Solving replicator equations with python

$\dot{x}_i = x_i[(Ax)_i - x' Ax]$

In [1]:
```
# Import the required modules
import numpy as np
import matplotlib.pyplot as plt
# This makes the plots appear inside the notebook
%matplotlib inline
```

In [2]:
```
from scipy.integrate import odeint

# Define a function which calculates the derivative
# def dy_dx(y, x):
#     return x - y

# xs = np.linspace(0,5,100)
# y0 = 1.0  # the initial condition
# ys = odeint(dy_dx, y0, xs)
#. ys = np.array(ys).flatten()
```

In [3]:
```
# define the projection to triangular coordinates
proj = np.array(
    [
        [
            -1 * np.cos(30. / 360. * 2. * np.pi),
            np.cos(30. / 360. * 2. * np.pi),
            0.
        ],
        [
            -1 * np.sin(30. / 360. * 2. * np.pi),
            -1 * np.sin(30. / 360. * 2. * np.pi),
            1.,
        ],
    ]
)
```

In [4]:
```
# determine points on boundary of triangle and their projection
ts = np.linspace(0, 1, 10000)
Bd1=np.array([ts,(1-ts),0*ts])
Bd2=np.array([0*ts,ts,(1-ts)])
Bd3=np.array([ts,0*ts,(1-ts)])
PBd1=proj@Bd1
PBd2=proj@Bd2
PBd3=proj@Bd3
```

The matrix $A = \begin{pmatrix} 5 & 1 & 3 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}$ **and the vector** $x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

should in Python be entered as A = np.array([[5, 1 ,3], [1, 1 ,1], [1, 2 ,1]]) and x = np.array([1, 2, 3])

Ax is obtained by A@x

x*x gives pointwise multiplication so this is again a column vector

inner product x.x then use np.transpose(x) @ x

In [24]:
```
# compute orbits

# game Ex 1.7 notes
A = np.array([[ 0, 1 ,0], [ 0, 0 ,2], [ 0, 0 ,1]]) # row, 2nd row,
3rd row
x01 = np.array([0.92, 0.01, 0.07])
x02 = np.array([0.65,0.05,0.3])
x03 = np.array([0.15, 0.05, 0.8])

# game Ex 1.8 notes
#A = np.array([[ 0, 6 ,-4], [ -3, 0 ,5], [ -1, 3 ,0]]) # row, 2nd r
ow, 3rd row
#x01 = np.array([0.01, 0.89, 0.1])
#x02 = np.array([0.05,0.85,0.1])
#x03 = np.array([0.45, 0.1, 0.45])


# game Ex 1.9 notes
#A = np.array([[ 0, 2 ,0], [ 2, 0 ,2], [ 1, 1 ,1]]) # row, 2nd row,
3rd row
#x01 = np.array([0.8, 0.1, 0.1])
#x02 = np.array([0.05,0.85,0.1])
#x03 = np.array([0.53, 0.02, 0.45])

# game Ex 1.10 notes
#A = np.array([[ 1, 5 ,0], [ 0, 1 ,5], [ 5, 0 ,4]]) # row, 2nd row,
3rd row
#x01 = np.array([0.8, 0.1, 0.1])
#x02 = np.array([0.05,0.85,0.1])
#x03 = np.array([0.5, 0.02, 0.48])

## game Ex 1.11  notes
#a=0.2
# a=2
#A = np.array([[ 0, 1 ,-a], [ -a, 0 ,1], [ 1, -a ,0]]) # row, 2nd r
ow, 3rd row
```

```
#x01 = np.array([0.9, 0.01, 0.09])
#x02 = np.array([0.01, 0.45, 0.54])
#x03 = np.array([0.15, 0.35, 0.5])

## game Ex 2.1  notes
A = np.array([[ -1, -1 ,1], [ 0, 0 ,0], [ -1, -1/2 ,1]]) # row, 2nd
row, 3rd row
x01 = np.array([0.9, 0.01, 0.09])
x02 = np.array([0.5, 0.05, 0.45])
x03 = np.array([0.6, 0.05, 0.35])




## game Ex 2.2  notes
b=4
c=1
epsilon=0.01
k=2
w=0.5
delta=w*epsilon
kappa=1-w+w*k*epsilon
theta=w*(1-(k+1)*epsilon)
sigma=(b*theta-c)/(c-c*theta)
A = np.array([[ 0, -1 ,delta*sigma], [ 1, 0 ,-kappa*sigma], [ delta
, -kappa ,0]]) # row, 2nd row, 3rd row
x01 = np.array([0.9, 0.01, 0.09])
x02 = np.array([0.5, 0.05, 0.45])
x03 = np.array([0.6, 0.05, 0.35])

# Exercise for all projects
A = np.array([[ 2, 0 ,1], [ 1, 2 ,0], [ 0, 1 ,2]]) # row, 2nd row,
3rd row
x01 = np.array([0.333, 0.333, 0.334])
x02 = np.array([0.6, 0.2, 0.2])
x03 = np.array([0.2, 0.6, 0.2])

xtr = np.transpose(x01)
def dx_dt(x, t):
    return x * (A@x - np.transpose(x) @ (A@x))
ts = np.linspace(0,100,10000)
xt1 = odeint(dx_dt, x01, ts)
xt2 = odeint(dx_dt, x02, ts)
xt3 = odeint(dx_dt, x03, ts)
xttr1=np.transpose(xt1)
xttr2=np.transpose(xt2)
xttr3=np.transpose(xt3)
print(delta,kappa,theta,sigma,A)
```

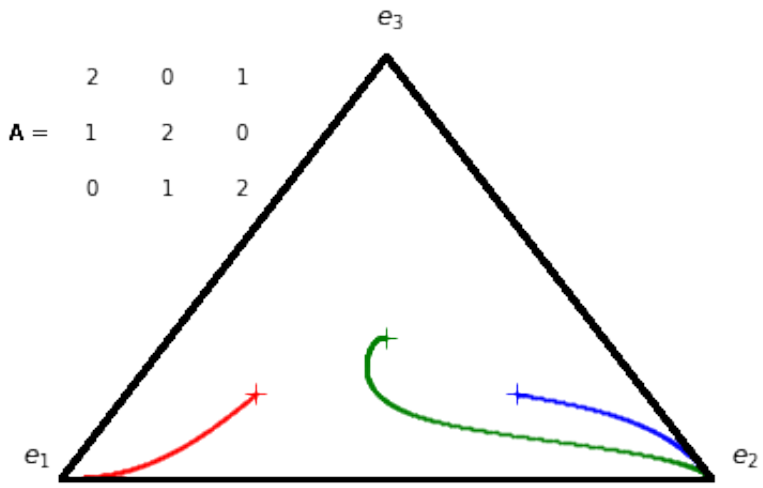
```
0.005 0.51 0.485 1.8252427184466018 [[2 0 1]
 [1 2 0]
 [0 1 2]]
```

In [ ]:

In [25]:
```
orbittriangle1=proj@xttr1
orbittriangle2=proj@xttr2
orbittriangle3=proj@xttr3
ic1=proj@x01
ic2=proj@x02
ic3=proj@x03
```

In [26]:
```
# plot the orbits and the boundary points (and also corner points)
plt.box(False)
plt.axis(False)

plt.plot(orbittriangle1[0],orbittriangle1[1],".",markersize=1,color
='green')
plt.plot(orbittriangle2[0],orbittriangle2[1],".",markersize=1,color
='red')
plt.plot(orbittriangle3[0],orbittriangle3[1],".",markersize=1,color
='blue')

plt.plot(ic1[0],ic1[1],"+",markersize=10,color='green')
plt.plot(ic2[0],ic2[1],"+",markersize=10,color='red')
plt.plot(ic3[0],ic3[1],"+",markersize=10,color='blue')

plt.text(-0.8660254-0.1, -0.5 +0.05 , "$e_1$",fontsize=12)
plt.text(+0.8660254+0.05, -0.5 +0.05 , "$e_2$",fontsize=12)
plt.text(0-0.03, 1 +0.1 , "$e_3$",fontsize=12)

for i in [0,1,2]:
    for j in [0,1,2]:
        c = A[i][j]
        plt.text(0.2*j-0.8, -0.2*i+0.9, str(c))
        plt.text(0.3-1.3,0.7,"A =")

#plt.text(0-0.03, 1 +0.1 ,A[0,0],A[0,1],A[0,2] ,fontsize=12)

plt.plot(PBd1[0], PBd1[1], ".",color='black',markersize=3)
plt.plot(PBd2[0], PBd2[1], ".",color='black',markersize=3)
plt.plot(PBd3[0], PBd3[1], ".",color='black',markersize=3)
#plt.plot(pE[0],pE[1],"+")
plt.savefig("Plots/flowportrait.pdf")

# plt.plot(ts, X2, "x", label="Foxes")
# plt.xlabel("Time")
# plt.ylabel("Population")
# plt.legend();
print([A[0,0]],[A[0,1]],[A[0,2]],[A[1,0]],[A[1,1]],[A[1,2]],[A[2,0]
],[A[2,1]],[A[2,2]])
```

```
[2] [0] [1] [1] [2] [0] [0] [1] [2]
```

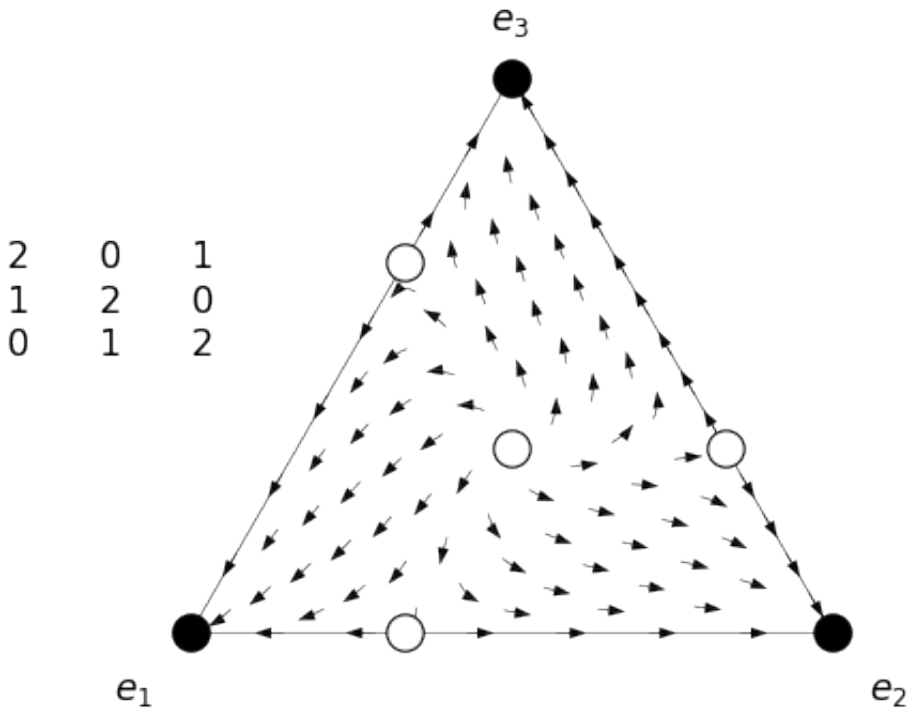


In [23]:
```
from egtplot import plot_static
```

In [ ]:
```
```

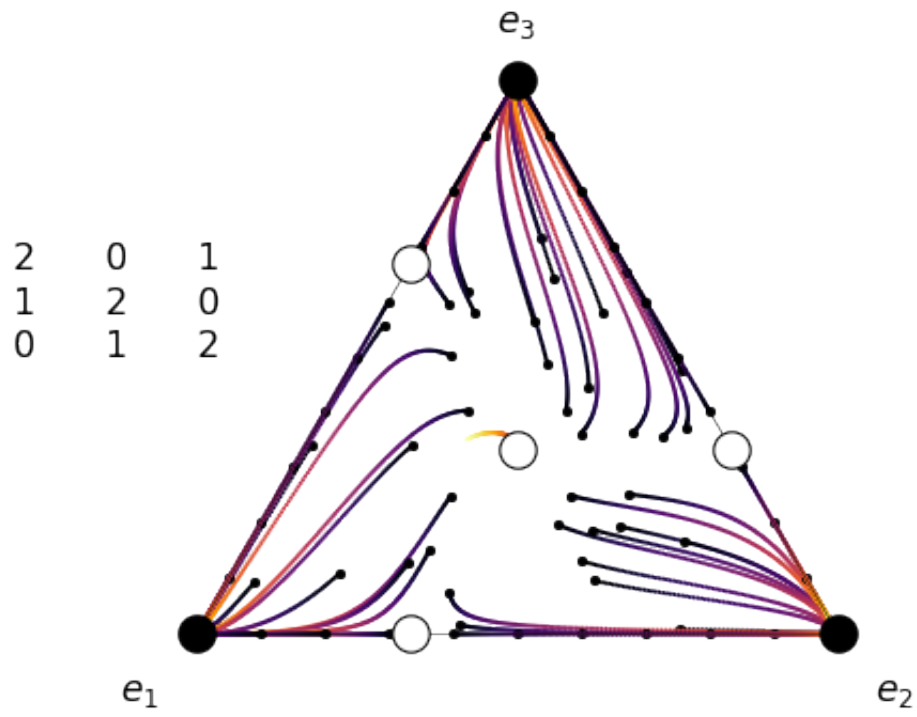
In [27]:
```
payoff_entries = [[A[0,0]],[A[0,1]],[A[0,2]],[A[1,0]],[A[1,1]],[A[1
,2]],[A[2,0]],[A[2,1]],[A[2,2]]]
simplex = plot_static(payoff_entries,eq=True,vert_labels=['$e_1$',
'$e_2$', '$e_3$'])
plt.savefig("Plots/simplex-arrowdiagram.pdf",bbox_inches="tight")
```

```
1it [00:00,  2.26it/s]
```



In [28]:
```
simplex = plot_static(payoff_entries,eq=True,ic_type='random',ic_num=40,paths=True,vert_labels=['$e_1$', '$e_2$', '$e_3$'])
plt.savefig("Plots/many-orbits.pdf",bbox_inches="tight")
```

```
1it [00:00,  2.29it/s]
```



In [ ]:

In [ ]:

In [ ]: