FergusReport

November 17, 2018

1 Singular Value Decomposition

```
In [1]: import os
        import numpy as np
        from matplotlib import pyplot as plt
        import time
        import matplotlib as mpl
        import warnings
        import matplotlib.image as mpimg
        import scipy
        warnings.filterwarnings("ignore", category=np.VisibleDeprecationWarning)
        mpl.rcParams['mathtext.fontset'] = 'cm'
        mpl.rcParams['mathtext.rm'] = 'serif'
        %matplotlib inline
        %config InlineBackend.figure_format = 'svg'
```

This project looks at singular value decompositions (SVD) by first examining standard eigenvalue decomposition and the requirement for an alternative matrix decomposition method.

2 Eigenvalue decomposition

We will start by looking at matrix diagonalisation to understand the meaning a process performed by each matrix.

$$\mathbb{A} = P\Delta P^{-1}$$

Where P is built as the eigenvectors of $\mathbb{A}$ and Δ is a diagonal matrix of the eigenvalues of $\mathbb{A}$. So how is this helpful?

$$P = \begin{pmatrix} \hat{u_1} & \hat{u_2} \end{pmatrix}$$

$$\Delta = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$$

2.0.1 Changing Basis

Consider a vector $\overline{x}$ which can be represented in either the eigenbasis of $\mathbb{A}$ or cartesian unit vectors.

$$\overline{x} = c_1\hat{u_1} + c_2\hat{u_2} = P\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

$$\overline{x} = \alpha\hat{e_1} + \beta\hat{e_2} = \mathbb{I}\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

If we wish to move from the cartesian basis to the eigenbasis we multiply by P^{-1}

$$P^{-1}\mathbb{I}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = P^{-1}P\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

Alternatively, to move from the eigenbasis to the cartesian basis we multiply by P

$$P\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \mathbb{I}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

2.0.2 Function as an operator

We will now consider the action of $\mathbb{A} = P\Delta P^{-1}$ on $\overline{x}$. First we move into the eigenbasis:

$$P^{-1}\overline{x} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

The action of Δ is to multiply each eigenvector $\hat{u_i}$ by its eigenvalue σ_i.

$$\Delta P^{-1}\overline{x} = \Delta\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \sigma_1 c_1\hat{u_1} + \sigma_2 c_2\hat{u_2}$$

Finally, P moves back to the cartesian basis.
With multple actions:

$$\mathbb{A}^n\overline{x} = P\Delta P^{-1}P\Delta P^{-1}\ldots P\Delta P^{-1}\overline{x} = P\Delta^n P^{-1}\overline{x}$$

2.0.3 Normal and non-normal operators

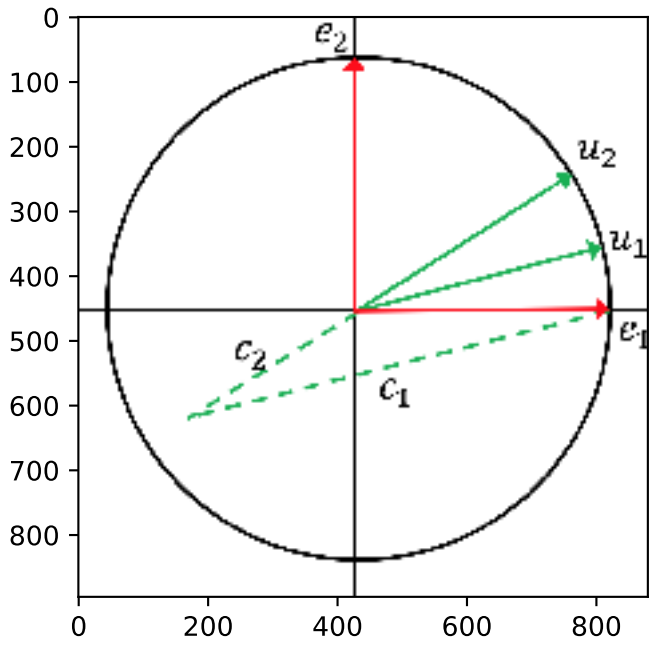
If $\mathbb{A}$ is normal then $\mathbb{A}\mathbb{A}^\dagger = \mathbb{A}^\dagger\mathbb{A}$ and P is unitary with orthonormal columns. This means the vectors $\hat{u_i}$ are orthonormal and:

$$P^{-1} = P^\dagger$$

$$||P|| = ||P^{-1}||$$

For non-normal $\mathbb{A}$, the columns of P are not orthonormal and P^{-1} can have very large elements. This is best imagined by considering the projection of $\hat{e_1}$ or $\hat{e_2}$ onto $\hat{u_1}$ and $\hat{u_2}$.

```
In [2]: img=mpimg.imread('projection.png')
        imgplot = plt.imshow(img)
        plt.show()
```



Above is a diagram showing the projection of $\hat{e_1}$ onto $\hat{u_1}$ and $\hat{u_2}$. This is expressed as

$$\hat{e_1} = c_1\hat{u_1} + c_2\hat{u_2}.$$

The values c_1 and c_2 are entries in the P^{-1} matrix, used to move from the cartesian basis to the eigenbasis. In this case they would appear as

$$P^{-1} = \begin{pmatrix} c_1 & \dots \\ c_2 & \dots \end{pmatrix}.$$

Similarly, we could consider the projection of $\hat{e_2}$ onto $\hat{u_1}$ and $\hat{u_2}$, such that

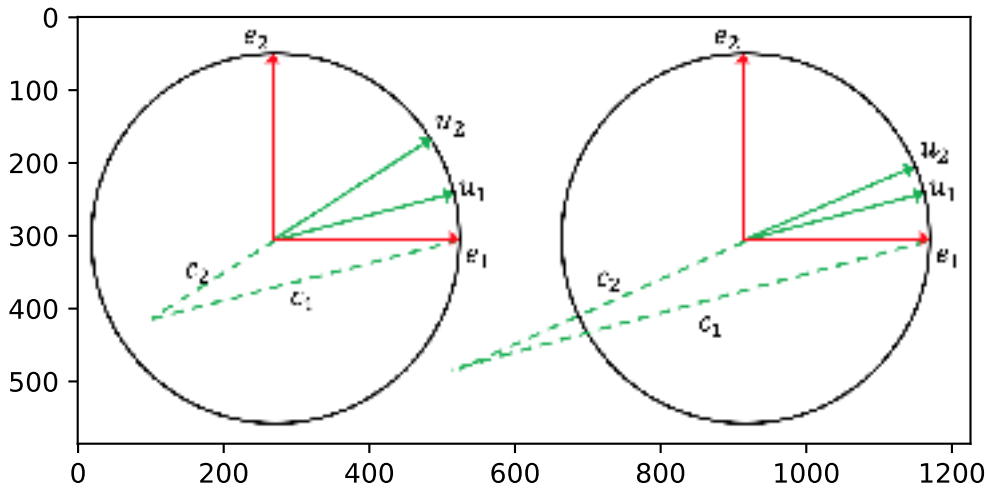
$$\hat{e_2} = c_3\hat{u_1} + c_4\hat{u_2}.$$

This would then give us the entire P^{-1} matrix as

$$P^{-1} = \begin{pmatrix} c_1 & c_3 \\ c_2 & c_4 \end{pmatrix}.$$

As the angle between $\hat{u_1}$ and $\hat{u_2}$ closes, the magnitudes of the entries in P^{-1} can grow very large and sensitive to perturbations in $\mathbb{A}$. Under perturbation a small change in the eigenvalues and eigenvectors of $\mathbb{A}$ will lead to a change in the angle of separation between $\hat{u_1}$ and $\hat{u_2}$. If the angle is small ($\hat{u_1}$ and $\hat{u_2}$ are close to parallel) then there will be significant changes in the entries of P^{-1}, because they are large.

```
In [3]: img=mpimg.imread('SVDimage2.png')
        imgplot = plt.imshow(img)
        plt.show()
```



2.0.4 Perturbations of normal and non-normal operators

To compare the sensitivity of normal and non-normal operators to perturbations, one of each is chosen and the eigenvalues are plotted with small perturbations to $\mathbb{A}$.

The normal operator chosen is:

$$A_{yes} = \begin{pmatrix} -1 & 1 \\ 1 & -2 \end{pmatrix}$$

Decomposing into $P\Delta P^{-1}$

$$P = \begin{pmatrix} \frac{1-\sqrt{5}}{2} & \frac{1+\sqrt{5}}{2} \\ 1 & 1 \end{pmatrix}$$

$$\Delta = \begin{pmatrix} \frac{-3-\sqrt{5}}{2} & 0 \\ 0 & \frac{-3+\sqrt{5}}{2} \end{pmatrix}$$

$$P^{-1} = \begin{pmatrix} \frac{-\sqrt{5}}{5} & \frac{5+\sqrt{5}}{10} \\ \frac{\sqrt{5}}{5} & \frac{5-\sqrt{5}}{10} \end{pmatrix}$$

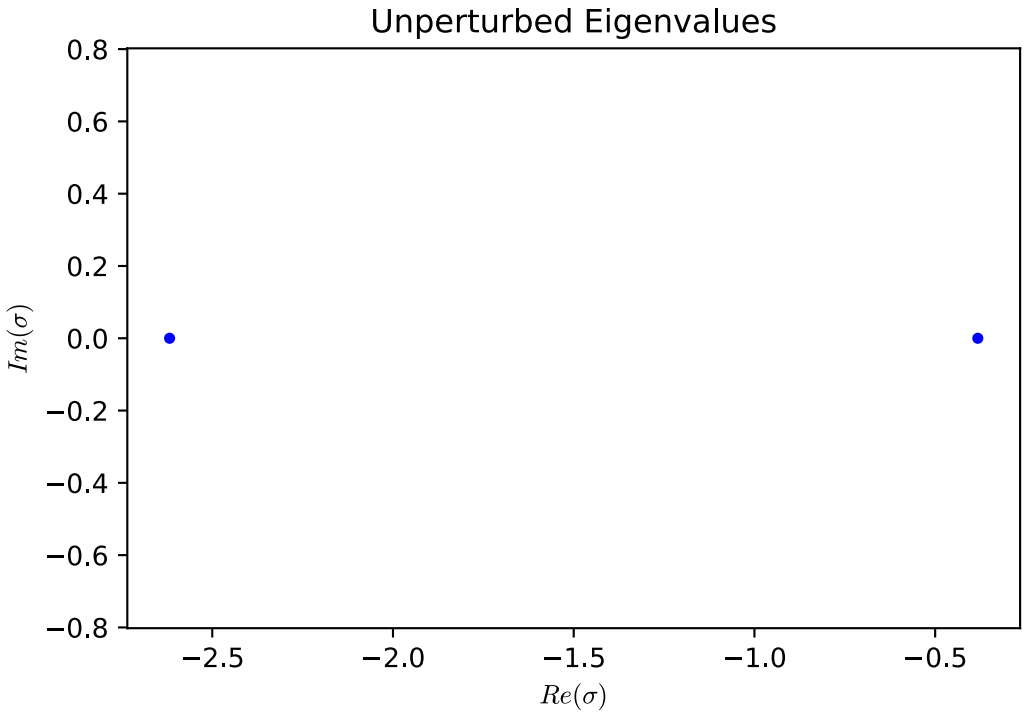
Checking that the eigenvectors are orthogonal and plotting the unperturbed eigenvalues:

$$\sigma_1 = \frac{-3-\sqrt{5}}{2}, \sigma_2 = \frac{-3+\sqrt{5}}{2}$$

```
In [4]: Ayes = [[-1,1],[1,-2]]

In [5]: u1=np.linalg.eig(Ayes)[1][:,0]
        u2=np.linalg.eig(Ayes)[1][:,1]

        np.dot(np.matrix.getH(u1),u2)
```

```
Out[5]: 0.0
```

```
In [6]: sig1=np.linalg.eig(Ayes)[0][0]
        sig2=np.linalg.eig(Ayes)[0][1]

        plt.scatter(np.real([sig1,sig2]),np.imag([sig1,sig2]),c='b',s=10)
        plt.title('Unperturbed Eigenvalues')
        plt.xlabel('$Re(\sigma)$')
        plt.ylabel('$Im(\sigma)$')
        plt.axis('equal')
        plt.show()
```



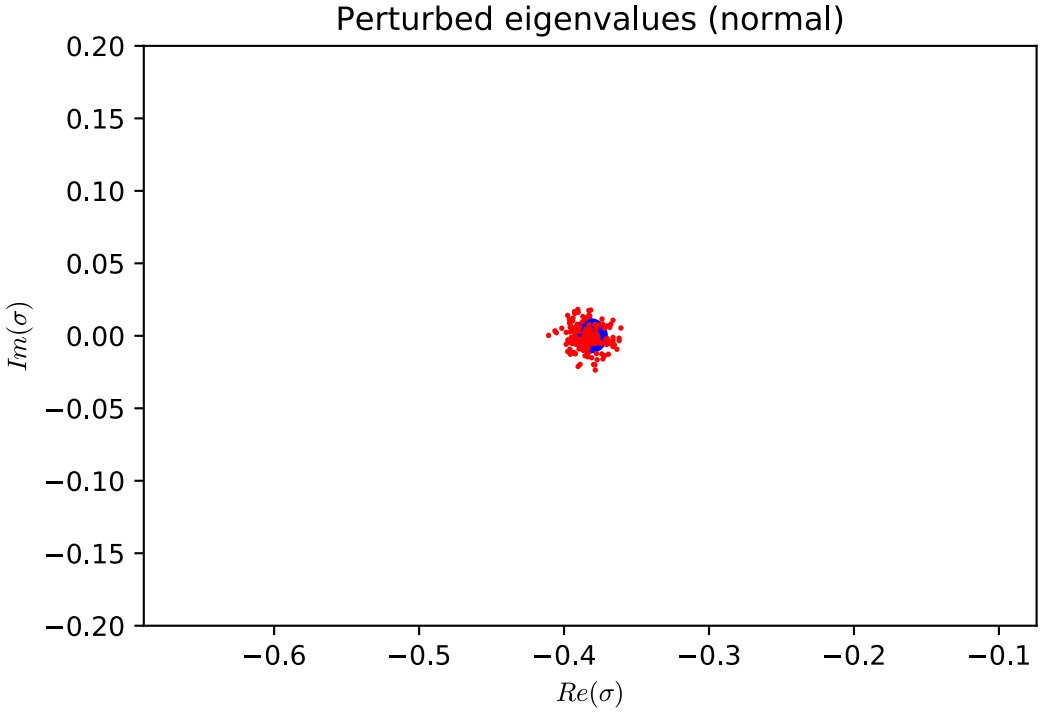
The perturbation experiment will involve multiple perturbations to all terms in $\mathbb{A}$ of order 10^{-2}. The pertrubation size is determined by comparing the norms of $\mathbb{A}$ and the pertrubation matrix $\mathbb{A}'$.

```
In [7]: pertradius=0.01
        print('Norm of A:',np.linalg.norm(Ayes))
        P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)
        print('Norm of pertrubation:',np.linalg.norm([[P[0],P[1]],[P[2],P[3]]]))
        print('Relative norm of pertrubation:',np.linalg.norm([[P[0],P[1]],[P[2],P[3]]])/np.li
```

```
Norm of A: 2.6457513110645907
Norm of pertrubation: 0.027058079301569556
Relative norm of pertrubation: 0.010226992683859615
```

To examine the change in eigenvalues as a result of the perturbations we well "zoom in" on the $\frac{-3+\sqrt{5}}{2}$ eigenvalue. The dispersion of eigenvalues roughly resembles the normal distribution from which the perturbation entries were obtained (mean zero, standard deviation 0.01).

```
In [8]: plt.scatter(np.real([sig1,sig2]),np.imag([sig1,sig2]),c='b',s=150)

        pertradius = 0.01

        for i in range(150):

            P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)

            AyesP = np.add([[-1,1],[1,-2]],[[P[0],P[1]],[P[2],P[3]]])

            sigP1=np.linalg.eig(AyesP)[0][0]
            sigP2=np.linalg.eig(AyesP)[0][1]
            plt.scatter(np.real([sigP1,sigP2]),np.imag([sigP1,sigP2]),c='r',s=1)

        plt.title('Perturbed eigenvalues (normal)')
        plt.xlabel('$Re(\sigma)$')
        plt.ylabel('$Im(\sigma)$')
        plt.axis('equal')
        plt.xlim(np.real(sig1)-0.2, np.real(sig1)+0.2)
        plt.ylim(np.imag(sig1)-0.2, np.imag(sig1)+0.2)
        plt.show()
```



Now we will examine the same experiment for a non-normal matrix:
The non-normal operator chosen is:

$$A_{no} = \begin{pmatrix} -1 & 1 \\ 0 & -2 \end{pmatrix}$$

Decomposing into $U\Delta U^{-1}$

$$P = \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\Delta = \begin{pmatrix} -2 & 0 \\ 0 & -1 \end{pmatrix}$$

$$P^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

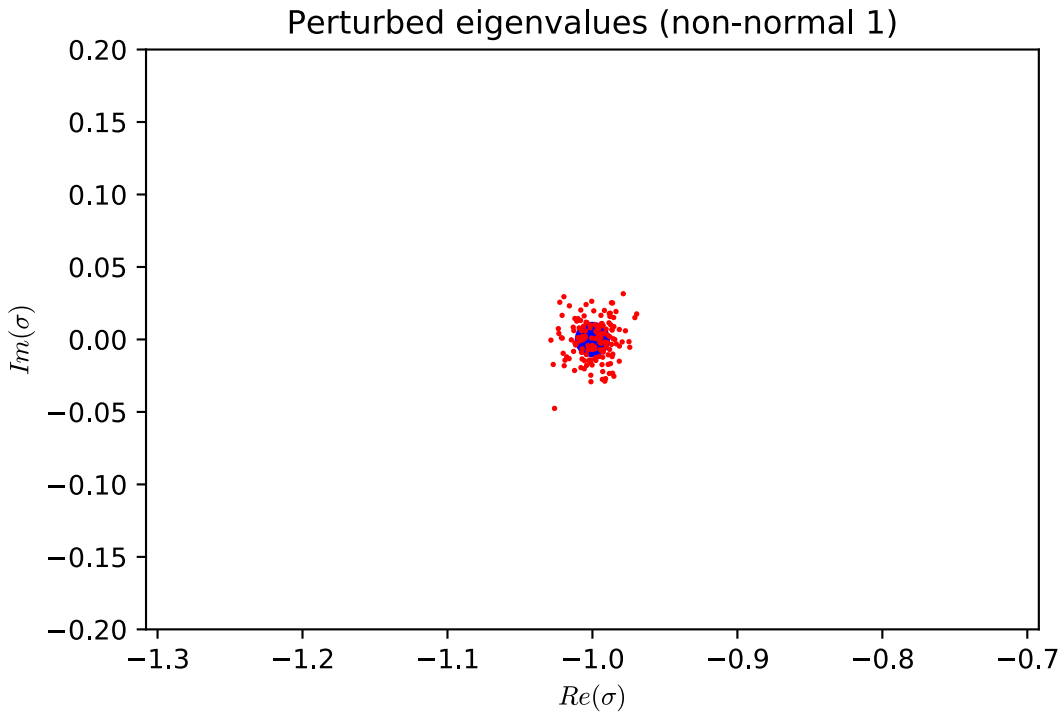
We see that the eigenvectors are not orthogonal with an angle of separation of $3\pi/4$. The perturbation experiment is repeated for this matrix with perturbations of approximately the same relative size. We zoon in on the -1 eigenvalue.

```
In [9]: Ano = [[-1,1],[0,-2]]
        pertradius=0.01
        print('Norm of A:',np.linalg.norm(Ano))
        P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)
        print('Norm of pertrubation:',np.linalg.norm([[P[0],P[1]],[P[2],P[3]]]))
        print('Relative norm of pertrubation:',np.linalg.norm([[P[0],P[1]],[P[2],P[3]]])/np.li
```

```
Norm of A: 2.449489742783178
Norm of pertrubation: 0.019350527179618163
Relative norm of pertrubation: 0.007899819640653633
```

```
In [10]: v1=np.linalg.eig(Ano)[1][:,0]
         v2=np.linalg.eig(Ano)[1][:,1]
         zig1=np.linalg.eig(Ano)[0][0]
         zig2=np.linalg.eig(Ano)[0][1]

         plt.scatter(np.real([zig1,zig2]),np.imag([zig1,zig2]),c='b',s=150)

         for i in range(150):

             P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)

             AnoP = np.add([[-1,1],[0,-2]],[[P[0],P[1]],[P[2],P[3]]])

             zigP1=np.linalg.eig(AnoP)[0][0]
             zigP2=np.linalg.eig(AnoP)[0][1]
             plt.scatter(np.real([zigP1,zigP2]),np.imag([zigP1,zigP2]),c='r',s=1)
```

```
plt.title('Perturbed eigenvalues (non-normal 1)')
plt.xlabel('$Re(\sigma)$')
plt.ylabel('$Im(\sigma)$')
plt.axis('equal')
plt.xlim(np.real(zig1)-0.2, np.real(zig1)+0.2)
plt.ylim(np.imag(zig1)-0.2, np.imag(zig1)+0.2)
plt.show()
```



The distribution about the unperturbed eigenvalues is significantly broader for the non-normal matrix. However, for a more decisive demonstration a second "more non-normal" matrix is used to show significantly greater sensitivity.

$$A_{nooo} = \begin{pmatrix} -1 & 10 \\ 0 & -2 \end{pmatrix}$$

Decomposing into $P\Delta P^{-1}$

$$P = \begin{pmatrix} -10 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\Delta = \begin{pmatrix} -2 & 0 \\ 0 & -1 \end{pmatrix}$$

$$P^{-1} = \begin{pmatrix} 10 & 1 \\ 1 & 0 \end{pmatrix}$$

10 replaces 1 in the upper right entry leading to eigenvectors which are closer to parallel, an angle of separation of approximately 174°.

$$\begin{pmatrix} -10 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

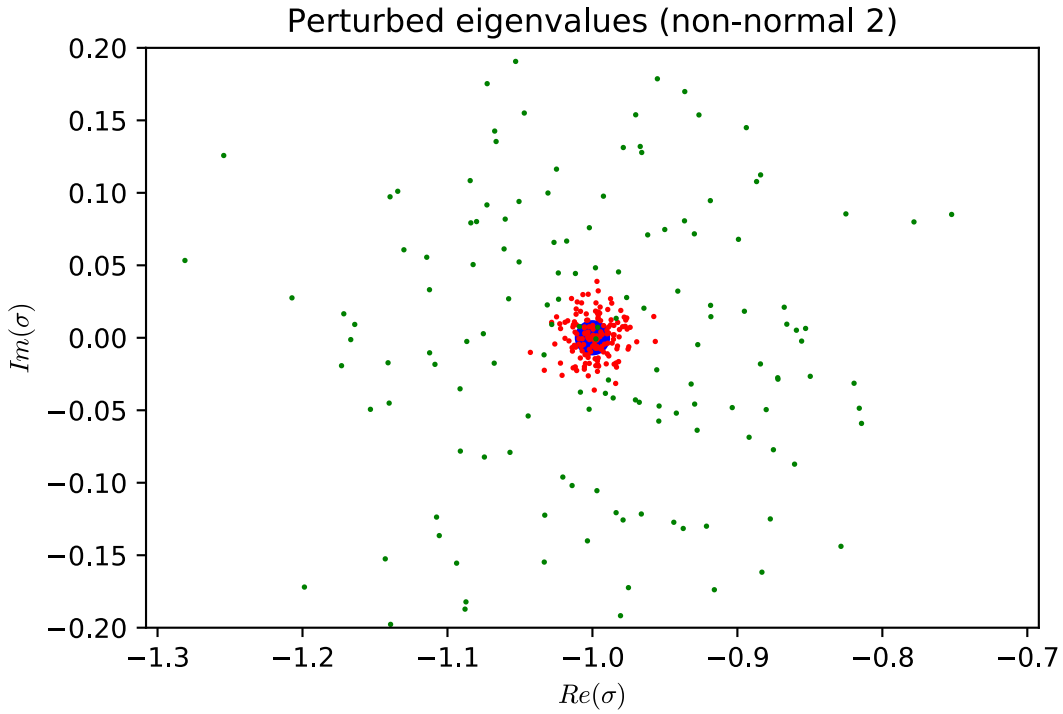
```
In [11]: Ano = [[-1,1],[0,-2]]
         pertradius=0.01
         print('Norm of A:',np.linalg.norm([[-1,10],[0,-2]]))
         P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)
         print('Norm of pertrubation:',np.linalg.norm([[P[0],P[1]],[P[2],P[3]]]))
         print('Relative norm of pertrubation:',np.linalg.norm([[P[0],P[1]],[P[2],P[3]]])/np.li
```

```
Norm of A: 10.246950765959598
Norm of pertrubation: 0.017754223436451944
Relative norm of pertrubation: 0.001732634794677801
```

The eigenvalues of both non-normal matrices so we can compare the impact of perturbation on both. The relative size of the pertrubation on the more non-normal matrix is smaller when compared by matrix norms.

```
In [12]: plt.scatter(np.real([zig1,zig2]),np.imag([zig1,zig2]),c='b',s=150)

         for i in range(150):

             P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)

             AnoP = np.add([[-1,1],[0,-2]],[[P[0],P[1]],[P[2],P[3]]])

             zigP1=np.linalg.eig(AnoP)[0][0]
             zigP2=np.linalg.eig(AnoP)[0][1]
             plt.scatter(np.real([zigP1,zigP2]),np.imag([zigP1,zigP2]),c='r',s=1)

         for i in range(150):

             P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)

             AnoP = np.add([[-1,10],[0,-2]],[[P[0],P[1]],[P[2],P[3]]])

             zigP1=np.linalg.eig(AnoP)[0][0]
             zigP2=np.linalg.eig(AnoP)[0][1]
             plt.scatter(np.real([zigP1,zigP2]),np.imag([zigP1,zigP2]),c='g',s=1)

         plt.title('Perturbed eigenvalues (non-normal 2)')
         plt.xlabel('$Re(\sigma)$')
         plt.ylabel('$Im(\sigma)$')
         plt.axis('equal')
         plt.xlim(np.real(zig1)-0.2, np.real(zig1)+0.2)
```

```
plt.ylim(np.imag(zig1)-0.2, np.imag(zig1)+0.2)
plt.show()
```



The far broader distribution of pertrubed eigenvalues from the non-normal matrix (in green) demonstrate the increased sensitivity as eigenvectors move towards parallel. For this reason it is important that we are able to decompose non-normal operators into an orthonormal basis.

3 SVD - Singular Value Decomposition

SVD is a decomposition of $\mathbb{A}$ which ensures orthonormal columns in the "equivalent" P matrix so that there is reduced sensitivity to perturbation in non-normal matrices.

$$\mathbb{A} = U\Sigma V^\dagger$$

Where U is built from the eigenvectors of $\mathbb{A}\mathbb{A}^\dagger$ and $V^\dagger$ is the conjugate transpose of V, which is built from the eigenvectors of $\mathbb{A}^\dagger\mathbb{A}$. Σ is a diagonal matrix of the square roots of eigenvalues for $\mathbb{A}\mathbb{A}^\dagger$ or $\mathbb{A}^\dagger\mathbb{A}$. These are the singular values in the decomposition of $\mathbb{A}$.

3.1 Properties and construction

3.1.1 1

(This follows a derivation produced in the thesis appendicies shared by Navid). $\mathbb{A}\mathbb{A}^\dagger$ and $\mathbb{A}^\dagger\mathbb{A}$ are hermitian. We can show that they are self-adjoint and positive definite with real, positive eigenvalues.

$$(\mathbb{A}^\dagger\mathbb{A})^\dagger = \mathbb{A}^\dagger(\mathbb{A}^\dagger)^\dagger = \mathbb{A}^\dagger\mathbb{A}.$$

$$\mathbb{A}^\dagger\mathbb{A}\hat{u}_i = \lambda_i\hat{u}_i(\mathbb{A}\hat{u}_i)^\dagger\mathbb{A}\hat{u}_i > 0$$

and

$$(\mathbb{A}\hat{u}_i)^\dagger\mathbb{A}\hat{u}_i = \hat{u}_i{}^\dagger\mathbb{A}^\dagger\mathbb{A}\hat{u}_i = \lambda_i\hat{u}_i{}^\dagger\hat{u}_i$$

Since $\hat{u}_i{}^\dagger\hat{u}_i > 0$, we then have that $\lambda_i > 0$.

The same method can be used for $\mathbb{A}\mathbb{A}^\dagger$. Given that $\mathbb{A}\mathbb{A}^\dagger$ and $\mathbb{A}^\dagger\mathbb{A}$ are hermitian, we then have that the columns of U and V are orthonormal, making them unitary matrices.

3.1.2 2

Additionally, we can define the hermitian operator $\mathbb{H} = (\mathbb{A}^\dagger\mathbb{A})^{1/2}$ as

$$\mathbb{H} = V\sqrt{S}V^\dagger.$$

Where S is the diagonal matrix with the eigenvalues of $\mathbb{A}^\dagger\mathbb{A}$ on its diagonal.

We then have that $= \mathbb{A}\mathbb{H}^{-1}$ is unitary since:

$$^\dagger = (\mathbb{A}\mathbb{H}^{-1})^\dagger\mathbb{A}\mathbb{H}^{-1} = \mathbb{H}^{-1}\mathbb{A}^\dagger\mathbb{A}\mathbb{H}^{-1} = \mathbb{H}^{-1}\mathbb{H}^2\mathbb{H}^{-1} = \mathbb{I}$$

$$^\dagger = \mathbb{A}\mathbb{H}^{-1}(\mathbb{A}\mathbb{H}^{-1})^\dagger = \mathbb{A}\mathbb{H}^{-1}\mathbb{H}^{-1}\mathbb{A}^\dagger = \mathbb{A}(\mathbb{A}^\dagger\mathbb{A})^{-1}\mathbb{A}^\dagger = \mathbb{I}$$

As a result, every operator $\mathbb{A}$ can be written as $\mathbb{A} = \mathbb{H}$. This is a polar decomposition of the operator $\mathbb{A}$.

3.1.3 3

Now if we let Σ be the diagonal matrix with elements the square roots of the elements of S we can rewrite $\mathbb{A}$ as

$$\mathbb{A} = V\Sigma V^\dagger$$

Since is unitary, if $\hat{v}_i$ and $\hat{v}_j$ are two orthogonal vectors then so are $\hat{v}_i$ and $\hat{v}_j$. Therefore V is unitary.

Now if we return to the operator $\mathbb{A}\mathbb{A}^\dagger$,

$$\mathbb{A}\mathbb{A}^\dagger = \mathbb{H}(\mathbb{H})^\dagger = (V\sqrt{S}V^\dagger)(V^\dagger\sqrt{S}V)^\dagger = (V)S(V)^\dagger$$

This gives that the unitary matrix $U = V$ is the matrix whose columns are the eigenvectors of $\mathbb{A}\mathbb{A}^\dagger$ and gives the final result that any operator $\mathbb{A}$ can be decomposed as $\mathbb{A} = U\Sigma V^\dagger$. Below is a computation of the components of the decomposition for the first non-normal matrix examined.

```
In [13]: Ano = [[-1,1],[0,-2]]
         print('A:',Ano)
         print('Decomposition:')
         print('U:',np.linalg.svd(Ano)[0])
         print('Sigma:',np.diag(np.linalg.svd(Ano)[1]))
         print('V(dagger):',np.linalg.svd(Ano)[2])
```

```
        print('U*Sigma*V(dagger):',np.linalg.svd(Ano)[0]@np.diag(np.linalg.svd(Ano)[1])@np.li
        print('-----')
        print('V:',np.matrix.getH(np.linalg.svd(Ano)[2]))
        print('H:',np.matrix.getH(np.linalg.svd(Ano)[2])@np.diag(np.linalg.svd(Ano)[1])@np.li
        print('Phi:', Ano @ np.linalg.inv(np.matrix.getH(np.linalg.svd(Ano)[2])@np.diag(np.li
        print('Phi*V:',Ano @ np.linalg.inv(np.matrix.getH(np.linalg.svd(Ano)[2])@np.diag(np.l
```

```
A: [[-1, 1], [0, -2]]
Decomposition:
U: [[-0.52573111 -0.85065081]
 [ 0.85065081 -0.52573111]]
Sigma: [[2.28824561 0.        ]
 [0.         0.87403205]]
V(dagger): [[ 0.22975292 -0.97324899]
 [ 0.97324899  0.22975292]]
U*Sigma*V(dagger): [[-1.00000000e+00  1.00000000e+00]
 [ 1.29356799e-16 -2.00000000e+00]]
-----
V: [[ 0.22975292  0.97324899]
 [-0.97324899  0.22975292]]
H: [[ 0.9486833  -0.31622777]
 [-0.31622777  2.21359436]]
Phi: [[-0.9486833   0.31622777]
 [-0.31622777 -0.9486833 ]]
Phi*V: [[-0.52573111 -0.85065081]
 [ 0.85065081 -0.52573111]]
```

3.2 Stability

The previous experiment, involving the matrices

$$\begin{pmatrix} -1 & 1 \\ 0 & -2 \end{pmatrix}$$

and

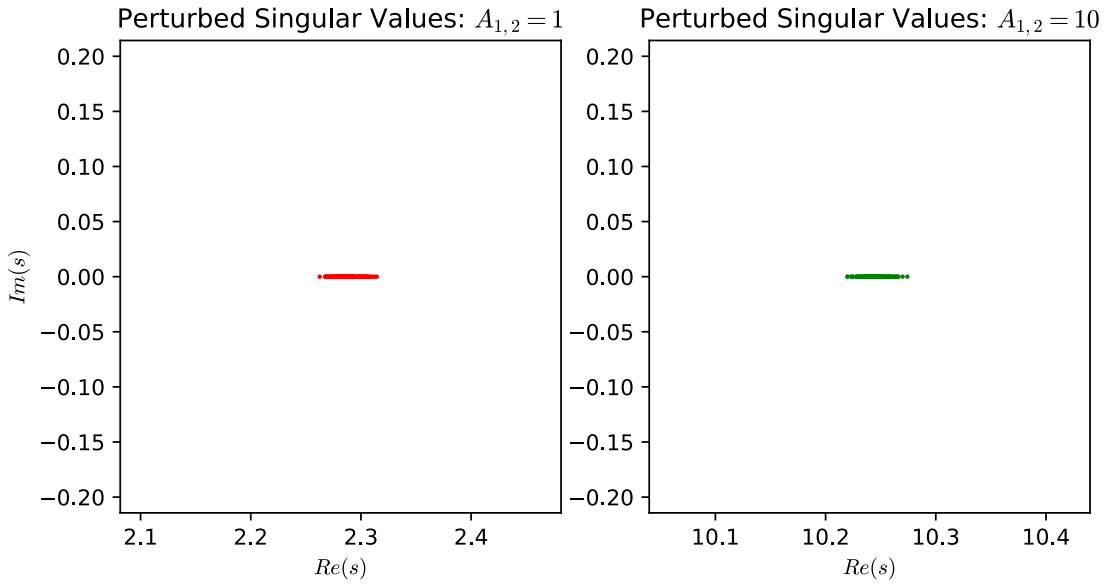
$$\begin{pmatrix} -1 & 10 \\ 0 & -2 \end{pmatrix}$$

has repeated using a singular value decomposition rather than eigenvalue decomposition. It is important to note here that it is the singular values (the diagonal of Σ) which are of interest as the when examining the effect of the pertrubation.

The plots below show the distribution of perturbed singular values for both non-normal operators $\mathbb{A}$ that were previously examined. It can be seen that the spreads are essentially identical - regardless of how "normal" each operator is. This is because SVD involves decomposition with an orthonormal basis. The singular values remain purely real and positive.

```
In [14]: #plt.scatter(np.real([zig1,zig2]),np.imag([zig1,zig2]),c='b',s=150)
         plt.figure(figsize=(8,4))
```

```
pertradius=0.01
plt.subplot(1,2,1)
for i in range(150):

    P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)

    AnoP = np.add([[-1,1],[0,-2]],[[P[0],P[1]],[P[2],P[3]]])

    U=np.linalg.svd(AnoP)[0]
    S=np.linalg.svd(AnoP)[1]
    Vdagger=np.linalg.svd(AnoP)[2]

    zigP1=S[0]
    zigP2=S[1]
    plt.scatter(np.real([zigP1,zigP2]),np.imag([zigP1,zigP2]),c='r',s=1)
plt.xlabel('$Re(s)$')
plt.ylabel('$Im(s)$')
plt.axis('equal')
plt.title(r'Perturbed Singular Values: $A_{1,2}=1$')
plt.xlim(np.real(zigP1)-0.2, np.real(zigP1)+0.2)
plt.ylim(np.imag(zigP1)-0.2, np.imag(zigP1)+0.2)


plt.subplot(1,2,2)
for i in range(150):

    P = pertradius*(np.random.randn(4)+np.random.randn(4)*1j)

    AnoP = np.add([[-1,10],[0,-2]],[[P[0],P[1]],[P[2],P[3]]])

    U=np.linalg.svd(AnoP)[0]
    S=np.linalg.svd(AnoP)[1]
    Vdagger=np.linalg.svd(AnoP)[2]

    zigP1=S[0]
    zigP2=S[1]
    plt.scatter(np.real([zigP1,zigP2]),np.imag([zigP1,zigP2]),c='g',s=1)

plt.title(r'Perturbed Singular Values: $A_{1,2}=10$')
plt.xlabel('$Re(s)$')
#plt.ylabel('$Im(\sigma)$')
plt.axis('equal')
plt.xlim(np.real(zigP1)-0.2, np.real(zigP1)+0.2)
plt.ylim(np.imag(zigP1)-0.02, np.imag(zigP1)+0.02)
plt.show()
```



4 SVD - excitation of a linear system

Through the fluid instabilities course we have examined linear systems and often the optimal excitation of a linear system. The singular value decomposition of an operator presents an intuitive way to find the state which optimally excites a system.

The diagonal values in Σ (the singular values) are all positive and ordered such that the first entry is the largest. To achieve optimal excitation we wish to isolate the largest singular value $\Sigma_{1,1}$,

$$U\Sigma \begin{pmatrix} 1 \\ \vdots \end{pmatrix}.$$

Since the columns of V are orthonormal

$$V^\dagger \hat{v_1} = \begin{pmatrix} 1 \\ \vdots \end{pmatrix}.$$

We then have

$$U\Sigma V^\dagger \hat{v_1} = U\Sigma_{1,1} \begin{pmatrix} 1 \\ \vdots \end{pmatrix} = \Sigma_{1,1}\hat{u_1}.$$

All $\Sigma_{i,i} > 0$, and any initial state can be represented as a linear combination of $\hat{v_i}$'s. Since all $\hat{v_i}$ are orthonormal and $\Sigma_{1,1}$ is the largest singular value, this gives the initial condition for optimal excitation of our system as $\hat{v_1}$. The final state under optimal excitation is then $\hat{u_1}$ at some magnitude.

4.0.1 Approximation

SVD provides an orthonormal basis as well as a size-ordered set of values which can be utilised in approximation of data or a system. Taking only the first n largest singular values and setting the additional ones to zero (call this $\Sigma^{(n)}$) allows for the computation of an approximation to the operator $\mathbb{A}$.

$$\mathbb{A}^{(n)} = U\Sigma^{(n)}V^{\dagger}$$

In the context of a linear system this isolates the most significant states in the system, or the perturbations which will grow the fastest and are of the most interest.

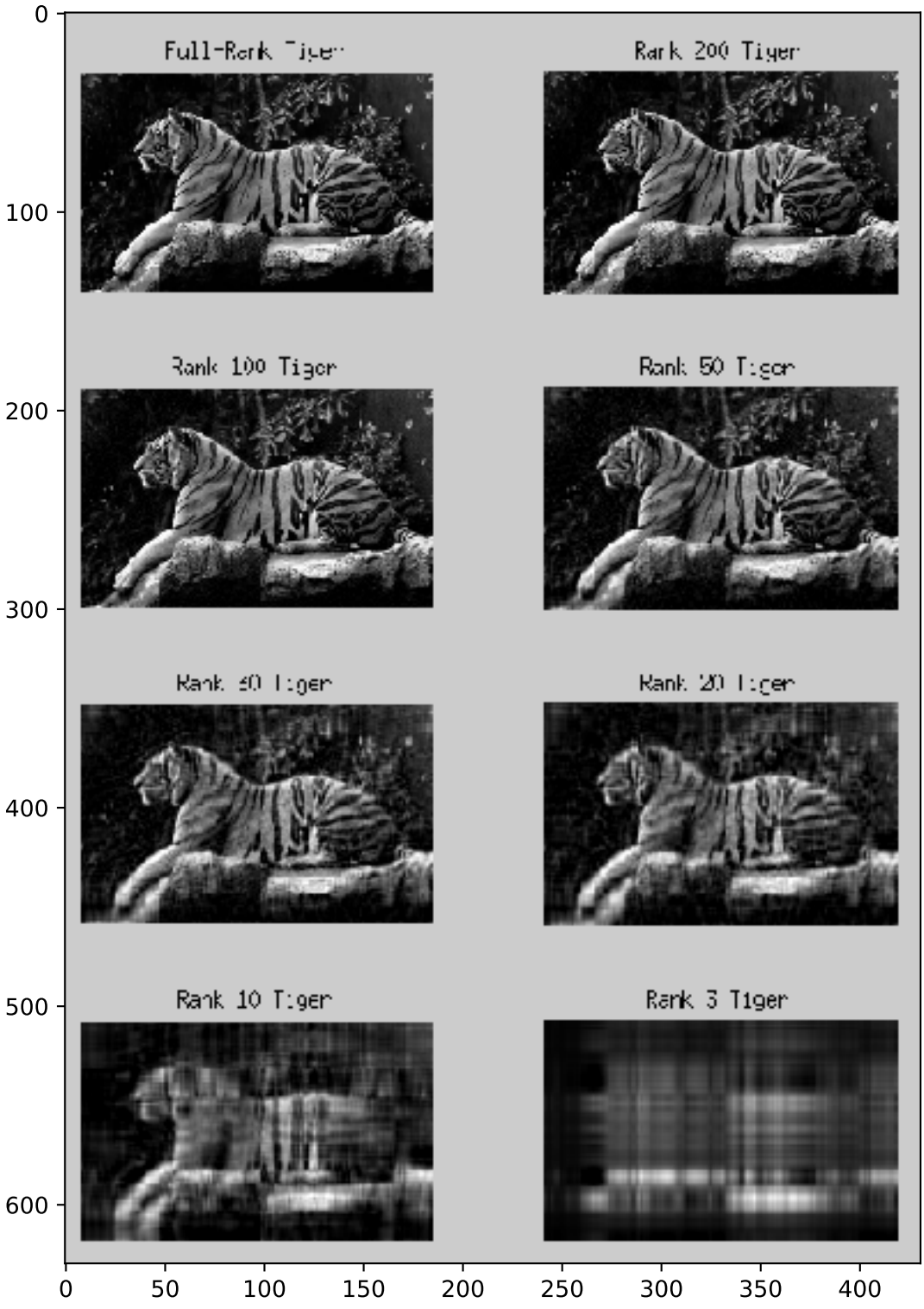
With a focus on computation time and data compression the same approach can be taken. The percentage of information retained with such approximation is caclulated as

$$\frac{\sum_{i=1}^{n} \Sigma_{i,i}}{\sum_{i} \Sigma_{i,i}}$$

This is displayed in the following black and white image of a tiger - the rank refers to n, the number of singular values not set to zero (taken from http://andrew.gibiansky.com/blog/mathematics/cool-linear-algebra-singular-value-decomposition/)

There is little resolution difference at high n and only rapid degeneration at low n. This allows for approximation of very large data sets to reduce memory and computation time constraints.

```
In [15]: plt.rcParams.update({'figure.figsize': (10,10)})
         img=mpimg.imread('tiger.png')
         imgplot = plt.imshow(img)
         plt.show()
```



SVD allows for the decomposition of non-normal matrices into an orthogonal basis, reducing sensitivity to perturbation when compared with standard eigenvalue decomposition. It also

allows for the identification of optimal initial states for excitation of a linear system and their associated final states. The consequences of these properties include applications in approximation of matrices to capture the most important information present and reduce computational and memory constraints.