2.1.1)

Given

$$\dot{x}_1 = -x_1 + 2x_2^3 + x_2$$
$$\dot{x}_2 = -x_1 - x_2$$

Find

1. Find all equilibrium points
2. Determine the type of each isolated equilibrium
3. Draw vector field plot

Solution

Find equilibrium points

>>>
```
/* Define equations */
xd1: -x1 + 2*x1^3 + x2;
xd2: -x1 - x2;

/* Find roots */
solve([xd1=0, xd2=0]);
```

$\mathrm{x2} + 2\mathrm{x1}^3 - \mathrm{x1}$

$[[\mathrm{x2} = 1, \mathrm{x1} = -1], [\mathrm{x2} = -1, \mathrm{x1} = 1], [\mathrm{x2} = 0, \mathrm{x1} = 0]]$

$-\mathrm{x2} - \mathrm{x1}$

Determine equilibrium point types

>>>
```
/* Calculate Jacobian */
J:jacobian([xd1, xd2], [x1, x2]);

/* Calculate eigenvalue for each equilibrium point  */
/* The eigenvalue output is of the following format */
/* [[eigenvalues], [multiplicity]]                  */

float(eivals(psubst([x1=-1, x2=1], J)));
float(eivals(psubst([x1=1, x2=-1], J)));
float(eivals(psubst([x1=0, x2=0], J)));
```

$[[-0.8284271247461907, 4.828427124746191], [1.0, 1.0]]$

$[[-0.8284271247461907, 4.828427124746191], [1.0, 1.0]]$

$[[-1.0i - 1.0, i - 1.0], [1.0, 1.0]]$

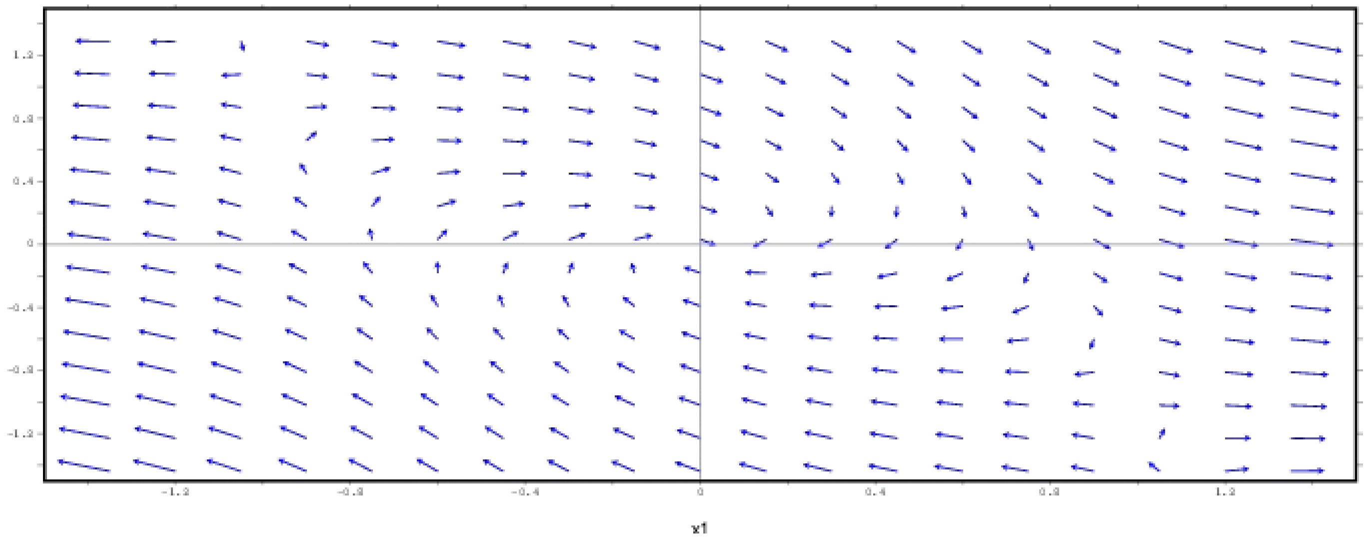
$$\begin{pmatrix} 6\mathrm{x1}^2 - 1 & 1 \\ -1 & -1 \end{pmatrix}$$

Based on the eigenvalues found from the Jacobians found at each of the equilibrium points, we can determine that they are:

- (-1, 1): Saddle
- (0,0): Stable focus
- (1, -1): Saddle

Draw vector field

>>>
```
plotdf([xd1, xd2], [x1, x2], [x1, -1.5, 1.5], [x2, -1.5, 1.5])$
```



2.1.2)

Given

$$\dot{x}_1 = x_1 + x_1 x_2$$
$$\dot{x}_2 = -x_2 + x_2^2 + x_1 x_2 - x_1^3$$

Find

1. Find all equilibrium points
2. Determine the type of each isolated equilibrium
3. Draw vector field plot

Solution

Find equilibrium points

>>>
```
/* Define equations */
xd1: x1 + x1*x2;
xd2:-x2 + x2^2 + x1*x2 - x1^3;

/* Find roots */
solve([xd1=0, xd2=0]);
```

$$\left[[x2 = 1, x1 = 0], [x2 = 0, x1 = 0], [x2 = -1, x1 = 1], \left[x2 = -1, x1 = -\frac{\sqrt{7}i + 1}{2}\right], \left[x2 = -1, x1 = \frac{\sqrt{7}i - 1}{2}\right]\right]$$

$$x1x2 + x1$$

$$x2^2 + x1x2 - x2 - x1^3$$

Determine equilibrium point types

```
>>> /* Calculate Jacobian */
J:jacobian([xd1, xd2], [x1, x2]);

/* Calculate eigenvalue for each equilibrium point  */
/* The eigenvalue output is of the following format */
/* [[eigenvalues], [multiplicity]]                  */

float(eivals(psubst([x1=0, x2=1], J)));
float(eivals(psubst([x1=0, x2=0], J)));
float(eivals(psubst([x1=1, x2=-1], J)));
```

$$[[-1.0, 1.0], [1.0, 1.0]]$$

$$\begin{pmatrix} x2+1 & x1 \\ x2-3x1^2 & 2x2+x1-1 \end{pmatrix}$$

$$[[-1.732050807568877i-1.0, 1.732050807568877i-1.0], [1.0, 1.0]]$$

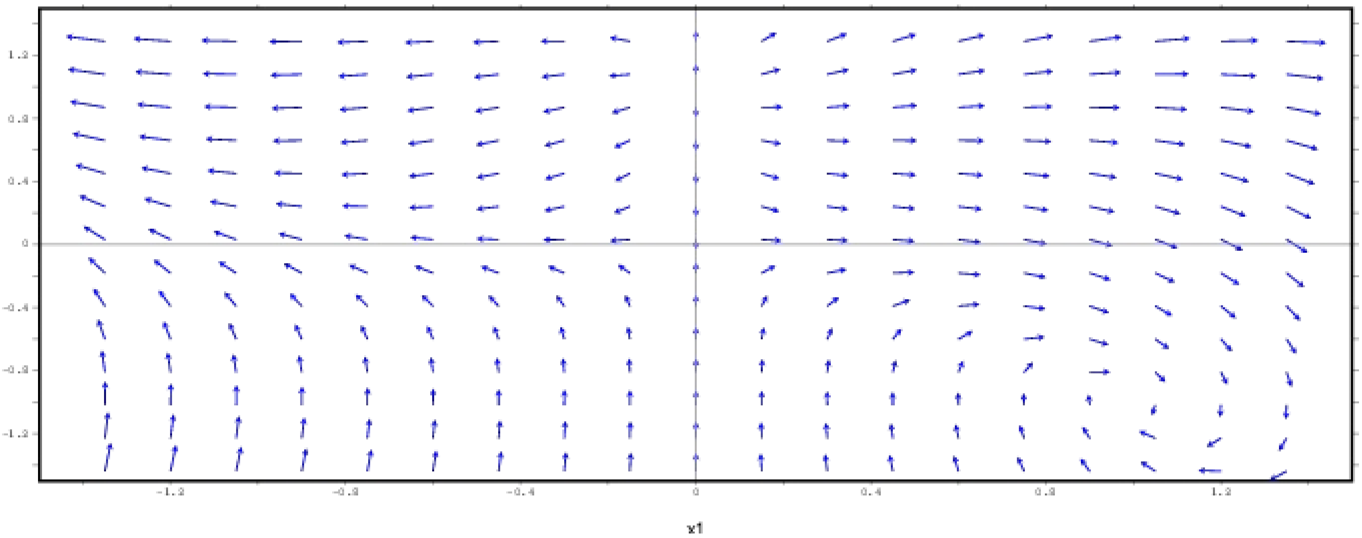
$$[[1.0, 2.0], [1.0, 1.0]]$$

Based on the eigenvalues found from the Jacobians found at each of the equilibrium points, we can determine that they are:

- (0,1): Unstable node
- (0,0): Saddle
- (1,-1): Stable focus

Draw vector field

```
>>> plotdf([xd1, xd2], [x1, x2], [x1, -1.2, 1.5], [x2, -1.5, 3.5])$
```



Canvas Problem)

Given

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -0.8x_1 - 10x_1^2x_2 + u$$

Find

1. Develop an approximate, higher order linearization
2. Plot results

Solution

```
>>> /* Define original state */
xd1: x2$
xd2:-0.8*x1 - 10*x1^2*x2$

/* Find roots */
solve([xd1=0, xd2=0]);

/* Calculate Jacobian */
J:jacobian([xd1, xd2], [x1, x2])$

/* Calculate eigenvalue for each equilibrium point  */
/* The eigenvalue output is of the following format */
/* [[eigenvalues], [multiplicity]]                  */
float(eivals(psubst([x1=0, x2=0], J)));

/* Draw vector field */
plotdf([xd1, xd2], [x1, x2], [x1, -1.2, 1.5], [x2, -1.5, 3.5])$
```

```
rat: replaced -0.8 by -4/5 = -0.8
```

$$[[x1 = 0, x2 = 0]]$$

$$[[-0.8944271909999159i, 0.8944271909999159i], [1.0, 1.0]]$$

Define new linearized states

```
>>> /* Define state */
xt1: x2(t)$
xt2: -0.8*x1(t)$
xt3: -10*x1(t)^2*x2(t)$

/* Differentiate to get forms of xt# */
diff(xt1);
diff(xt2);
diff(xt3);

/* Define first derivative of new state */
xdt1: x2 + x3$
xdt2: -0.8*x1$
xdt3: (20*x2*x1^2)/(0.8) - (10*x2^2*x1)/(0.8)$

/* Calculate Jacobian */
J:jacobian([xdt1, xdt2, xdt3], [x1, x2, x3])$

/* Evalulate Jacobian at equilibrium point */
float(eivals(psubst([x1=0, x2=0], J)));
```

$$-0.8\left(\frac{d}{dt}x1(t)\right)dt$$

$$\left(-10x1(t)^2\left(\frac{d}{dt}x2(t)\right) - 20x1(t)\,x2(t)\left(\frac{d}{dt}x1(t)\right)\right)dt$$

$$[[-0.8944271909999159i, 0.8944271909999159i, 0.0], [1.0, 1.0, 1.0]]$$

$$\frac{d}{dt}x2(t)\,dt$$

From this we can see that we have retained the same eigenvalues (showing that the higher order state is another relization). Below are some slices of the contour plots.

```
>>> /* Draw vector field slices */
plotdf([xdt1], [x2, x3])$
```

```
>>> plotdf([xdt2, xdt3], [x1, x2])$
```