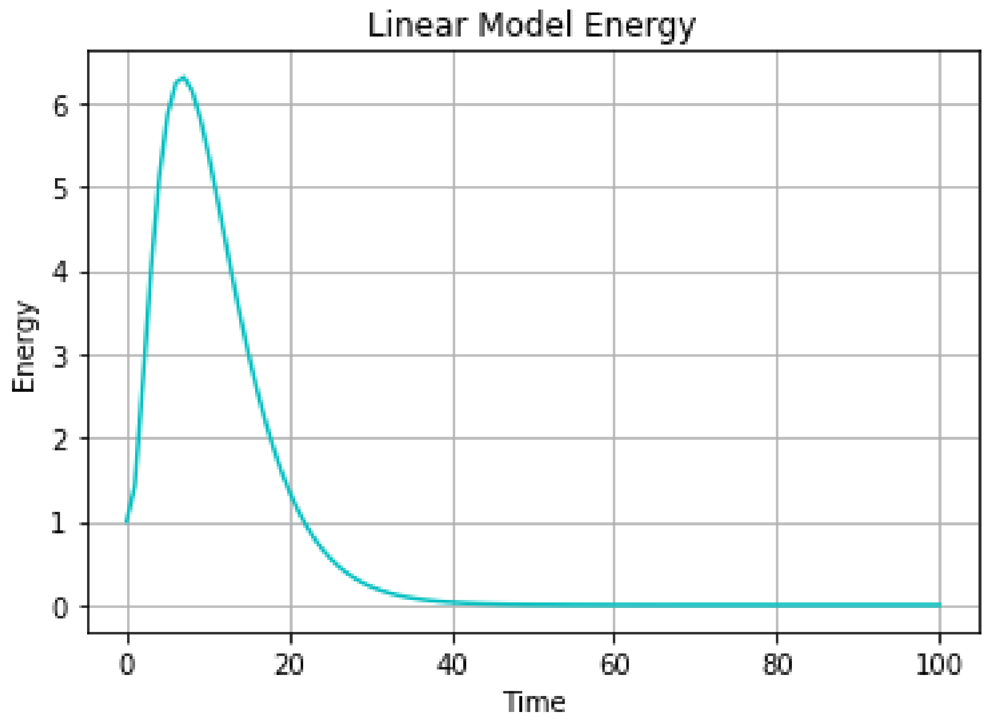
R=10 The eiegen-vectors of this system are virtually parallel and therefore this system is highly non-normal. This is reflected in the contour plots which protrude onto the right half of the complex plane for all epsilon. The maximum value of $\epsilon^{-1}\lambda_\epsilon$ was calculated to be 1.7, approximated 30 times larger than the value calculated for R=0.3. From this, we can expect large transient growth for the right initial conditions. This is exactly what is seen, for an intial state perpendicular to the red eigenvector there is initially large growth however the system is also rotated to align with the red eigenvector and once it is close enough to this eigenvector the energy begins to decay.

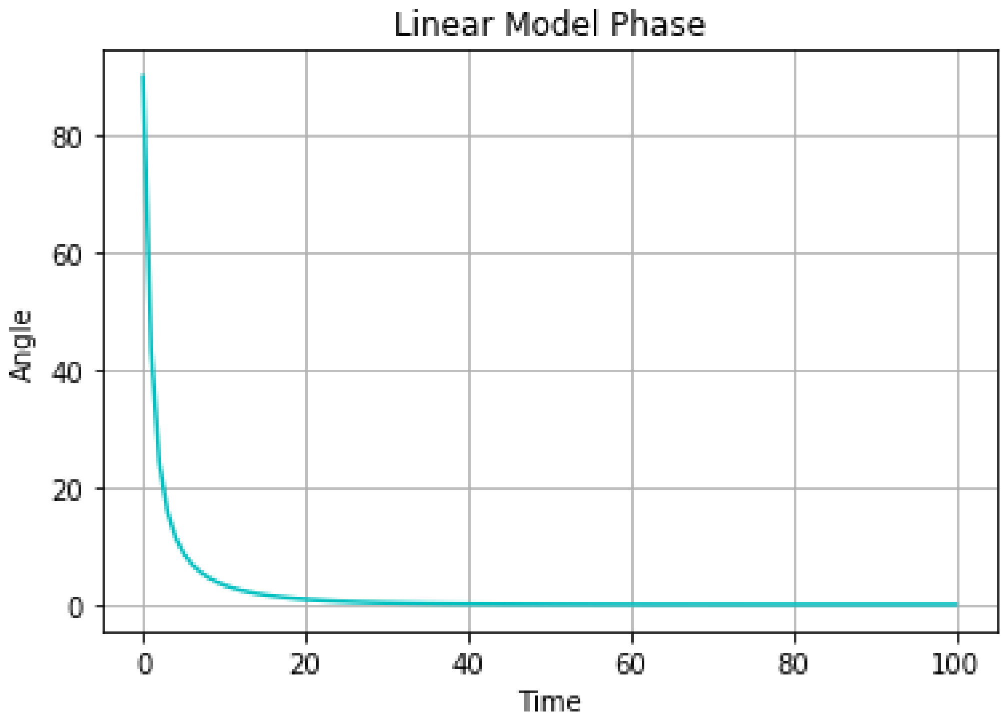
```
In [133]: R = 10
          A = np.array([[-1/R,1],[0,-2/R]])
          print(max(map(lambda r: r*1/epsilonCalc(A, r), np.linspace(-3,10,1000))))
          eigenvecPlot(R=10)
          contourEpsilonPlot(R=10)
          linearGrowthPlots(R=10)
```

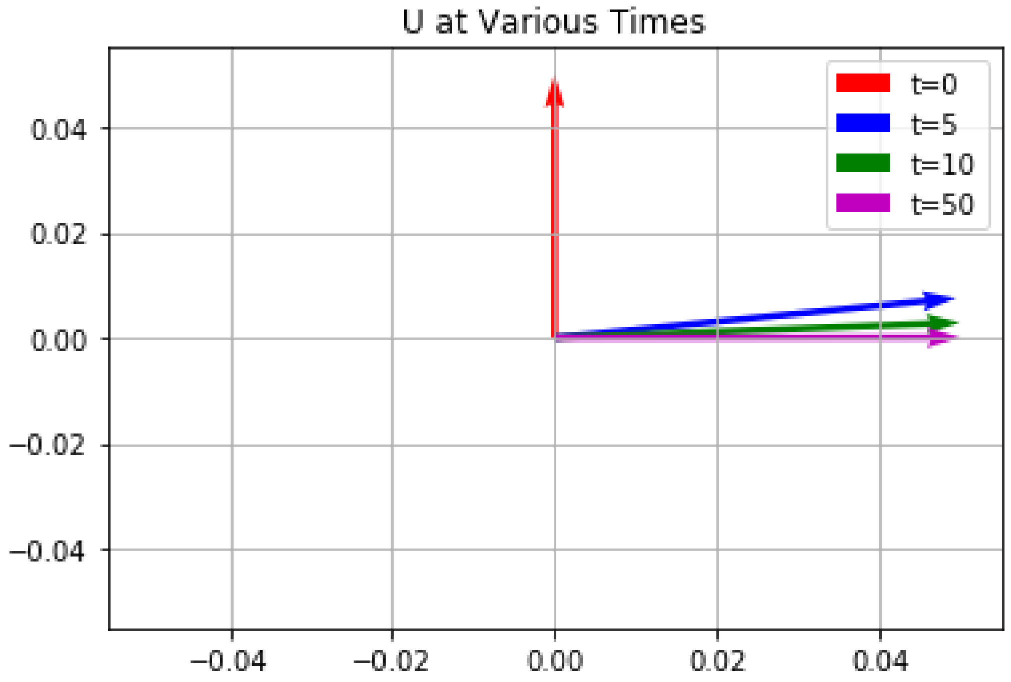
```
1.715124946948326
```




16.000
4.000
2.000
1.000
8.000
16.000


Linear Model Energy
Energy
Time


Linear Model Phase
Angle
Time
0
20
40
60
80
100


U at Various Times
t=0
t=5
t=10
t=50
0.04
0.02
0.00
−0.02
−0.04

1.4 Model of Transition to Turbulence

1.4.1 Introduction of Non-Linear System

This section will consider the following simple non-linear dynamical system:

$$\frac{d\mathbf{u}}{dt} = A\mathbf{u} + \|\mathbf{u}\|\, B\mathbf{u}$$

Where:

$$A = \begin{bmatrix} -R^{-1} & 1 \\ 0 & -2R^{-1} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Unpacking this system, the first term corresponds to the non-normal linear dynamical system that we explored above, and the second term is non-linear and acts perpendicular to the flow and thus it is energy conserving. The second term can be thought of as energy conserving "mixing" - it does not change the magnitude of $\mathbf{u}$, it only rotates it. The derivative of energy growth with respect to time is given by:

$$\frac{d(\mathbf{u}\cdot\mathbf{u})}{dt} = \mathbf{u}\cdot\frac{d\mathbf{u}}{dt} = \mathbf{u}\cdot\big(A\mathbf{u}\big) + \|\mathbf{u}\|\,\mathbf{u}\cdot\big(B\mathbf{u}\big) = \mathbf{u}\cdot\big(A\mathbf{u}\big)$$

$$\frac{d\,\|\mathbf{u}\|^2}{dt} = u_x\big(R^{-1}u_x + u_y\big) + 2R^{-1}u_yu_y$$

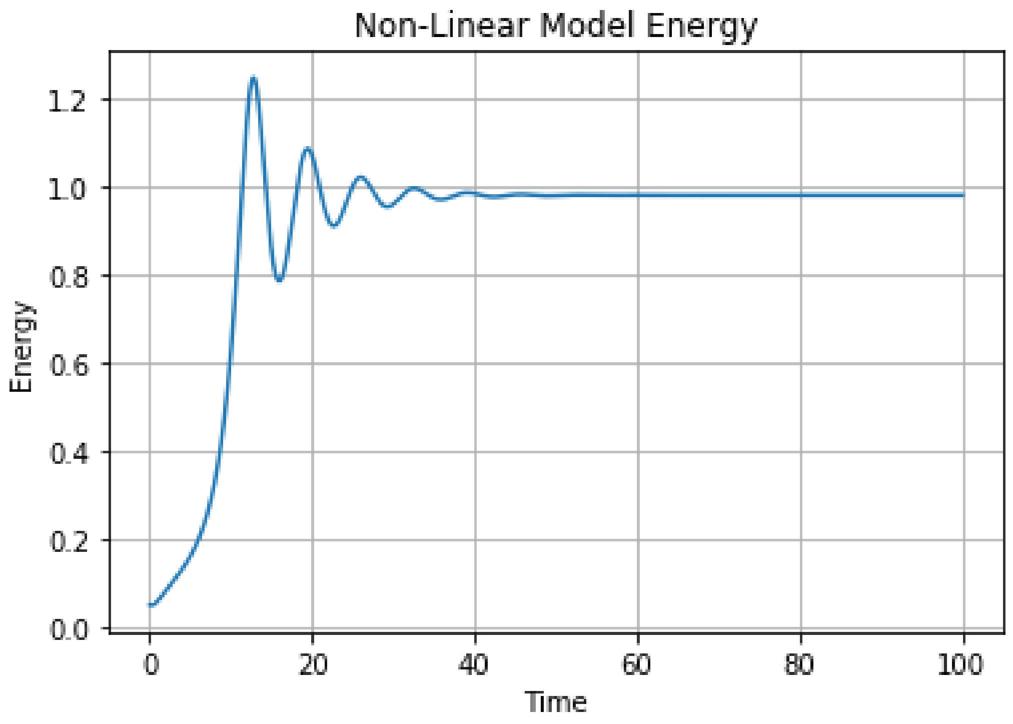
And thus the energy growth rate is only dependant on A and $\mathbf{u}$. The energy growth is is proportional to the magnitude of the energy and dependant on the phase of $\mathbf{u}$. There are some phases of $\mathbf{u}$ which allow large energy growth, however in the linear case $\mathbf{u}$ is rotated away from these states. The non-linear mixing term serves to rotate the system back towards states with large energy growth enabling self-sustaining energy growth, this rotation effect increases quadratically with energy. A question naturally arises then, can the energy grow quickly enough to enable the non-linear mixing term to conteract the rotation towards the unstable eigenvectore and therefore become self sustaining? The plots below show that this is indeed possible and that there are two fixed points of this system. This is some threshold value of the initial energy above which the energy growth is self-sustained yielding the non-zero energy fixed point, and below which the energy growth is not self-sustaining and decays to zero.

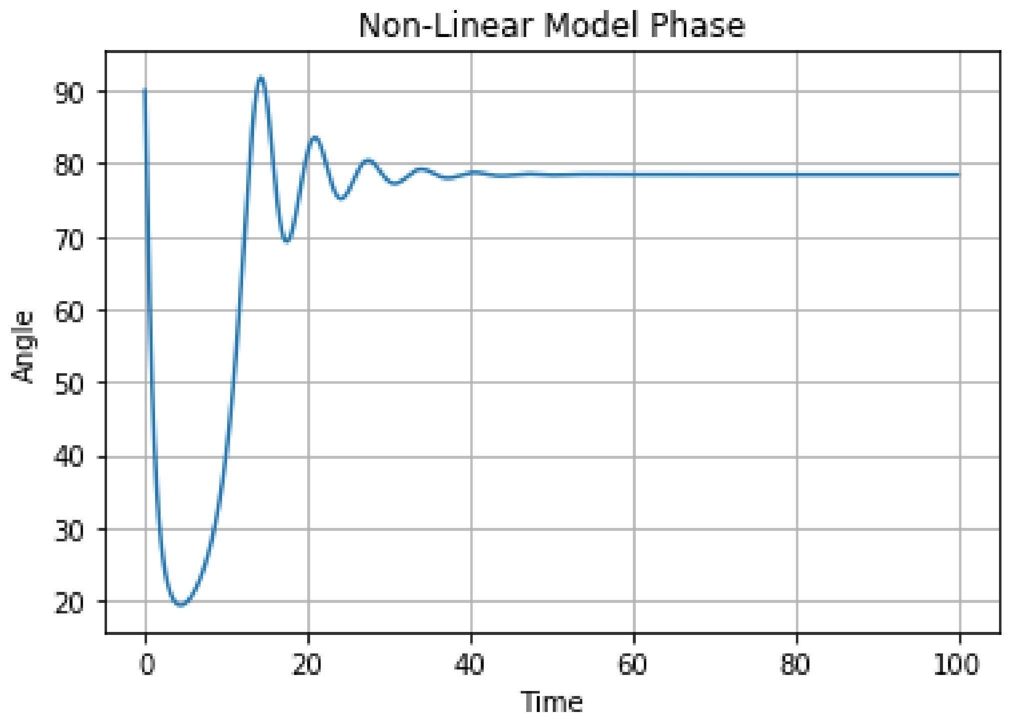
1.4.2 Plots of the Non-Linear System

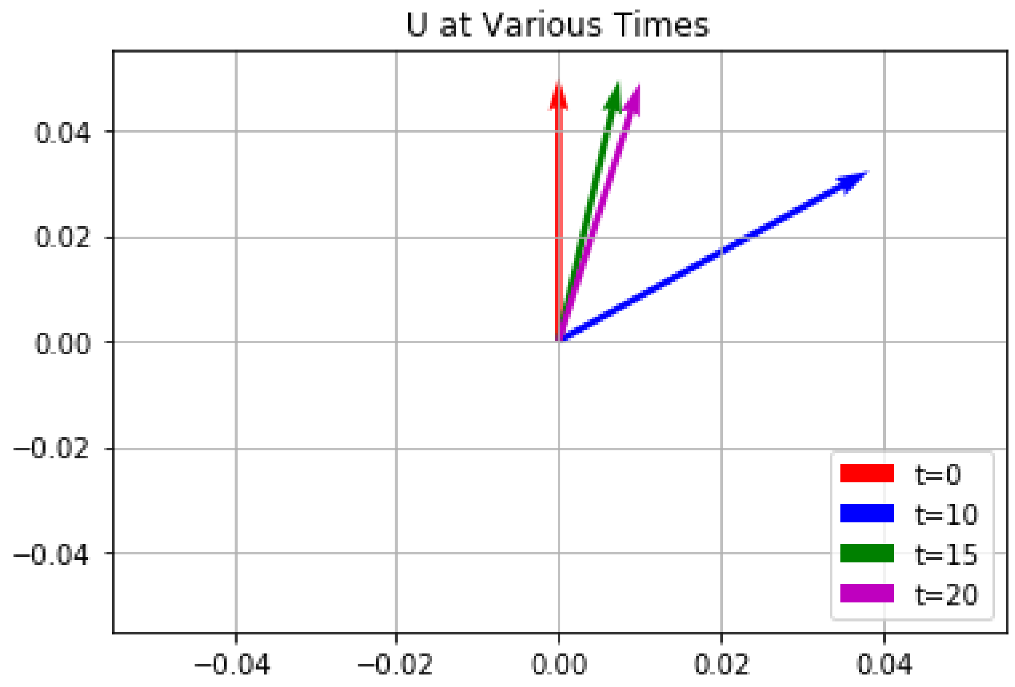
For the following plots R=10 and all initial states are perpendicular to the red eigenvector as above. The energy (half of the magnitude squared) are plotted with respect to time. The first series of plots mimic those shown above for the linear system. There is initial enery growth and rotation towards the stable eigenvector however past a certain point the non-linear term becomes sufficiently large an the system rotates away from the eigenvector towards a initial state where growth can occur. The system overshoots the fixed point and oscillates about it before settling down.

The finally plot shows the energy growth for a variety of initial energy states, below a certain threshold the system decays to zero and above this threshold the system transitions to the non-zero energy equilibrium.

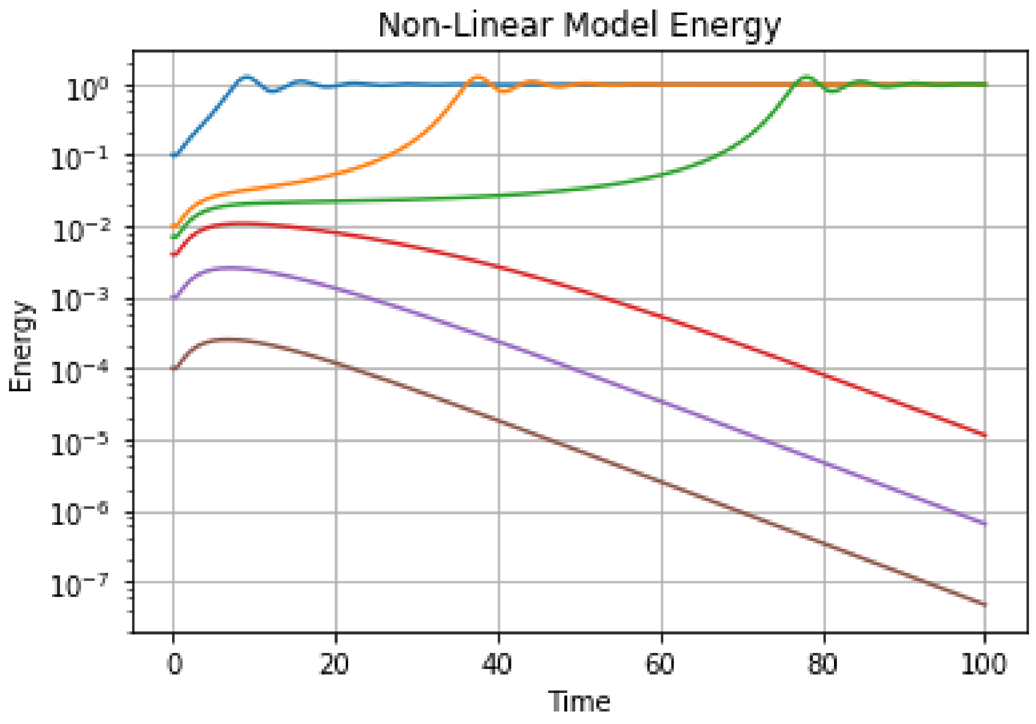
```
In [137]: nonLineModel(5e-2)
          nonLineModelInits()
```







```
In [138]: nonLineModelInits()
```



1.4.3 Mathematical Theory of the Transition Threshold

Consided an initial state of amplitude a_0, by the same arguments used in the linear system upto time R the growth is approximately linear and magnitude of the system is now or order Ra_0 but the system is now almost perpendicular to the stable eigenvector. The non-linear term has the effect of redistributing some of this magnitude back to the initial direction - this term is quadradtic and is order $(Ra_0)^2$. Therefore the energy re-distributed is of order $(Ra_0)^2 R$ as this has been occuring for time R. If this value is of order Ra_0 or greater there is now more magnitude in the initial direction than before and the energy growth is self sustaining. Therefore energy growth will occur if:

$$O\big(R^3 a_0^2\big) \geq O\big(Ra_0\big)$$

There the threshold magnitude is $O(R^{-2})$ which for $R = 10$ is 0.01, very close to the value observed.

1.5 Conclusions

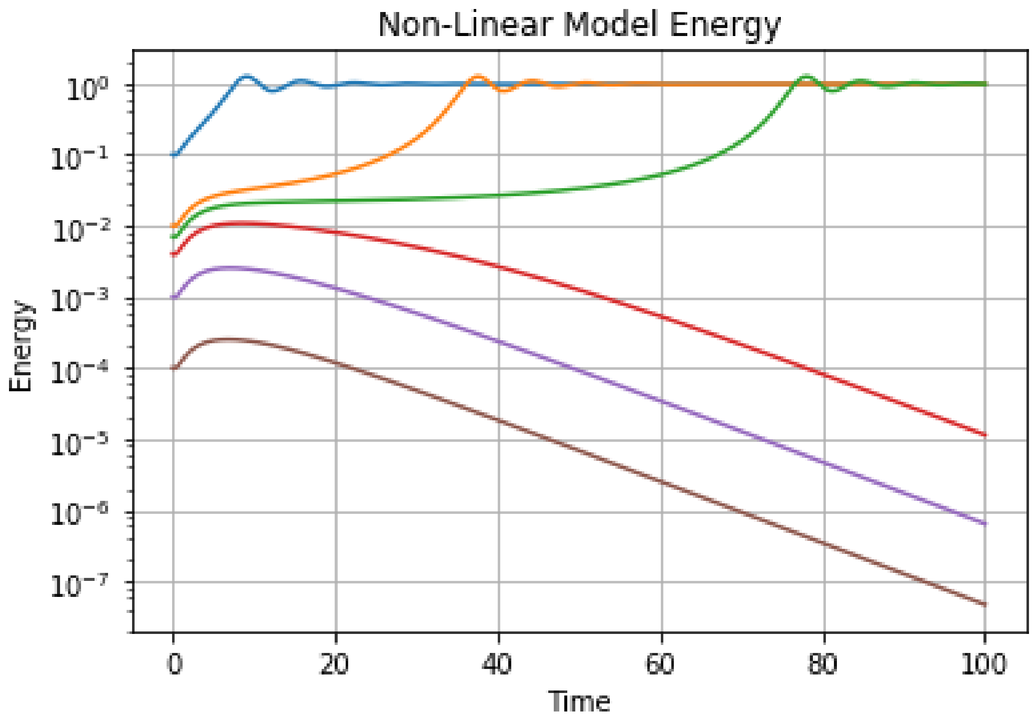
It appears that eigen-stable systems can experience large transient growth rates despite stable eigenvalues. These results can be quantitatively explained via pseudo-spectra - if there a state with unstable pseudo-eigenvalue that almost resonates this state can lead to large transient growth. We also saw that while this transient growth was happening the system is simulatenously rotated towards the stable eigenvalues; causing the system to eventually decay to zero. However, this initial transient growth can be sufficient to trigger non-linear effects - in our toy model of transition to turbulence this enabled self-sustaining growth which caused the system to transition to a high-energy equilibrium for initial states with sufficient energy. While the physical applicability of this to fluid flows is limited the basic ideas still hold - initial transient growth for non-normal eigenstable systems can be sufficient to disrupt the steady state flow.

1.6 Appendix: Python Code

```
In [1]: import numpy as np
        from scipy.linalg import expm, norm
        from matplotlib import pyplot as plt

In [2]: def epsilonCalc(A,c):
                I = np.identity(2)
                return np.linalg.norm(np.linalg.det((A-c*I)))

        def contourEpsilonPlot(R = 10):
            A = np.array([[-1/R, 1], [0, -2/R]])
            I = np.identity(2)

            n = 100
            d = 4
            m = -1.5/R

            re_axis = np.linspace(m-d,max(m+d,0.5), num=n)
            grid = np.stack([re_axis+1j*num for num in np.linspace(-d,d,num=n)]).tr

            epsilonCalcVec = np.vectorize(lambda c: epsilonCalc(A,c))
            eps = epsilonCalcVec(grid)

            X, Y = np.real(grid), np.imag(grid)

            CS = plt.contour(X, Y, eps, levels=[0.01,0.1,1,2,4,8,16])
            plt.clabel(CS, inline=1, fontsize=10)
            plt.show()

In [9]: re_inds, im_inds = np.where(np.isclose(grid,(-0.1+-0j), atol=2d/n))
        re_ind, im_ind = re_inds[0], im_inds[0]

        eps_inv[re_ind,im_ind]


          File "<ipython-input-9-8f3c113a9b9b>", line 1
        re_inds, im_inds = np.where(np.isclose(grid,(-0.1+-0j), atol=2d/n))
                                                                      ^
    SyntaxError: invalid syntax



In [ ]: def eigenvecPlot(R = 10):
            A = np.array([[-1/R, 1], [0, -2/R]])
            origin = [[0],[0]]

            eigenvals, eigenvecs = np.linalg.eig(A)
            eigenvecs[:,1] = -eigenvecs[:,1]
            plt.figure(1)
            plt.title('Eigenvectors of A with R={}'.format(R))
            plt.quiver(*origin, eigenvecs[0,:], eigenvecs[1,:], color=['r','b'],
                       angles='xy', scale_units='xy', scale=20)
            plt.show()

        def linearGrowthPlots(R = 10):
            A = np.array([[-1/R, 1], [0, -2/R]])

            ts = np.linspace(0,100,num=100)
            exp_Ats = np.array(list(map(lambda t: expm(t*A), ts)))

            u0 = np.array([0.0,0.05])
            us = np.array(list(map(lambda eAt: np.dot(eAt,u0), exp_Ats)))
          u_eng = np.array(list(map(lambda u: norm(u)**2/norm(u0)**2, us)))
          u_arg = np.array(list(map(lambda u: np.arccos(u[0]/norm(u)), us)))
          unit_us = np.stack([np.cos(u_arg), np.sin(u_arg)]).transpose()


          plt.figure(1)
          plt.title('Linear Model Energy')
          plt.grid()
          plt.plot(ts,u_eng,'c')
          plt.ylabel('Energy')
          plt.xlabel('Time')

          plt.figure(2)
          plt.title('Linear Model Phase')
          plt.plot(ts,(180/np.pi)*u_arg, 'c')
          plt.grid()
          plt.ylabel('Angle')
          plt.xlabel('Time')

          plt.figure(3)
          plt.title('U at Various Times')
          origin = [[0],[0]]
          for t,c in [(0,'r'),(5,'b'),(10,'g'),(50,'m')]:
              plt.quiver(*origin, unit_us[t,0], unit_us[t,1], color=c,
                         angles='xy', scale_units='xy', scale=20)
          plt.legend(['t=0','t=5', 't=10', 't=50'])
          plt.grid()
          plt.show()

In [7]: def plotVecs():
          v1_s = [np.array([1,0]),np.array([0,1])]
          v2_s = [np.array([1,0]),np.array([0.9,0.1])]
          origin = [[0],[0]]
          plt.figure(1)
          plt.title('Normal Operator Eigenvectors')
          plt.quiver(*origin, v1_s[0][0], v1_s[0][1], color='r',
                     angles='xy', scale_units='xy', scale=20)
          plt.quiver(*origin, v1_s[1][0], v1_s[1][1], color='b',
                     angles='xy', scale_units='xy', scale=20)

          plt.figure(2)
          plt.title('Non-Normal Operator Eigenvectors')
          plt.quiver(*origin, v2_s[0][0], v1_s[0][1], color='r',
                     angles='xy', scale_units='xy', scale=20)
          plt.quiver(*origin, v2_s[1][0], v1_s[1][1], color='b',
                     angles='xy', scale_units='xy', scale=20)
          plt.show()
```

```
def nonLineModel(init=1e-2):

    R = 10
    A = np.array([[-1/R, 1], [0, -2/R]])
    B = np.array([[0,-1],[1,0]])
    u0 = init*np.array([0.0,1])
    us = [u0]

    dt = 0.005
    nsteps = int(100/dt)

    for t in range(nsteps):
        try:
            du = np.dot(A,us[-1])+norm(us[-1])*np.dot(B, us[-1])
            us.append(us[-1]+dt*du)
        except ValueError:
            print('Crap!')
            break

    u_eng = np.array(list(map(lambda u: norm(u), us)))
    u_arg = np.array(list(map(lambda u: np.arccos(u[0]/norm(u)), us)))
    unit_us = np.stack([np.cos(u_arg), np.sin(u_arg)]).transpose()

    plt.figure(1)
    plt.title('Non-Linear Model Energy')
    plt.plot(np.linspace(0,nsteps*dt,len(u_eng)), u_eng)
    plt.grid()
    plt.ylabel('Energy')
    plt.xlabel('Time')

    plt.figure(2)
    plt.title('Non-Linear Model Phase')
    plt.plot(np.linspace(0,nsteps*dt,len(u_arg)) ,(180/np.pi)*u_arg)
    plt.grid()
    plt.ylabel('Angle')
    plt.xlabel('Time')

    plt.figure(3)
    plt.title('U at Various Times')
    origin = [[0],[0]]
    times = list(map(int, [0, 0.1*nsteps, 0.2*nsteps, 0.5*nsteps]))
    for t,c in zip(times,['r','b','g','m']):
        plt.quiver(*origin, unit_us[t,0], unit_us[t,1], color=c,
                   angles='xy', scale_units='xy', scale=20)
    plt.legend(['t=0','t=10','t=15','t=20','t=50'], loc='lower right')
    plt.grid()
    plt.show()
```

```
In [5]: def nonLineModelInits(inits=[1e-1,1e-2,7e-3,4e-3,1e-3,1e-4]):
            R = 10
            A = np.array([[-1/R, 1], [0, -2/R]])
            B = np.array([[0,-1],[1,0]])

            for init in inits:
                u0 = init*np.array([0.0,1])
                us = [u0]

                dt = 0.005
                nsteps = int(100/dt)

                for t in range(nsteps):
                    try:
                        du = np.dot(A,us[-1])+norm(us[-1])*np.dot(B, us[-1])
                        us.append(us[-1]+dt*du)
                    except ValueError:
                        print('Crap!')
                        break

                u_eng = np.array(list(map(lambda u: norm(u), us)))
                plt.semilogy(np.linspace(0,nsteps*dt,len(u_eng)), u_eng)

            plt.title('Non-Linear Model Energy')
            plt.grid()
            plt.ylabel('Energy')
            plt.xlabel('Time')
            plt.show()

In [74]: #linearGrowthPlots(R = 10)
         #nonLineModel(5e-2)
         #nonLineModelInits()
```


Non-Linear Model Energy
Energy
Time
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
20
40
60
80
100

```
In [ ]:
```