Question 1. Linear Dynamical System.

We have the following linear dynamical system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{s}(t)$$

where

$$\mathbf{A} = \begin{pmatrix} -0.5 & -0.5 & 0 \\ -0.5 & -0.5 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

1.1 Assuming $\mathbf{s}(t) = 0$, we need to compute and sketch the solutions for the initial conditions

$$\mathbf{x}_{0,1} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{x}_{0,2} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}_{0,3} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{x}_{0,4} = \begin{pmatrix} 0 \\ 0 \\ 10^{-6} \end{pmatrix}$$

Note that the initial value problem has a solution of the form

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0$$

For non-singular $\mathbf{A}$, we can rewrite it as follows

$$\mathbf{x}(t) = \mathbf{Q}e^{\mathbf{\Lambda}t}\mathbf{Q}^{-1}\mathbf{x}_0$$

where $\mathbf{Q}$ is a matrix of eigenvectors and $\mathbf{\Lambda}$ is a daigonal matrix with non-zero entries representing corresponding eigenvalues. To solve this, we have to first find the eigenvalues and eigenvectors of $\mathbf{A}$. Hence, we start by solving the characteristic equation

$$\det(\mathbf{A} - \mathbf{I}\lambda) = 0$$

$$\begin{vmatrix} -0.5-\lambda & -0.5 & 0 \\ -0.5 & -0.5-\lambda & 0 \\ 0 & 0 & 2-\lambda \end{vmatrix} = 0 \quad \Longleftrightarrow \quad -\lambda^3 + \lambda^2 + 2\lambda = 0$$

We find that $\mathbf{A}$ has three distinct eigenvalues, namely, $\lambda_1 = 0$, $\lambda_2 = 2$, $\lambda_3 = -1$, with the corresponding eigenvectors

$$\mathbf{v}_1 = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{v}_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{v}_3 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Now, we have

$$\mathbf{Q} = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{\Lambda} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

and we can thus write our solution as follows

$$\mathbf{x}(t) = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{2t} & 0 \\ 0 & 0 & e^{-t} \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}^{-1} \mathbf{x}_0$$

$$\implies \mathbf{x}(t) = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} -0.5 & 0.5 & 0 \\ 0 & 0 & e^{2t} \\ 0.5e^{-t} & 0 & 0 \end{pmatrix} \mathbf{x}_0$$

$$\implies \mathbf{x}(t) = \begin{pmatrix} 0.5 + 0.5e^{-t} & -0.5 & 0 \\ -0.5 + 0.5e^{-t} & 0.5 & 0 \\ 0 & 0 & e^{2t} \end{pmatrix} \mathbf{x}_0$$

And in a non-matrix form this becomes

$$\mathbf{x}(t) = \begin{pmatrix} (0.5 + 0.5e^{-t})x_1 - 0.5x_2 \\ (-0.5 + 0.5e^{-t})x_1 + 0.5x_2 \\ e^{2t}x_3 \end{pmatrix}$$

Finally, for the aforementioned initial conditions, we have

$$\mathbf{x}_{0,1}(t) = \begin{pmatrix} 0.5 + 0.5e^{-t} & -0.5 & 0 \\ -0.5 + 0.5e^{-t} & 0.5 & 0 \\ 0 & 0 & e^{2t} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.5e^{-t} \\ 0.5e^{-t} \\ 0 \end{pmatrix}$$

$$\mathbf{x}_{0,2}(t) = \begin{pmatrix} 0.5 + 0.5e^{-t} & -0.5 & 0 \\ -0.5 + 0.5e^{-t} & 0.5 & 0 \\ 0 & 0 & e^{2t} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.5 + 0.5e^{-t} \\ -0.5 + 0.5e^{-t} \\ 0 \end{pmatrix}$$

$$\mathbf{x}_{0,3}(t) = \begin{pmatrix} 0.5 + 0.5e^{-t} & -0.5 & 0 \\ -0.5 + 0.5e^{-t} & 0.5 & 0 \\ 0 & 0 & e^{2t} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.5 \\ 0.5 \\ 0 \end{pmatrix}$$

$$\mathbf{x}_{0,4}(t) = \begin{pmatrix} 0.5 + 0.5e^{-t} & -0.5 & 0 \\ -0.5 + 0.5e^{-t} & 0.5 & 0 \\ 0 & 0 & e^{2t} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 10^{-6} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 10^{-6}e^{t} \end{pmatrix}$$

And the phase diagrams for these initial conditions appear in Figures 1-4.

1.2 The eigenvalues and eigenvectors of $\mathbf{A}$ were computed in 1.1. Eigenvalue sign tells us about the direction and speed of the dynamics evolution, and the eigenvectors tell us in which direction the flow will be linear, i.e. only affected by the dynamics matrix as a scalar multiplication.

1.3 The projections of $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ appear in Figures 5-7. The dynamics observed in Figure 5 can be explained using the fact that the eigenvalue associated with eigenvector $\mathbf{v}_1$ is zero. Therefore, we see an invariant manifold along the direction of this eigenvector, with particles attracted towards it. As for Figures 6 and 7, there is a saddle point at $(0, 0)$, for there is one stable and one unstable direction.

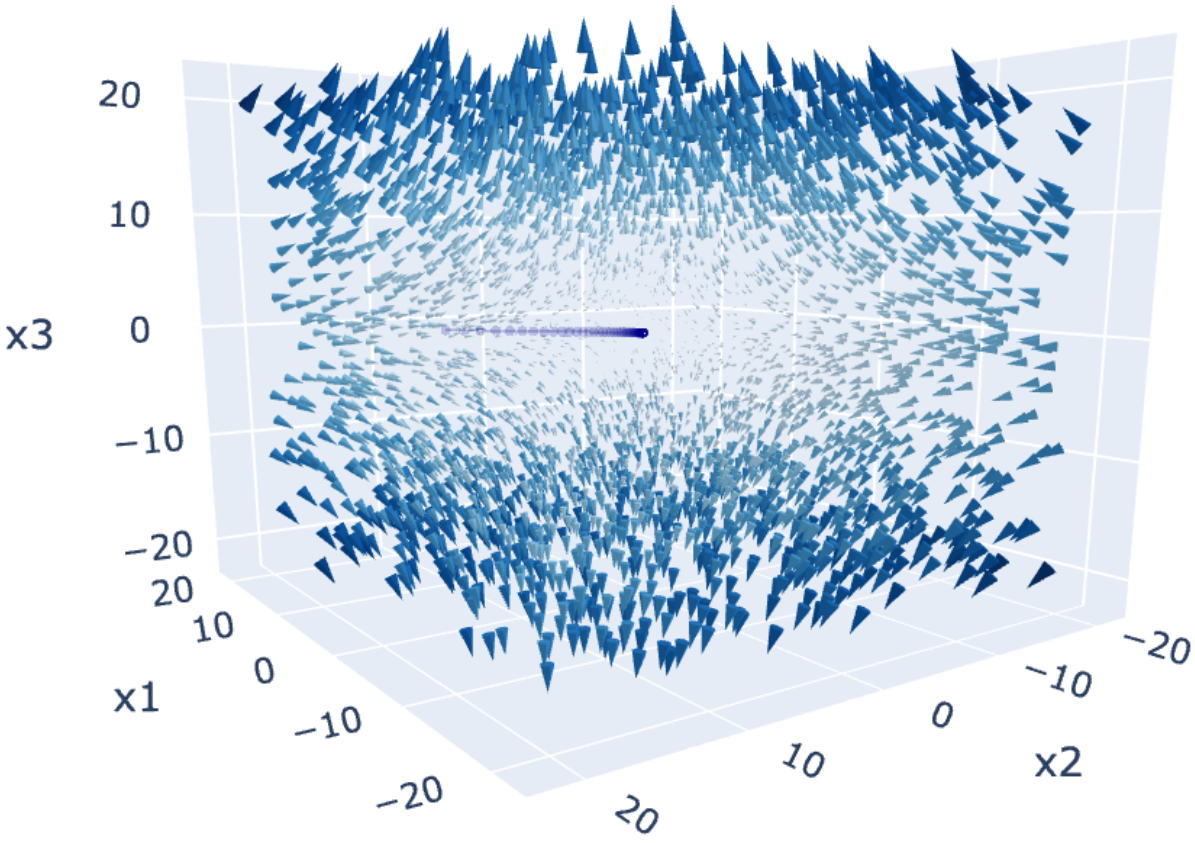

Figure 1: 3D dynamics of $\mathbf{x}(t)$ following the initial condition $\mathbf{x}_{0,1}$ for $t \in [-3, 3]$.

1.4 Now we want to project the vector field of the dynamics onto the subspace defined by the basis

$$\mathbf{e}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad \mathbf{e}_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Recall the formula for projecting vector $\mathbf{x}$ onto a line spanned by vector $\mathbf{u}$

$$\text{proj}_{\mathbf{u}}(\mathbf{x}) = \frac{\mathbf{u} \cdot \mathbf{x}}{\|\mathbf{u}\|^2} \mathbf{u}$$

Therefore, to project vector $\mathbf{x}$ onto a subspace V spanned by $\mathbf{e}_1$ and $\mathbf{e}_2$, we need to do the following

$$\text{proj}_V(\mathbf{x}) = \frac{\mathbf{e}_1 \cdot \mathbf{x}}{\|\mathbf{e}_1\|^2} \mathbf{e}_1 + \frac{\mathbf{e}_2 \cdot \mathbf{x}}{\|\mathbf{e}_2\|^2} \mathbf{e}_2$$

And since $\mathbf{e}_1$ and $\mathbf{e}_2$ form the basis, we know they are unit vectors, and hence we are left with

$$\text{proj}_V(\mathbf{x}) = (\mathbf{e}_1 \cdot \mathbf{x})\, \mathbf{e}_1 + (\mathbf{e}_2 \cdot \mathbf{x})\, \mathbf{e}_2$$

The eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are mapped onto themselves by this projection, and the eigenvector $\mathbf{v}_3$ is mapped to $\mathbf{0}$, and hence we see an unstable manifold.

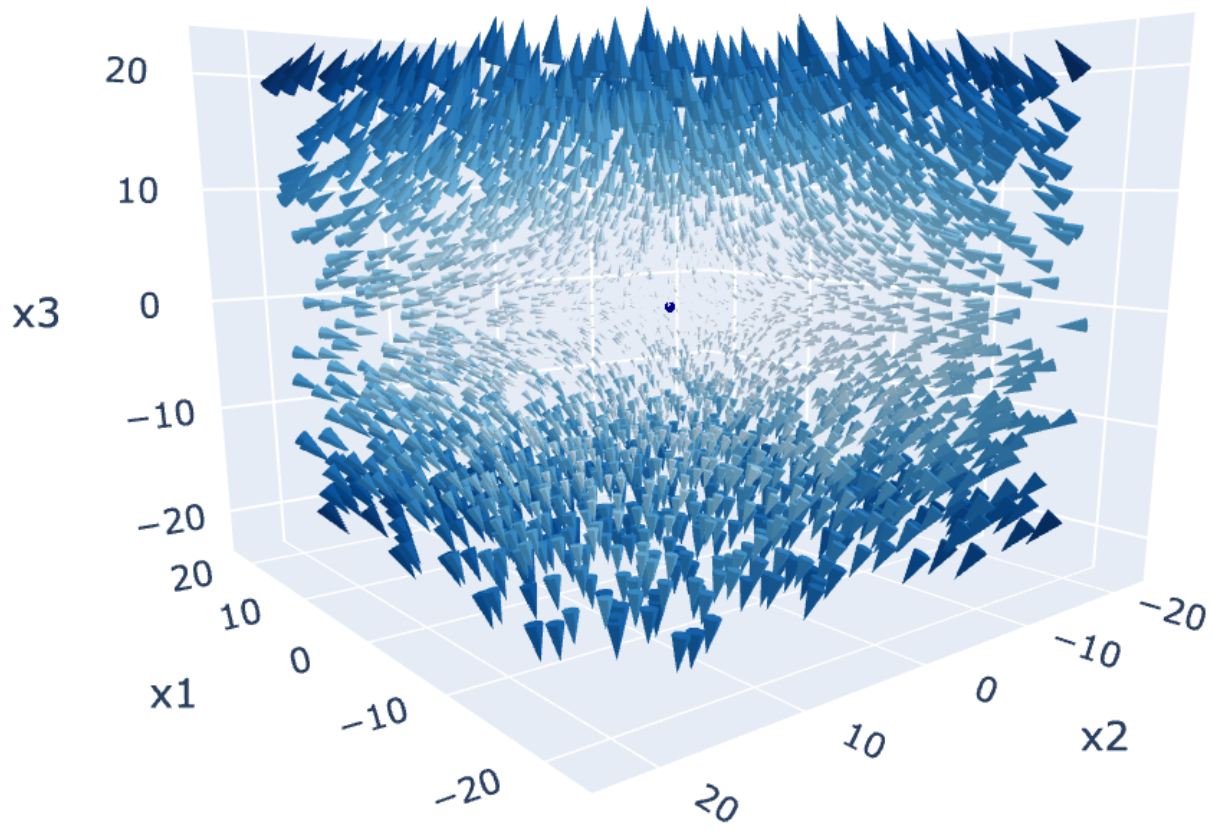

Figure 2: 3D dynamics of $\mathbf{x}(t)$ following the initial condition $\mathbf{x}_{0,2}$.

1.5 Now we want to compute the stationary solution for $(t \to \infty)$ with a constant input $\mathbf{s} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$ and the initial condition $\mathbf{x}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$. To find the solution, we will use the undetermined coefficients methods to solve this system of inhomogeneous DEQs. Recall that our general solution to the homogeneous system was

$$\mathbf{x}(t) = c_1\mathbf{v}_1 e^{\lambda_1 t} + c_2\mathbf{v}_2 e^{\lambda_2 t} + c_3\mathbf{v}_3 e^{\lambda_3 t}$$

We need to find a particular solution $\mathbf{x}_p(t)$, so let

$$\mathbf{x}_p(t) = \begin{pmatrix} x_1 t + b_1 \\ x_2 t + b_2 \\ 0 \end{pmatrix}, \qquad \text{(since our input is constant)}$$

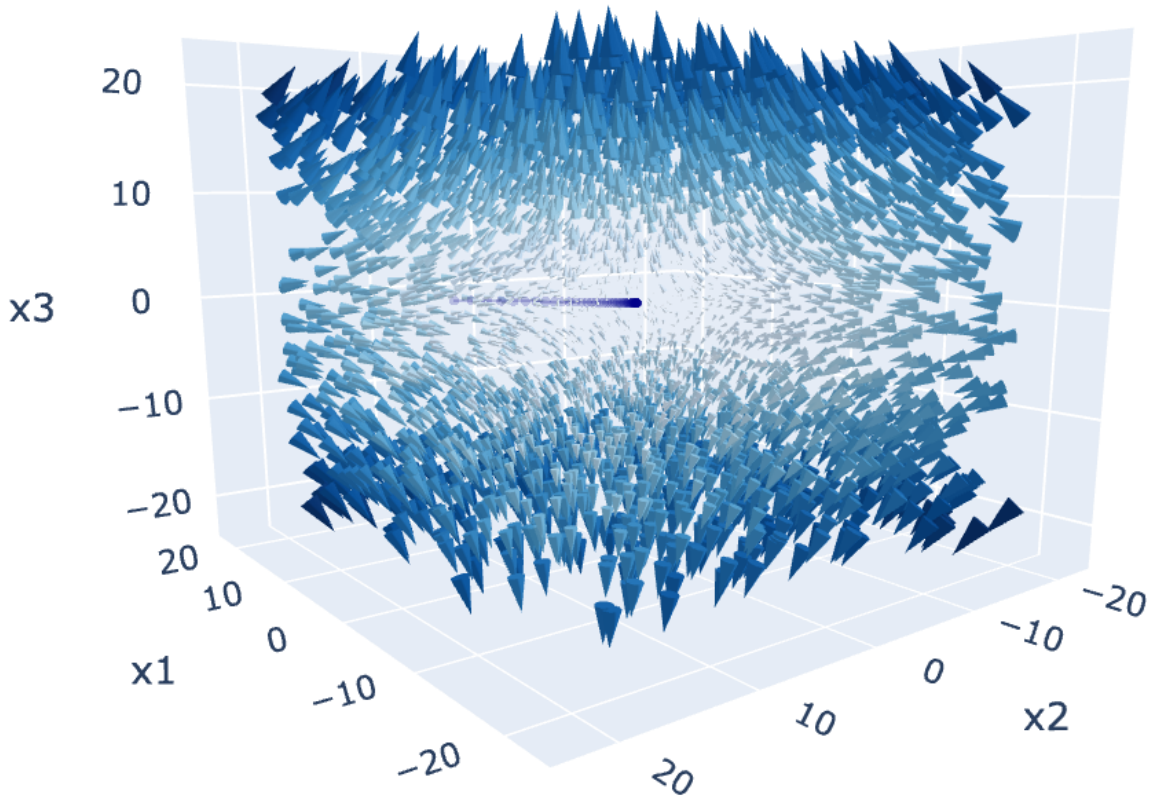

Figure 3: 3D dynamics of $\mathbf{x}(t)$ following the initial condition $\mathbf{x}_{0,3}$ for $t \in [-3, 3]$.

And now we plug this into our orginal system to find x_1, x_2, and x_3

$$\begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \mathbf{A} \begin{pmatrix} x_1 t + b_1 \\ x_2 t + b_2 \\ 0 \end{pmatrix} + \mathbf{s}$$

$$\implies \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.5 & -0.5 & 0 \\ -0.5 & -0.5 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 t + b_1 \\ x_2 t + b_2 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

$$\implies \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.5(x_1 t + b_1) - 0.5(x_2 t + b_2) + 1 \\ -0.5(x_1 t + b_1) - 0.5(x_2 t + b_2) + 2 \\ 0 \end{pmatrix}$$

$$\begin{aligned} x_1 &= -0.5(x_1 t + b_1) - 0.5(x_2 t + b_2) + 1 \\ \implies x_2 &= -0.5(x_1 t + b_1) - 0.5(x_2 t + b_2) + 2 \\ 0 &= 0 \end{aligned}$$

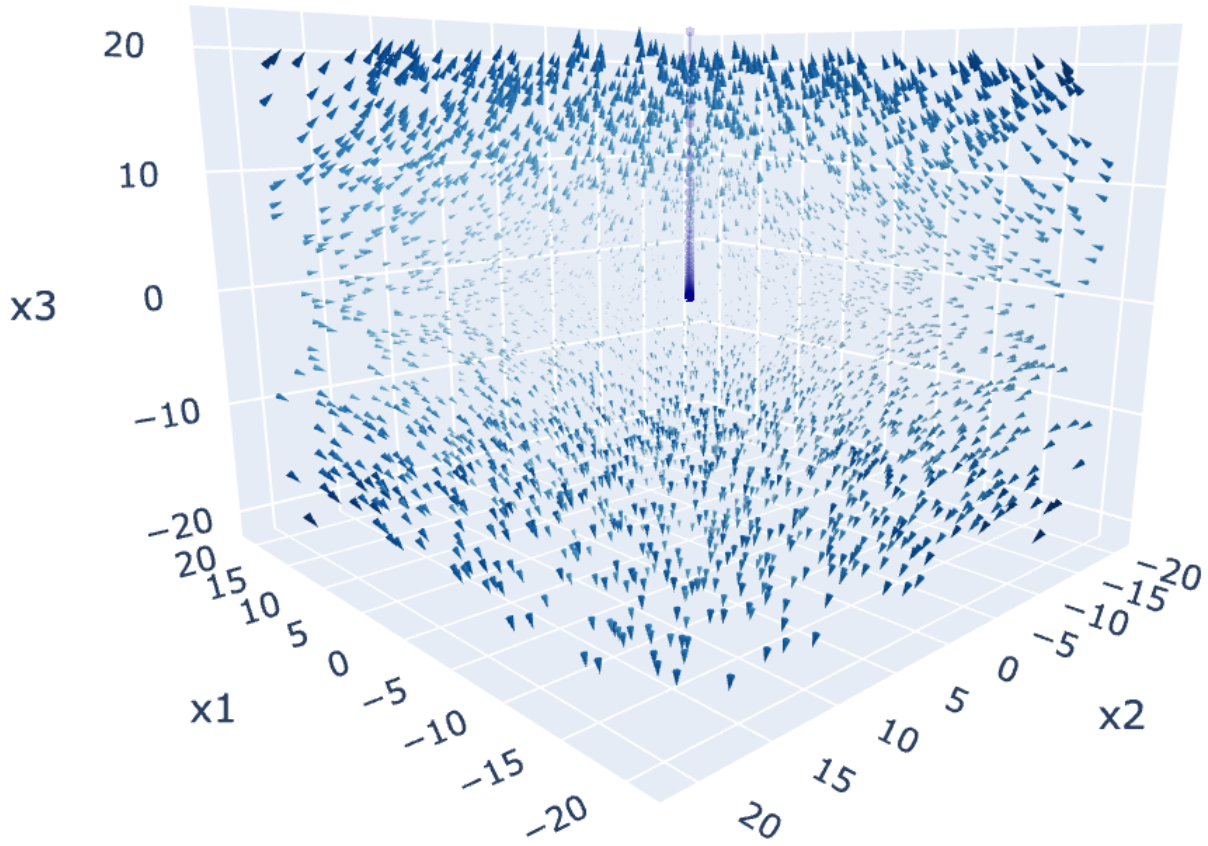

Figure 4: 3D dynamics of $\mathbf{x}(t)$ following the initial condition $\mathbf{x}_{0,4}$ for $t \in [-17, 17]$.

Eventually, we find that our particular solution takes the form

$$\mathbf{x}_p(t) = \begin{pmatrix} -t/2 \\ t/2 + 3 \\ 0 \end{pmatrix}$$

And the overall general solution for the inhomogeneous DEQ is

$$\mathbf{x}(t) = c_1 \mathbf{v}_1 e^{\lambda_1 t} + c_2 \mathbf{v}_2 e^{\lambda_2 t} + c_3 \mathbf{v}_3 e^{\lambda_3 t} + \mathbf{x}_p(t)$$

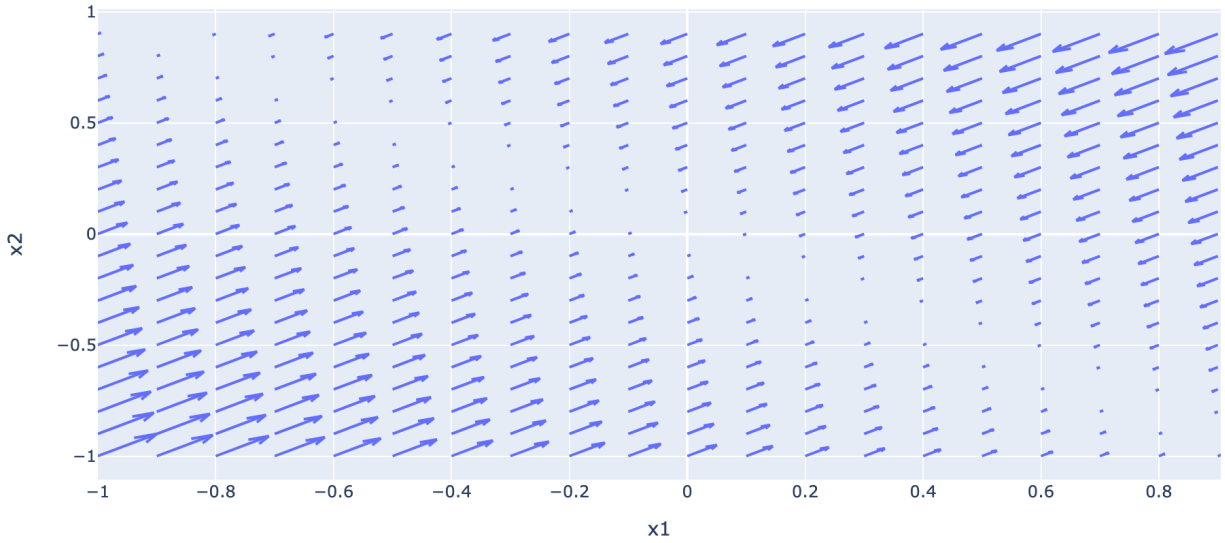

Figure 5: Projection of $\mathbf{f}(x) = \mathbf{Ax}$ onto a plane defined by $x_3 = 0$.

Question 2. Nonlinear Decision Network.

2.1 We are given a system of nonlinear DEQs

$$\tau \dot{u}_1(t) = -u_1(t) - c[u_2(t) - \Theta]_+ + s_1$$
$$\tau \dot{u}_2(t) = -u_2(t) - c[u_1(t) - \Theta]_+ + s_2$$

where the linear threshold function is defined by $[x]_+ = \max(0, x)$.
This network model can be re-parameterised as follows

$$\frac{d\tilde{u}_1(\tilde{t})}{d\tilde{t}} = -\tilde{u}_1(\tilde{t}) - c[\tilde{u}_2(\tilde{t})]_+ + \tilde{s}_1$$
$$\frac{d\tilde{u}_2(\tilde{t})}{d\tilde{t}} = -\tilde{u}_2(\tilde{t}) - c[\tilde{u}_1(\tilde{t})]_+ + \tilde{s}_2$$

where we have

$$\tilde{u}_1(t) = u_1(t) - \Theta$$
$$\tilde{u}_2(t) = u_2(t) - \Theta$$
$$\tilde{s}_1 = s_1 - \Theta$$
$$\tilde{s}_2 = s_2 - \Theta$$
$$\tilde{t} = \frac{t}{\tau}$$

For notation purposes, we drop the tilde in the above system.

2.2 The reparameterised system can be written in the matrix form as show below

$$\mathbf{f}(\mathbf{u}) = -\mathbf{u} - c \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} [\mathbf{u}(\tilde{t})]_+ + \mathbf{s}$$

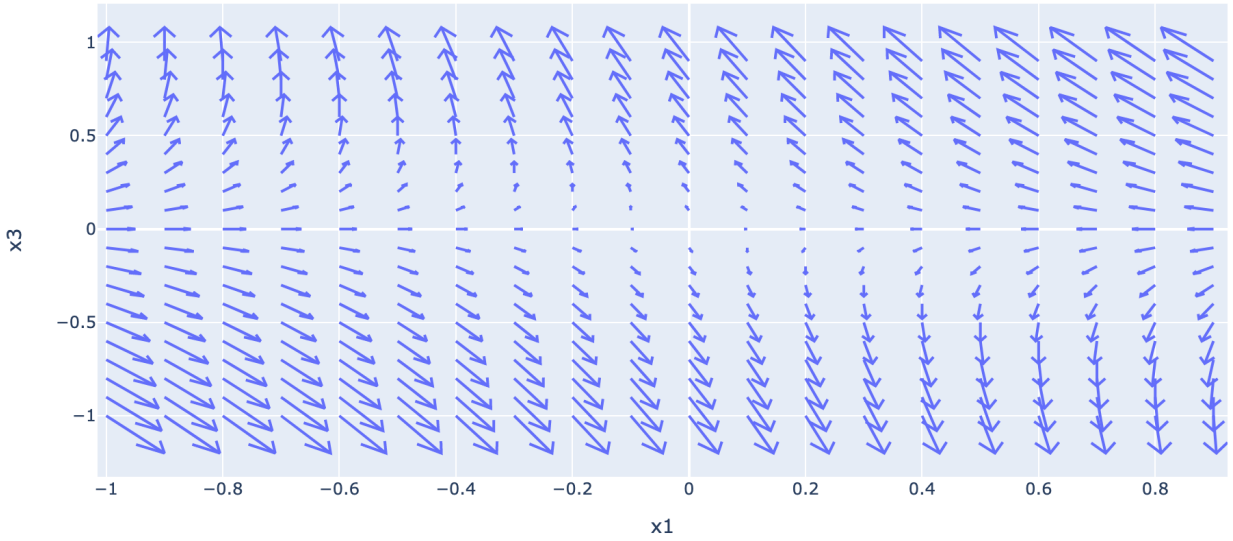

Figure 6: Projection of $\mathbf{f}(x) = \mathbf{A}\mathbf{x}$ onto a plane defined by $x_2 = 0$.

with $\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ and $\mathbf{s} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$. Assuming $c = 2$ and $s_1 = s_2 = 1$, the vector field of this dynamics appears in Figure 8.

2.3 To compute the fixed points, we evaluate the following equation

$$\mathbf{f}(\mathbf{u}) = 0$$

We have

$$\begin{cases} -u_1(t) - 2[u_2(t)]_+ + 1 = 0 \\ -2[u_1(t)]_+ - u_2(t) + 1 = 0 \end{cases}$$

$$\implies \begin{cases} u_1(t) + 2[u_2(t)]_+ = 1 \\ 2[u_1(t)]_+ + u_2(t) = 1 \end{cases}$$

$$\implies \begin{cases} u_1(t) = 1 - 2[u_2(t)]_+ \\ u_2(t) = 1 - 2[u_1(t)]_+ \end{cases}$$

Now, we need to consider the four quadrants. For $u_1, u_2 > 0$, we have

$$\begin{cases} u_1(t) = 1 - 2u_2(t) \\ u_2(t) = 1 - 2u_1(t) \end{cases} \implies \begin{cases} u_1(t) = \frac{1}{3} \\ u_2(t) = \frac{1}{3} \end{cases}$$

Similarly, for $u_1 > 0, u_2 < 0$

$$\begin{cases} u_1(t) = 1 \\ u_2(t) = -1 \end{cases}$$

For $u_1 < 0, u_2 > 0$

$$\begin{cases} u_1(t) = -1 \\ u_2(t) = 1 \end{cases}$$

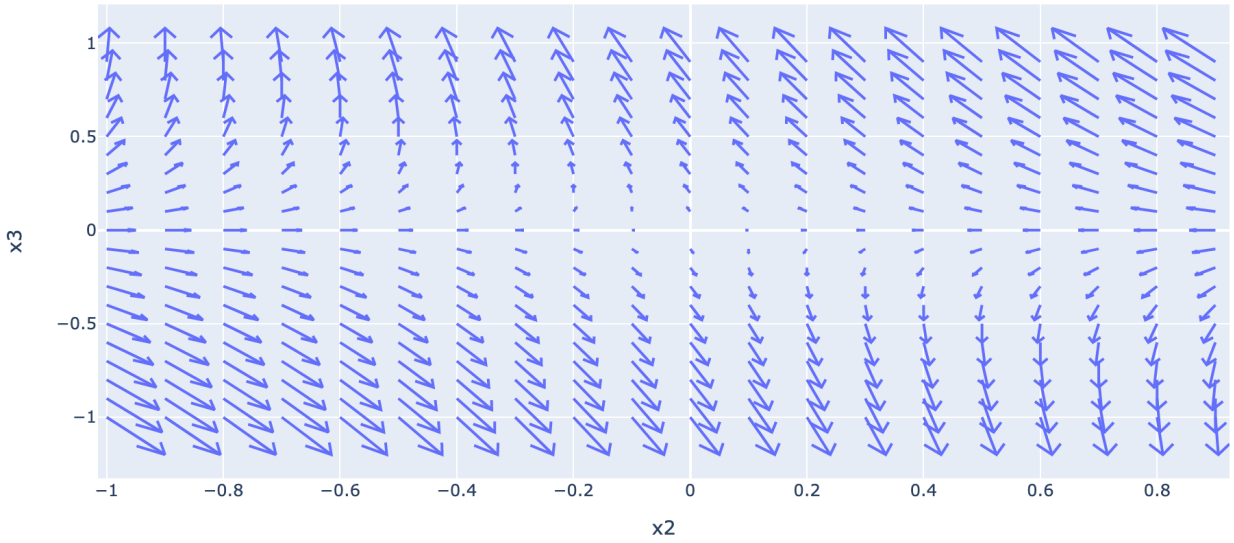

Figure 7: Projection of $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ onto a plane defined by $x_1 = 0$.

For $u_1 < 0, u_2 < 0$ we get a contradiction, thus in this quadrant there are no fixed points.

2.4 To compute the dynamics matrix $\mathbf{A}$, we need to solve for

$$0 = \mathbf{A}\mathbf{u} + \mathbf{s}$$

Which we have already done in 2.3. As such, we have

$$\mathbf{A}_{u_1>0,u_2>0} = \begin{pmatrix} -1 & -2 \\ -2 & -1 \end{pmatrix}, \quad \mathbf{A}_{u_1<0,u_2>0} = \begin{pmatrix} -1 & -2 \\ 0 & -1 \end{pmatrix}, \quad \mathbf{A}_{u_1<0,u_2<0} = \begin{pmatrix} -1 & 0 \\ -2 & -1 \end{pmatrix}$$

We have already showed how to eigendecompose a matrix in 1.1. Thus, following the same procedure, the eigenvalues for $\mathbf{A}_{u_1>0,u_2>0}$ are $\lambda_1 = -3$ and $\lambda_2 = 1$, which describes a stable dynamics towards the fixed point (attractor). For $\mathbf{A}_{u_1<0,u_2>0}$, there is a single eigenvalue $\lambda = -1$ with algebraic multiplicity 2, which also yeilds an attractor. Finally, for $\mathbf{A}_{u_1<0,u_2<0}$ we also have $\lambda = -1$ with algebraic multiplicity 2.

2.5 For the two different inputs, the stationary solutions $\mathbf{u}(\infty)$ for the below initial conditions

- $\mathbf{u}(0) = \mathbf{0}$
- $\mathbf{u}(0) = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$
- $\mathbf{u}(0) = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$

were computed numerically using the Forward Euler method approximation. The simulation results appear in Figures 9 and 10. As clearly visible from the figures,

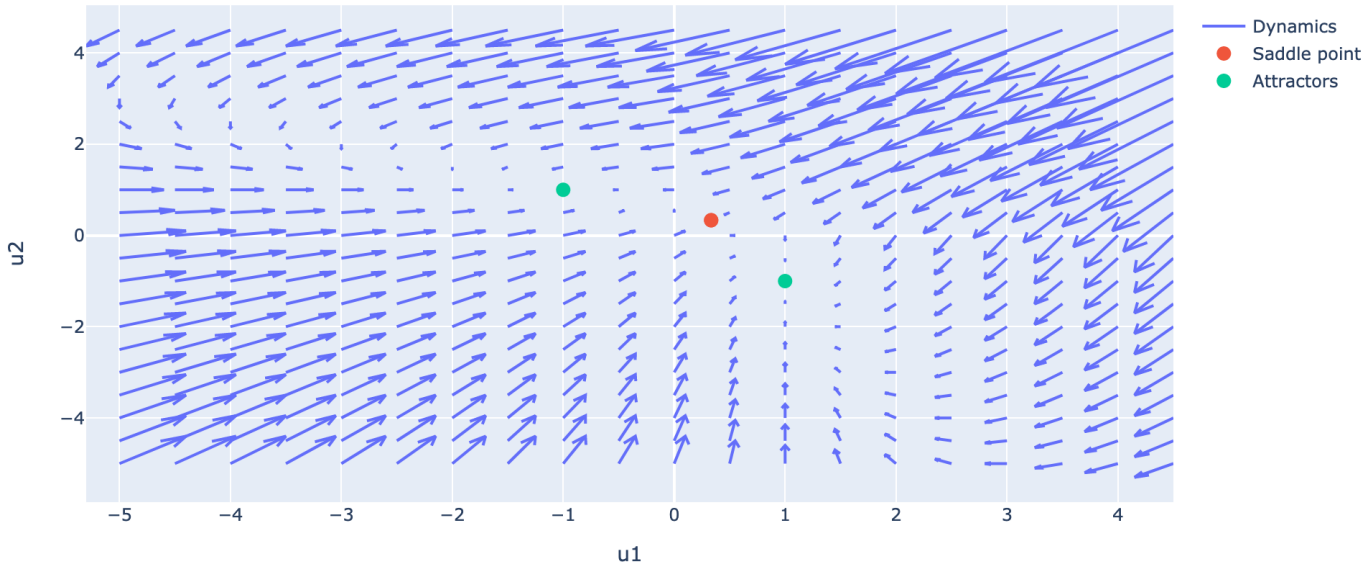

Figure 8: Decision network dynamics with a constant input $s_1 = s_2 = 1$.

depending on the input the same initial conditions converge to different attractors. This means that the two neurons are able to discriminate between the two input signals by the means of different fixed firing rates.

2.6 For $s_1 = s_2 = 1$ and $c = -2$ the dynamics and the stationary solutions like like shown in Figure 11. There are no fixed points in this system, and hence the solutions do not converge.

2.7 We now replace the ReLU nonlinearity in our original system with the following step funciton

$$1(x) = \begin{cases} 1, & (x > 0) \\ 0, & (x \leq 0) \end{cases}$$

The vector field now changes to what can be seen in Figure 12 along with the stationary solutions specified by the initial conditions in 2.5 and inputs in 2.6. All the solutions converge towards the attractor at $(3, 3)$, which is obvious from our fixed point computations in 2.3. As opposed to ReLU, our new nonlinearity introduces an upper bound on convergence.

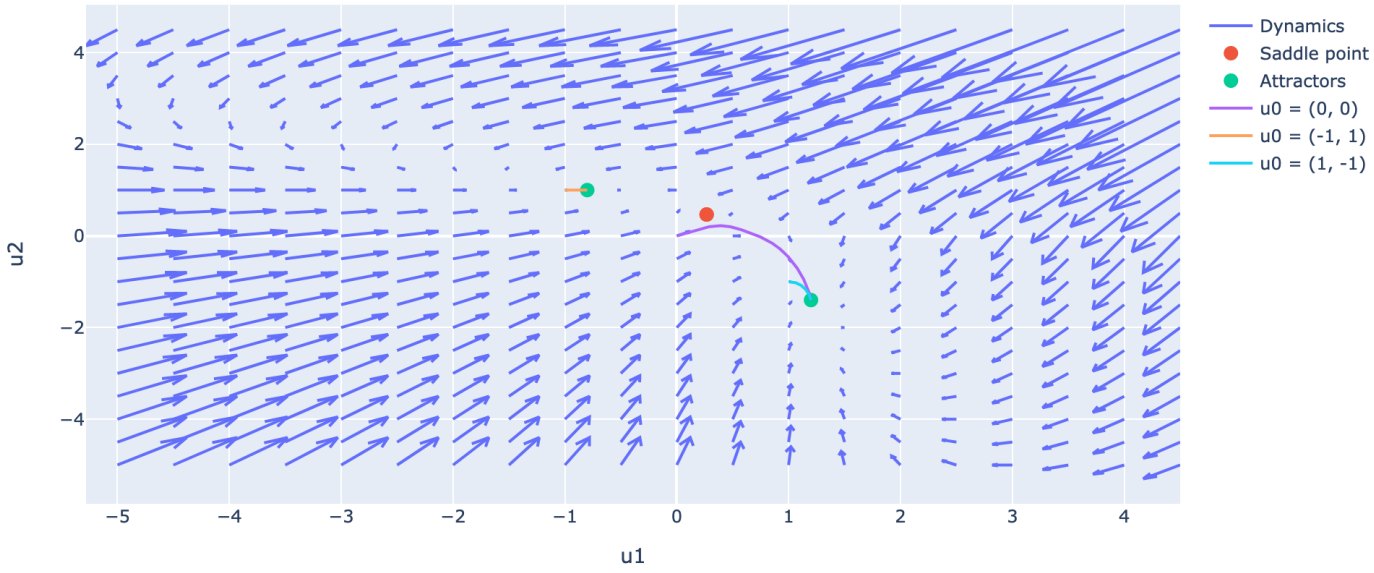

Figure 9: Stationary solutions with the constant input $s_1 = 1.2$, $s_2 = 1$.

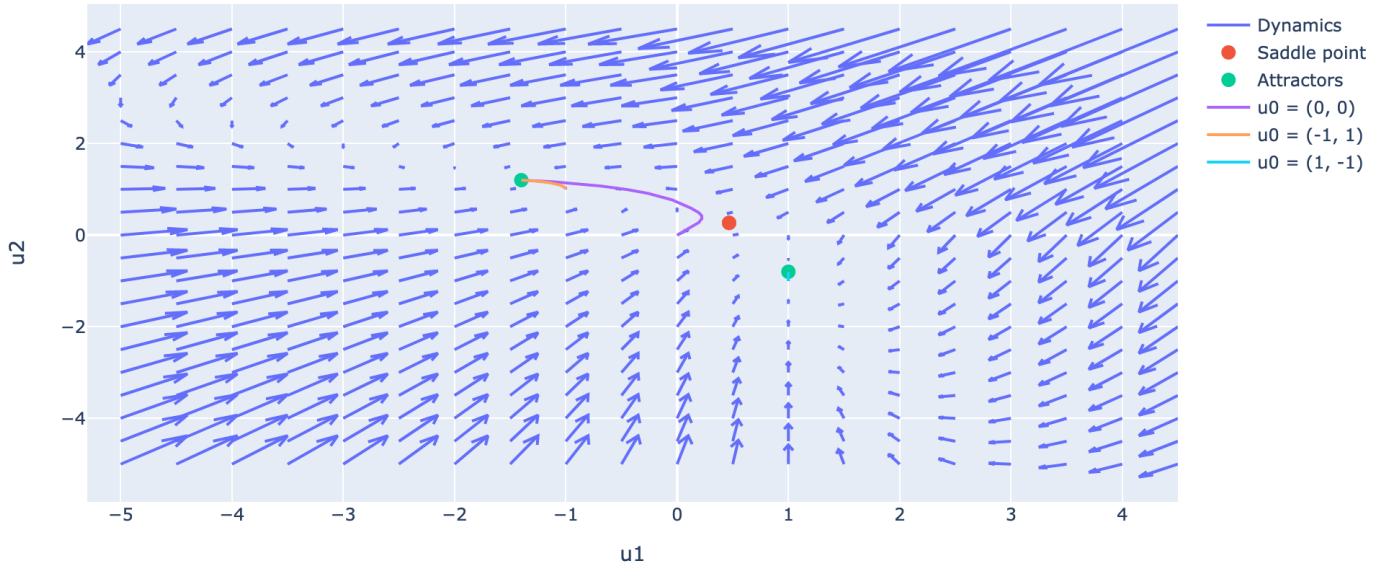

Figure 10: Stationary solutions with the constant input $s_1 = 1$, $s_2 = 1.2$.

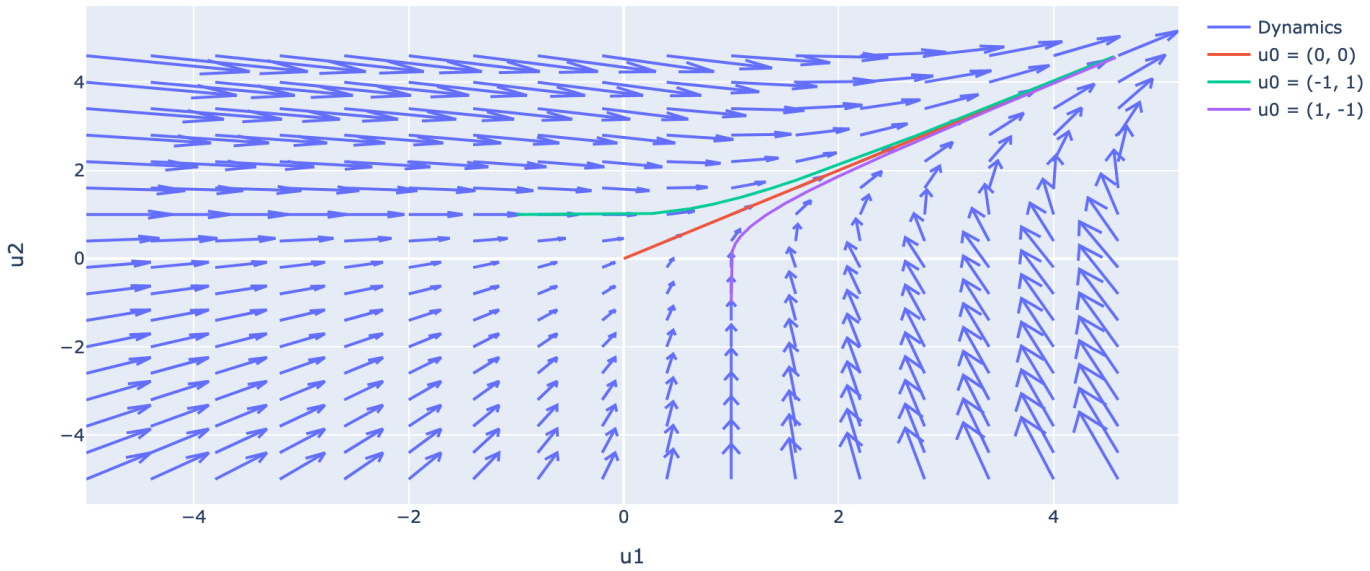

Figure 11: Stationary solutions with the constant input $s_1 = 1$, $s_2 = 1.2$ and $c = -2$.

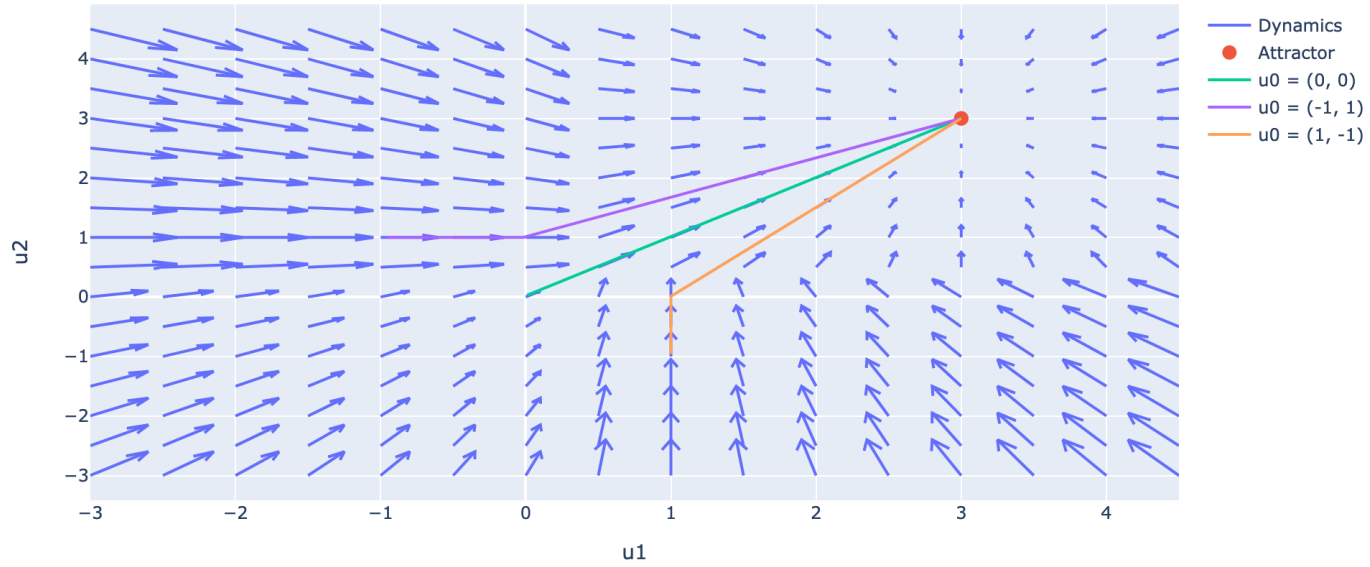

Figure 12: Stationary solutions with the constant input $s_1 = 1$, $s_2 = 1$ and $c = -2$ with the nonlinearity changed to $1(x)$.

LD

December 11, 2019

[ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.graph_objs as go
import plotly.figure_factory as ff
import pandas as pd
import numpy as np
```

[ ]:
```
def project(V, x):
    dot1 = np.dot(V[0], x)
    dot2 = np.dot(V[1], x)
    return dot1, dot2
```

[ ]:
```
def ReLU(x):
    return x * (x > 0)

def one(x):
    return 1 * (x > 0)
```

[ ]:
```
x = np.random.permutation(np.arange(-20, 20, 0.01))
y = np.random.permutation(np.arange(-20, 20, 0.01))
z = np.random.permutation(np.arange(-20, 20, 0.01))

u = -0.5*x - 0.5*y
v = -0.5*x - 0.5*y
w = 2*z

data = [{
    "type": "cone",
    "x": x,
    "y": y,
    "z": z,
    "u": u,
    "v": v,
    "w": w,
    "colorscale": 'Blues',
    "sizemode": "absolute",
    "sizeref": 50
}]
```

```
layout = {
    "scene": {
        "aspectratio": {"x": 1, "y": 1, "z": 0.8},
        "camera": {
            "eye": {"x": 1.2, "y": 1.2, "z": 0.6}
        },
    'xaxis_title': 'x1',
    'yaxis_title': 'x2',
    'zaxis_title': 'x3'
    }
}

fig = go.Figure(data=data, layout=layout)
fig.show()
```

[ ]:
```
x, y = np.meshgrid(np.arange(-1, 1, 0.1), np.arange(-1, 1, 0.1))

u = -0.5*x - 0.5*y
v = -0.5*x - 0.5*y
w = 2*y
```

[ ]:
```
fig = ff.create_quiver(x, y, u, w)
fig['layout'].update(xaxis_title='x1', yaxis_title='x3')
fig.show()
```

[ ]:
```
e1 = (1/np.sqrt(2))*np.array([1, -1, 0])
e2 = np.array([0, 0, 1])
V = [e1, e2]
```

[ ]:
```
x = np.arange(-1, 1, 0.05)
y = np.arange(-1, 1, 0.05)

u = -0.5*x - 0.5*y
v = -0.5*x - 0.5*y

xp = []
yp = []

up = []
vp = []

for i in range(len(x)):
    for j in range(len(y)):
        xd, yd = project(V, [x[i], y[j], 0])
        ud, vd = project(V, [u[i], v[j], 0])
        xp.append(xd)
        yp.append(yd)
        up.append(ud)
```

```
        vp.append(vd)
```

[ ]:
```
fig = ff.create_quiver(xp, yp, up, vp)
fig['layout'].update(xaxis_title='x1', yaxis_title='x3')
fig.show()
```

[ ]:
```
# Question 2.2
x, y = np.meshgrid(np.arange(-5, 5, 0.5), np.arange(-5, 5, 0.5))
s1 = s2 = 1
c = 2

du1 = -x -c*ReLU(y) + s1
du2 = -y -c*ReLU(x) + s2
```

[ ]:
```
fig = ff.create_quiver(x, y, du1, du2, name='Dynamics')
fig['layout'].update(xaxis_title='u1', yaxis_title='u2')
fig.add_trace(go.Scatter(x=[1/3, 1/3], y=[1/3,1/3],
                    mode='markers',
                    marker_size=10,
                    name='Saddle point'))
fig.add_trace(go.Scatter(x=[1, -1], y=[-1, 1],
                    mode='markers',
                    marker_size=10,
                    name='Attractors'))
fig.show()
```

[ ]:
```
# Question 2.5
x, y = np.meshgrid(np.arange(-5, 5, 0.5), np.arange(-5, 5, 0.5))
s1 = 1
s2 = 1.2
c = 2

du1 = -x -c*ReLU(y) + s1
du2 = -y -c*ReLU(x) + s2

xt = 0
yt = 0
u0t = []
v0t = []
for t in range(1, 10000):
    xt = xt + 0.001*(-xt -c*ReLU(yt) + s1)
    yt = yt + 0.001*(-yt -c*ReLU(xt) + s2)

    u0t.append(xt)
    v0t.append(yt)

xt1 = -1
yt1 = 1
u1t = []
```

```
v1t = []
for t in range(1, 10000):
    xt1 = xt1 + 0.001*(-xt1 -c*ReLU(yt1) + s1)
    yt1 = yt1 + 0.001*(-yt1 -c*ReLU(xt1) + s2)

    u1t.append(xt1)
    v1t.append(yt1)

xt2 = 1
yt2 = -1
u2t = []
v2t = []
for t in range(1, 10000):
    xt2 = xt2 + 0.001*(-xt2 -c*ReLU(yt2) + s1)
    yt2 = yt2 + 0.001*(-yt2 -c*ReLU(xt2) + s2)

    u2t.append(xt2)
    v2t.append(yt2)
```

[ ]:
```
fig = ff.create_quiver(x, y, du1, du2, name='Dynamics')
fig['layout'].update(xaxis_title='u1', yaxis_title='u2')
fig.add_trace(go.Scatter(x=[0.467,0.467], y=[0.267, 0.267],
                     mode='markers',
                     marker_size=10,
                     name='Saddle point'))
fig.add_trace(go.Scatter(x=[-1.4, 1], y=[1.2, -0.8],
                     mode='markers',
                     marker_size=10,
                     name='Attractors'))
fig.add_trace(go.Scatter(x=u0t, y=v0t,
                     marker=dict(
                         size=2,
                         autocolorscale=True),
                     name='u0 = (0, 0)'))
fig.add_trace(go.Scatter(x=u1t, y=v1t,
                     marker=dict(
                         size=2,
                         autocolorscale=True),
                     name='u0 = (-1, 1)'))
fig.add_trace(go.Scatter(x=u2t, y=v2t,
                     marker=dict(
                         size=2,
                         autocolorscale=True),
                     name='u0 = (1, -1)'))
fig.show()
```

[ ]:
```
# Question 2.6
x, y = np.meshgrid(np.arange(-5, 5, 0.6), np.arange(-5, 5, 0.6))
```

```
s1 = 1
s2 = 1
c = -2

du1 = -x -c*ReLU(y) + s1
du2 = -y -c*ReLU(x) + s2

xt = 0
yt = 0
u0t = []
v0t = []
for t in range(1, 1600):
    xt = xt + 0.001*(-xt -c*ReLU(yt) + s1)
    yt = yt + 0.001*(-yt -c*ReLU(xt) + s2)

    u0t.append(xt)
    v0t.append(yt)

xt1 = -1
yt1 = 1
u1t = []
v1t = []
for t in range(1, 1600):
    xt1 = xt1 + 0.001*(-xt1 -c*ReLU(yt1) + s1)
    yt1 = yt1 + 0.001*(-yt1 -c*ReLU(xt1) + s2)

    u1t.append(xt1)
    v1t.append(yt1)

xt2 = 1
yt2 = -1
u2t = []
v2t = []
for t in range(1, 1600):
    xt2 = xt2 + 0.001*(-xt2 -c*ReLU(yt2) + s1)
    yt2 = yt2 + 0.001*(-yt2 -c*ReLU(xt2) + s2)

    u2t.append(xt2)
    v2t.append(yt2)
```

[ ]:
```
fig = ff.create_quiver(x, y, du1, du2, name='Dynamics')
fig['layout'].update(xaxis_title='u1', yaxis_title='u2')
fig.add_trace(go.Scatter(x=u0t, y=v0t,
                         marker=dict(
                             size=2,
                             autocolorscale=True),
                         name='u0 = (0, 0)'))
```

```
fig.add_trace(go.Scatter(x=u1t, y=v1t,
                         marker=dict(
                             size=2,
                             autocolorscale=True),
                         name='u0 = (-1, 1)'))
fig.add_trace(go.Scatter(x=u2t, y=v2t,
                         marker=dict(
                             size=2,
                             autocolorscale=True),
                         name='u0 = (1, -1)'))
fig.show()
```

[ ]:
```
# Question 2.7
x, y = np.meshgrid(np.arange(-3, 5, 0.5), np.arange(-3, 5, 0.5))
s1 = s2 = 1
c = -2

du1 = -x -c*one(y) + s1
du2 = -y -c*one(x) + s2

xt = 0
yt = 0
u0t = []
v0t = []
for t in range(1, 2000):
    xt = xt + 0.01*(-xt -c*one(yt) + s1)
    yt = yt + 0.01*(-yt -c*one(xt) + s2)

    u0t.append(xt)
    v0t.append(yt)

xt1 = -1
yt1 = 1
u1t = []
v1t = []
for t in range(1, 2000):
    xt1 = xt1 + 0.01*(-xt1 -c*one(yt1) + s1)
    yt1 = yt1 + 0.01*(-yt1 -c*one(xt1) + s2)

    u1t.append(xt1)
    v1t.append(yt1)

xt2 = 1
yt2 = -1
u2t = []
v2t = []
for t in range(1, 2000):
```

```
    xt2 = xt2 + 0.01*(-xt2 -c*one(yt2) + s1)
    yt2 = yt2 + 0.01*(-yt2 -c*one(xt2) + s2)

    u2t.append(xt2)
    v2t.append(yt2)
```

[ ]:
```
fig = ff.create_quiver(x, y, du1, du2, name='Dynamics')
fig['layout'].update(xaxis_title='u1', yaxis_title='u2')
fig.add_trace(go.Scatter(x=[3, 3], y=[3, 3],
                      mode='markers',
                      marker_size=10,
                      name='Attractor'))
fig.add_trace(go.Scatter(x=u0t, y=v0t,
                      marker=dict(
                          size=2,
                          autocolorscale=True),
                      name='u0 = (0, 0)'))
fig.add_trace(go.Scatter(x=u1t, y=v1t,
                      marker=dict(
                          size=2,
                          autocolorscale=True),
                      name='u0 = (-1, 1)'))
fig.add_trace(go.Scatter(x=u2t, y=v2t,
                      marker=dict(
                          size=2,
                          autocolorscale=True),
                      name='u0 = (1, -1)'))
fig.show()
```