```
In[236]:= Clear["Global`*"]
```

BDH Exercises

Set up system and Jacobian

```
In[327]:= system = {
    x (1 - x) - x * y,
    y (1 - y) + x * y - y * z,
    z (1 - z) + y * z
   }

Out[327]= {(1 - x) x - x y, x y + (1 - y) y - y z, y z + (1 - z) z}

In[328]:= vars = {x, y, z}

Out[328]= {x, y, z}

In[363]:= jacobian = D[system, {vars}]

Out[363]= {{1 - 2 x - y, -x, 0}, {y, 1 + x - 2 y - z, -y}, {0, z, 1 + y - 2 z}}

In[364]:= jacobian // TraditionalForm
```

Out[364]//TraditionalForm=

$$\begin{pmatrix} -2x-y+1 & -x & 0 \\ y & x-2y-z+1 & -y \\ 0 & z & y-2z+1 \end{pmatrix}$$

```
In[330]:= equilibria = {
    {x → 1, y → 0, z → 0},
    {x → 0, y → 1, z → 0},
    {x → 0, y → 0, z → 1},
    {x → 1, y → 0, z → 1}
   }

Out[330]= {{x → 1, y → 0, z → 0}, {x → 0, y → 1, z → 0}, {x → 0, y → 0, z → 1}, {x → 1, y → 0, z → 1}}
```

Calculating the eigenvalues of the equilibrium of coexistence

```
In[389]:= coexistence = (Solve[{system == 0, x ≠ 0, y ≠ 0, z ≠ 0}, {x, y, z}])
```

Out[389]= $\left\{\left\{x \to \frac{2}{3}, y \to \frac{1}{3}, z \to \frac{4}{3}\right\}\right\}$

```
In[390]:= Eigenvalues[D[sys, {vars}] /. coexistence]
```

Out[390]= $\left\{-1, -\frac{2}{3} + \frac{2 i}{3}, -\frac{2}{3} - \frac{2 i}{3}\right\}$

Top Predator Disease State

In[254]:=
```
dsys = {
   x (1 - x) - x * y,
   y (1 - y) + x * y - y * z,
   z (1 - z) + y * z - γ * z
  }
```

Out[254]= $\{(1-x)\,x - x\,y,\ x\,y + (1-y)\,y - y\,z,\ y\,z + (1-z)\,z - z\,\gamma\}$

In[255]:=
```
dsolns = Solve[dsys == zeros, vars]
```

Out[255]= $\left\{\{x\to 0, y\to 0, z\to 1-\gamma\}, \{x\to 0, y\to 1, z\to 0\}, \left\{x\to 0, y\to \frac{\gamma}{2}, z\to \frac{2-\gamma}{2}\right\}, \{x\to 1, y\to 0, z\to 0\}, \{x\to 1, y\to 0, z\to 1-\gamma\}, \left\{x\to \frac{2-\gamma}{3}, y\to \frac{1+\gamma}{3}, z\to -\frac{2}{3}(-2+\gamma)\right\}, \{x\to 0, y\to 0, z\to 0\}\right\}$

In[256]:=
```
eoc = dsolns[[6]]
```

Out[256]= $\left\{x\to \frac{2-\gamma}{3}, y\to \frac{1+\gamma}{3}, z\to -\frac{2}{3}(-2+\gamma)\right\}$

In[257]:=
```
eqpoint = {x, y, z} /. eoc
```

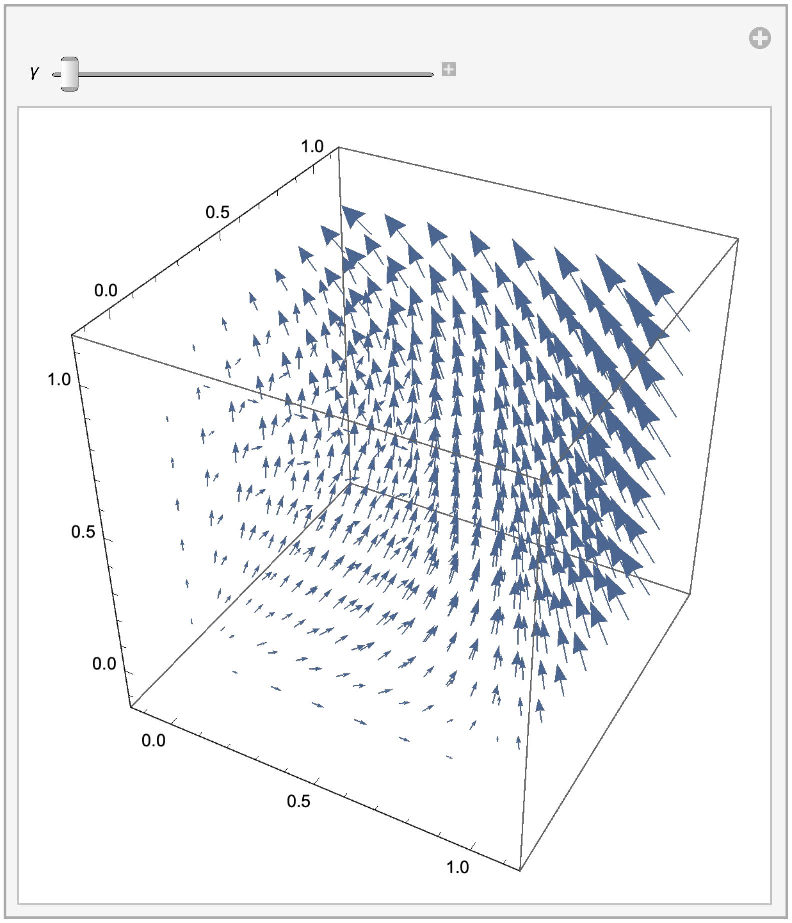
Out[257]= $\left\{\frac{2-\gamma}{3}, \frac{1+\gamma}{3}, -\frac{2}{3}(-2+\gamma)\right\}$

In[258]:=
```
D[eqpoint, γ]
```

Out[258]= $\left\{-\frac{1}{3}, \frac{1}{3}, -\frac{2}{3}\right\}$

In[259]:=
```
Manipulate[VectorPlot3D[{x (1 - x) - x * y, y (1 - y) + x * y - y * z, z (1 - z) + y * z - γ * z},
  {x, 0, 1}, {y, 0, 1}, {z, 0, 1}], {γ, 0, 1}]
```

Out[259]=



In[379]:=
```
djac = D[dsys, {vars}] /. eoc
```

Out[379]=

$$\left\{\left\{1+\frac{1}{3}(-1-\gamma)-\frac{2(2-\gamma)}{3},\ \frac{1}{3}(-2+\gamma),\ 0\right\},\right.$$
$$\left\{\frac{1+\gamma}{3},\ 1+\frac{2-\gamma}{3}+\frac{2}{3}(-2+\gamma)-\frac{2(1+\gamma)}{3},\ \frac{1}{3}(-1-\gamma)\right\},$$
$$\left.\left\{0,\ -\frac{2}{3}(-2+\gamma),\ 1+\frac{4}{3}(-2+\gamma)-\gamma+\frac{1+\gamma}{3}\right\}\right\}$$

In[380]:=
```
djac // TraditionalForm
```

Out[380]//TraditionalForm=

$$\begin{pmatrix} \frac{1}{3}(-\gamma-1)-\frac{2(2-\gamma)}{3}+1 & \frac{\gamma-2}{3} & 0 \\ \frac{\gamma+1}{3} & \frac{2-\gamma}{3}+\frac{2(\gamma-2)}{3}-\frac{2(\gamma+1)}{3}+1 & \frac{1}{3}(-\gamma-1) \\ 0 & -\frac{2}{3}(\gamma-2) & \frac{4(\gamma-2)}{3}-\gamma+\frac{\gamma+1}{3}+1 \end{pmatrix}$$

In[381]:=
```
eqpoint // TraditionalForm
```
Out[381]//TraditionalForm=

$$\left\{\frac{2-\gamma}{3}, \frac{\gamma+1}{3}, -\frac{2}{3}(\gamma-2)\right\}$$

In[263]:=
```
Reduce[eqpoint > 0, γ]
```
Out[263]= $-1 < \gamma < 2$

Equilibria Point Evaluation

First equilibrium point (1,0,0)

Find the eigenvalues and eigenvectors associated with the point

In[334]:=
```
jone = jacobian /. equilibria[[1]]
```
Out[334]= {{-1, -1, 0}, {0, 2, 0}, {0, 0, 1}}

In[335]:=
```
Eigensystem[jone]
```
Out[335]= {{2, -1, 1}, {{-1, 3, 0}, {1, 0, 0}, {0, 0, 1}}}

In[360]:=
```
sysone = jone.(vars - (vars /. equilibria[[1]]))
```
Out[360]= {1 - x - y, 2 y, z}

Three dimensional vector plot of the linearized system about the point

In[361]:=
```
Show[
 VectorPlot3D[
  {sysone},
  {x, 0.0, 1.2},
  {y, 0.0, 1.2},
  {z, 0.0, 1.2},
  Axes → True,
  AxesLabel → {
    Style["x", Bold, FontSize → 24],
    Style["y", Bold, FontSize → 24], Style["z", Bold, FontSize → 24]},
  VectorColorFunction → "Rainbow",
  VectorPoints → 5,
  VectorScale → {0.1, .7, None},
  PerformanceGoal → "Quality"
 ],
 Graphics3D[{
   PointSize[.05], Black, Point[{1, 0, 0}],
   PointSize[.04], Red, Point[{1, 0, 0}]
  }],
 ImageSize → Full
]
```

Two Dimensional “slices” along each of the center planes.

```
StreamPlot[{-x - y, 2 y}, {x, 0.0, 1.5}, {y, 0.0, 1.5}, FrameLabel → {
   Style["x", Bold, FontSize → 24], Style["z", Bold, FontSize → 24]},
 Frame → True, ImageSize → Full,
 StreamColorFunction → "Rainbow", PlotRangePadding → None]
StreamPlot[{-x, y}, {x, 0.0, 1.5}, {y, 0.0, 1.5}, FrameLabel → {
   Style["x", Bold, FontSize → 24], Style["y", Bold, FontSize → 24]}, Frame → True,
 ImageSize → Full, StreamColorFunction → "Rainbow", PlotRangePadding → None]
StreamPlot[{y, z}, {y, 0.0, 1.5}, {z, 0.0, 1.5}, FrameLabel → {
   Style["y", Bold, FontSize → 24], Style["z", Bold, FontSize → 24]}, Frame → True,
 ImageSize → Full, StreamColorFunction → "Rainbow", PlotRangePadding → None]
```

Second equilibrium point (0,1,0)

In[366]:=
```
equilibria[[2]]
```
Out[366]=
```
{x → 0, y → 1, z → 0}
```

In[365]:=
```
jtwo = jacobian /. equilibria[[2]]
```
Out[365]=
```
{{0, 0, 0}, {1, -1, -1}, {0, 0, 2}}
```

In[368]:=
```
Eigensystem[jtwo]
```
Out[368]=
```
{{2, -1, 0}, {{0, -1, 3}, {0, 1, 0}, {1, 1, 0}}}
```

In[369]:=
```
systwo = jtwo.(vars - (vars /. equilibria[[2]]))
```
Out[369]=
```
{0, 1 + x - y - z, 2 z}
```

Third equilibrium point (0,0,1)

In[370]:=
```
equilibria[[3]]
```
Out[370]=
```
{x → 0, y → 0, z → 1}
```

In[371]:=
```
jthree = jacobian /. equilibria[[3]]
```
Out[371]=
```
{{1, 0, 0}, {0, 0, 0}, {0, 1, -1}}
```

In[372]:=
```
Eigensystem[jthree]
```
Out[372]=
```
{{-1, 1, 0}, {{0, 0, 1}, {1, 0, 0}, {0, 1, 1}}}
```

In[373]:=
```
systhree = jthree.(vars - (vars /. equilibria[[3]]))
```
Out[373]=
```
{x, 0, 1 + y - z}
```

Fourth equilibrium point (1,0,1)

In[374]:=
```
equilibria[[4]]
```
Out[374]=
```
{x → 1, y → 0, z → 1}
```

In[375]:=
```
jfour = jacobian /. equilibria[[4]]
```
Out[375]=
```
{{-1, -1, 0}, {0, 1, 0}, {0, 1, -1}}
```

In[376]:=
```
Eigensystem[jfour]
```
Out[376]=
```
{{-1, -1, 1}, {{0, 0, 1}, {1, 0, 0}, {-1, 2, 1}}}
```

In[377]:=
```
sysfour = jfour.(vars - (vars /. equilibria[[4]]))
```
Out[377]=
```
{1 - x - y, y, 1 + y - z}
```