MarkovChain-HW

May 5, 2020

[12]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Problem1 Consider the Markov kernel for the five families living in an island, where the numbers are changed,

[4]:
```
K = np.array([[0.3, 0.6, 0.1, 0.0, 0.0], [0.2, 0.0, 0.7, 0.0, 0.1], [0.0, 0.5,␣
↪0.0, 0.5, 0.0], [0.0, 0.0, 0.4, 0.1, 0.5], [0.4, 0.1, 0.0, 0.4, 0.1]])
```

1. Calculate the five eigenvalues, and their corresponding left and right eigenvectors

[23]:
```
w,v= np.linalg.eig(K)
np.set_printoptions(precision=4)
print("eigenvalues:")
w
```

```
eigenvalues:
```

[23]:
```
array([ 1.    +0.j    ,  0.334 +0.2395j,  0.334 -0.2395j, -0.3847+0.j    ,
       -0.7833+0.j    ])
```

[24]:
```
print("right eigenvectors:")
v
```

```
right eigenvectors:
```

[24]:
```
array([[ 0.4472+0.j    ,  0.6731+0.j    ,  0.6731-0.j    ,
         0.4473+0.j    ,  0.2978+0.j    ],
       [ 0.4472+0.j    ,  0.0807+0.2351j,  0.0807-0.2351j,
        -0.5528+0.j    , -0.6409+0.j    ],
       [ 0.4472+0.j    , -0.2554+0.2015j, -0.2554-0.2015j,
         0.2545+0.j    ,  0.6197+0.j    ],
       [ 0.4472+0.j    , -0.3479-0.2228j, -0.3479+0.2228j,
         0.357 +0.j    , -0.3299+0.j    ],
       [ 0.4472+0.j    ,  0.1482-0.4321j,  0.1482+0.4321j,
        -0.5497+0.j    ,  0.0871+0.j    ]])
```

[25]:
```
w,vl= np.linalg.eig(K.T)
```

[26]:
```
print("left eigenvectors:")
vl
```

```
left eigenvectors:
```

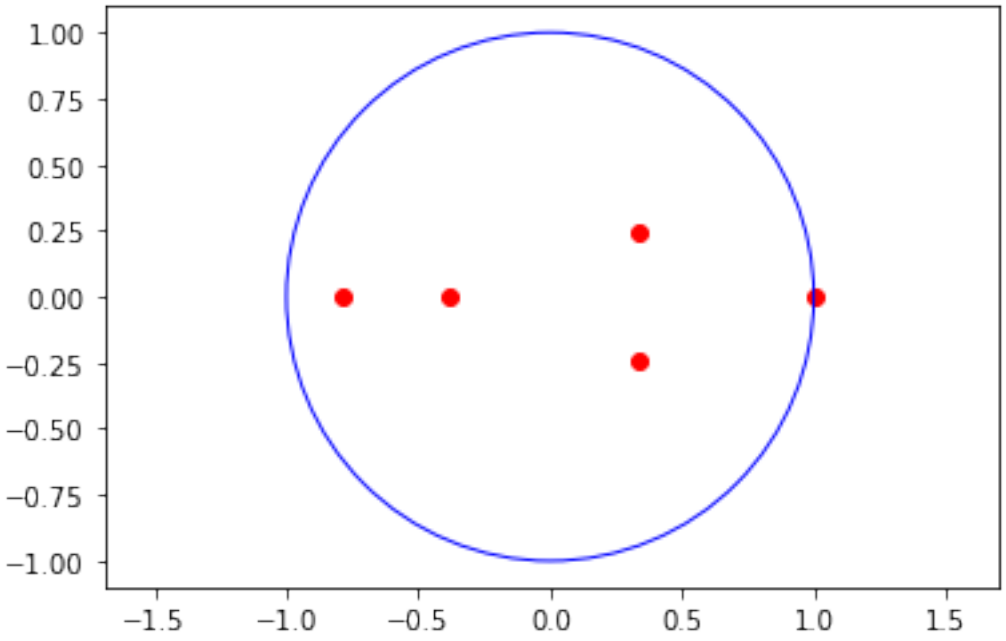
[26]:
```
array([[ 0.3236+0.j    ,  0.0009-0.4421j,  0.0009+0.4421j,
         0.0495+0.j    , -0.4251+0.j    ],
       [ 0.5118+0.j    , -0.4449-0.0741j, -0.4449+0.0741j,
         0.4221+0.j    ,  0.3314+0.j    ],
       [ 0.5732+0.j    , -0.3603+0.268j , -0.3603-0.268j ,
        -0.6516+0.j    ,  0.1427+0.j    ],
       [ 0.4564+0.j    ,  0.317 +0.2482j,  0.317 -0.2482j,
         0.5251+0.j    , -0.611 +0.j    ],
       [ 0.3104+0.j    ,  0.4872+0.j    ,  0.4872-0.j    ,
        -0.345 +0.j    ,  0.5619+0.j    ]])
```

Plot the 5 eigenvalues (complex numbers) in a 2D plane, i.e. show them as dots in a unit circle (draw the unit circle for reference).

[16]:
```
X = [x.real for x in w]
Y = [x.imag for x in w]

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
circ = plt.Circle((0, 0), radius=1, edgecolor='b', facecolor='None')
ax.add_patch(circ)
plt.scatter(X,Y, color='red')
ax.set_aspect('equal', 'datalim')

plt.show()
```



What is its invariant probability π?

[101]:
```
pi = vl[:, 0] / np.linalg.norm(vl[:, 0], ord=1)
pi = np.matrix(pi).getH().real # - or not?
print('pi:\n', pi)
```

```
pi:
 [[0.1488]
 [0.2353]
 [0.2635]
 [0.2098]
 [0.1427]]
```

What is the value of λ_{slem}?

[35]:
```
descend_abs_evals = sorted(w, key=abs, reverse=True)
lambda_slem = abs(descend_abs_evals[1].real)
#print('evals:\n', descend_abs_evals)
print(' _slem:\n', lambda_slem)
```

```
 _slem:
 0.7833305170331075
```

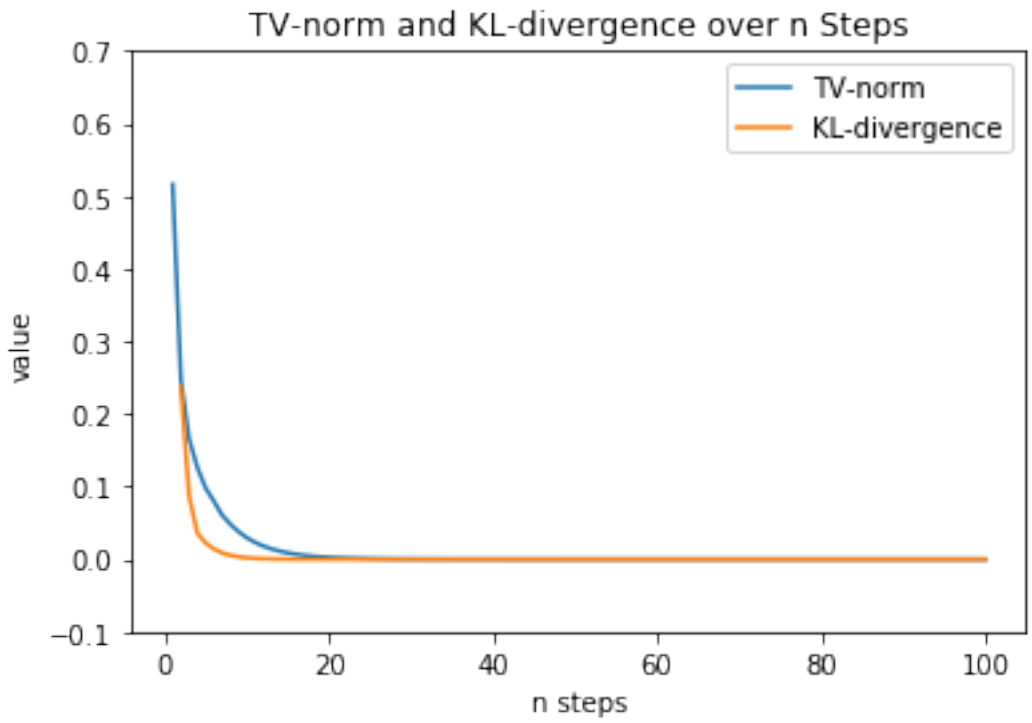
The absolute value of the second largest eigenvalue is 0.7833. λ_{slem} is 0.7833

Plot dTV(n) and dKL(n) for the first 1000 steps.

[91]:
```
v= np.array([1,0,0,0,0])
mu =v
dTV=[]
dKL=[]

for i in range(1000):
    mu = mu @ K
    dTV.append( sum(abs(mu-pi))/2  )
    dKL.append( sum (pi*np.log(pi/mu)) )
# Plot dTV(n) and dKL(n) for the first 1000 steps.
steps=range(1, 1001)
fig = plt.figure()
plt.plot(steps[:100],dTV[:100])
plt.plot(steps[:100],dKL[:100])
plt.ylim(-0.1,0.7)
plt.xlabel('n steps')
plt.ylabel('value')
plt.title('TV-norm and KL-divergence over n Steps')
plt.legend(['TV-norm', 'KL-divergence'])
fig.savefig('TV-norm-KL-divergence.png', dpi=300)
plt.show()
```

```
/Users/AppleMoony/anaconda2/envs/p36workshop/lib/python3.6/site-
packages/ipykernel_launcher.py:9: RuntimeWarning: divide by zero encountered in
true_divide
  if __name__ == '__main__':
/Users/AppleMoony/anaconda2/envs/p36workshop/lib/python3.6/site-
packages/ipykernel_launcher.py:9: RuntimeWarning: invalid value encountered in
true_divide
  if __name__ == '__main__':
/Users/AppleMoony/anaconda2/envs/p36workshop/lib/python3.6/site-
packages/ipykernel_launcher.py:9: RuntimeWarning: invalid value encountered in
multiply
  if __name__ == '__main__':
/Users/AppleMoony/anaconda2/envs/p36workshop/lib/python3.6/site-
packages/numpy/core/_asarray.py:85: ComplexWarning: Casting complex values to
real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```



3. Calculate the contraction coefficient for K.

$$A(n) = \|v_1 K^n - v_2 K^n\|_{TV} \leq C^n(K)$$

[92]:
```
KTV=[]
for i in range(5):
    for j in range(i, 5):
        KTV.append( sum(abs(K[i, ]- K[j, ]))/2  )

print("The contraction coefficient for K is : " +str(max(KTV)))
```

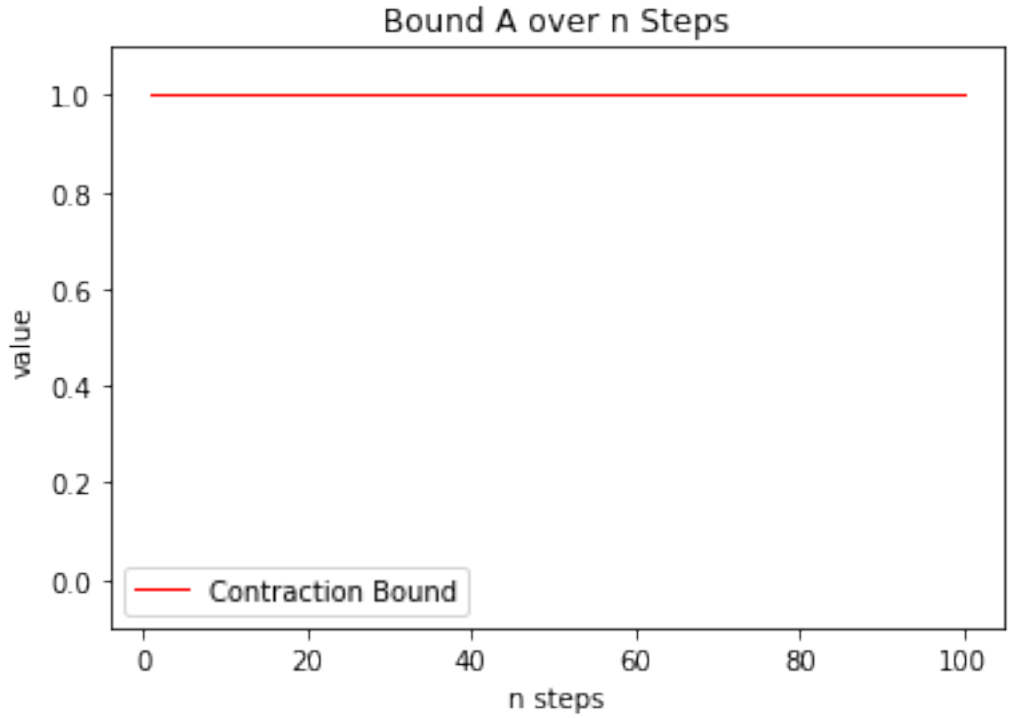
```
The contraction coefficient for K is : 1.0
```

[94]:
```
np.linalg.matrix_power(K, 200)
```

[94]:
```
array([[0.1488, 0.2353, 0.2635, 0.2098, 0.1427],
       [0.1488, 0.2353, 0.2635, 0.2098, 0.1427],
       [0.1488, 0.2353, 0.2635, 0.2098, 0.1427],
       [0.1488, 0.2353, 0.2635, 0.2098, 0.1427],
       [0.1488, 0.2353, 0.2635, 0.2098, 0.1427]])
```

[89]:
```
fig = plt.figure()
plt.plot(steps[:100], C_out[:100], color='red', linewidth=1)
plt.ylim(-0.1,1.1)
```

```
plt.xlabel('n steps')
plt.ylabel('value')
plt.title('Bound A over n Steps')
plt.legend(['Contraction Bound'])
fig.savefig('contraction_over_n_steps.png', dpi=300)
plt.show()
```



4. There is another bound – the Diaconis-Hanlon bound below,

$$B(n) = \|\pi - vK^n\|_{TV} \leq \sqrt{\frac{1-\pi(x_0)}{4\pi(x_0)}}\lambda_{slem}^n$$

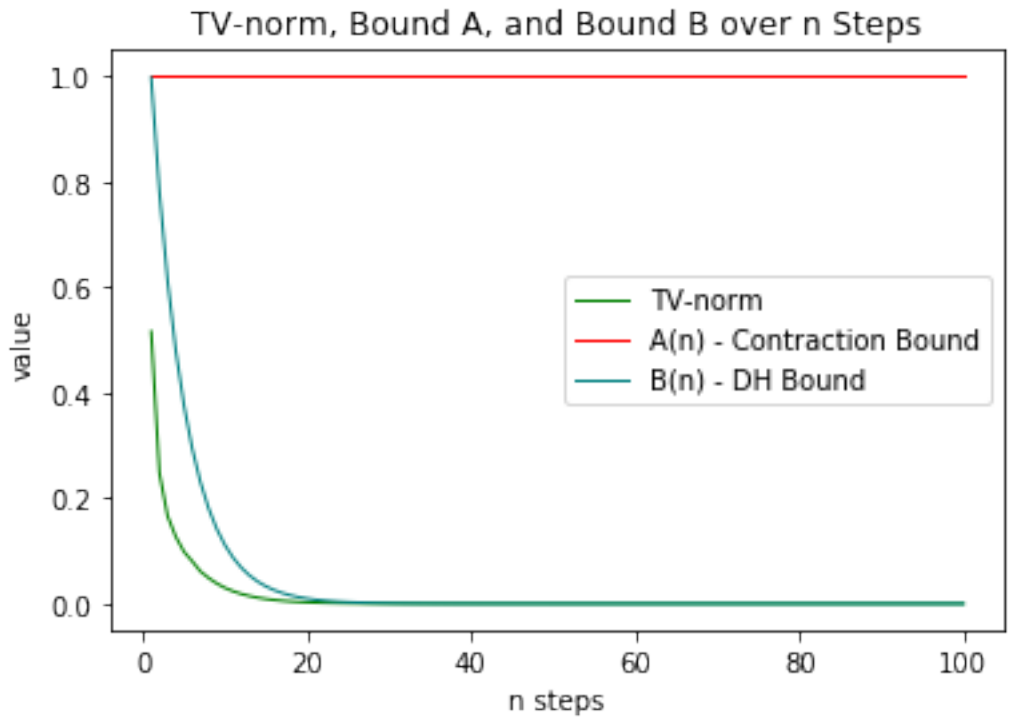
Plot the real convergence rate dTV(n) in comparison with A(n) and B(n).

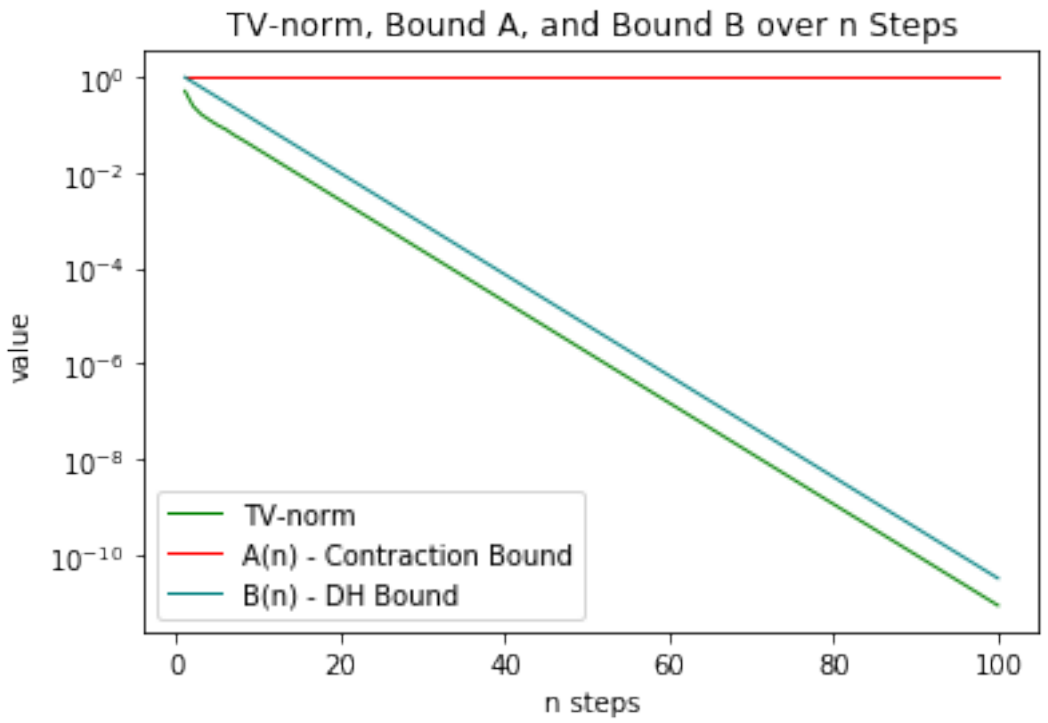
[107]:
```
dh_out=[]
x0=1

for i in range(1000):
    frac = 1 - pi[x0-1] / 4 * pi[x0-1]
    dh_out.append( float(np.sqrt(frac) * lambda_slem**i) )
```

[111]:
```
fig = plt.figure()
plt.plot(steps[:100], dTV[:100], color='green', linewidth=1)
```

```
plt.plot(steps[:100], C_out[:100], color='red', linewidth=1)
plt.plot(steps[:100], dh_out[:100], color='teal', linewidth=1)
plt.xlabel('n steps')
plt.ylabel('value')
plt.title('TV-norm, Bound A, and Bound B over n Steps')
plt.legend(['TV-norm', 'A(n) - Contraction Bound', 'B(n) - DH Bound'])
fig.savefig('TV-norm-A-B-bounds.png', dpi=300)
plt.show()
fig = plt.figure()
plt.plot(steps[:100], dTV[:100], color='green', linewidth=1)
plt.plot(steps[:100], C_out[:100], color='red', linewidth=1)
plt.plot(steps[:100], dh_out[:100], color='teal', linewidth=1)
plt.yscale('log')
plt.xlabel('n steps')
plt.ylabel('value')
plt.title('TV-norm, Bound A, and Bound B over n Steps')
plt.legend(['TV-norm', 'A(n) - Contraction Bound', 'B(n) - DH Bound'])
fig.savefig('TV-norm-A-B-bounds.png', dpi=300)
plt.show()
```



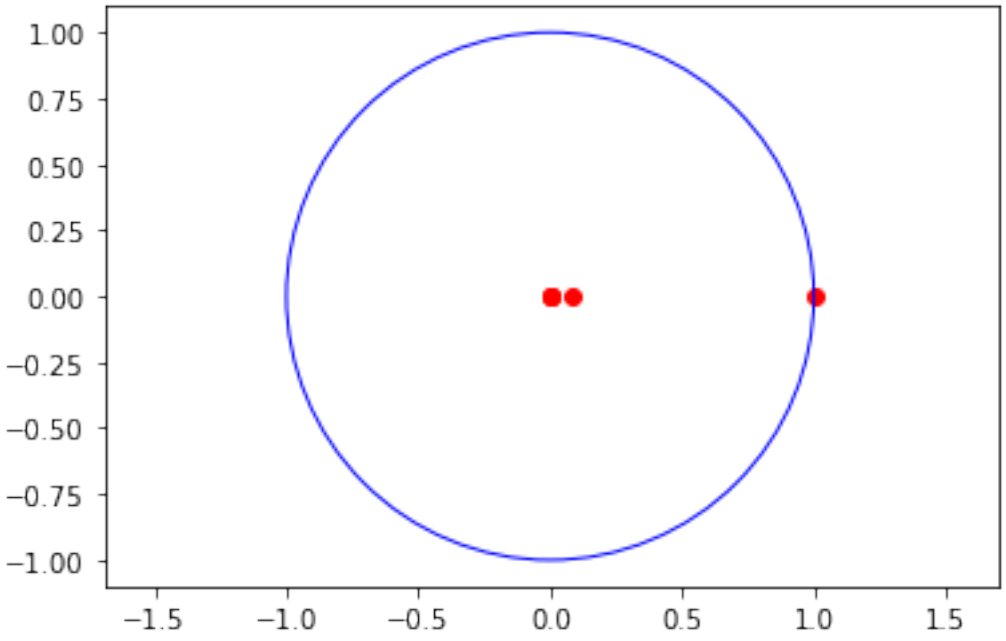


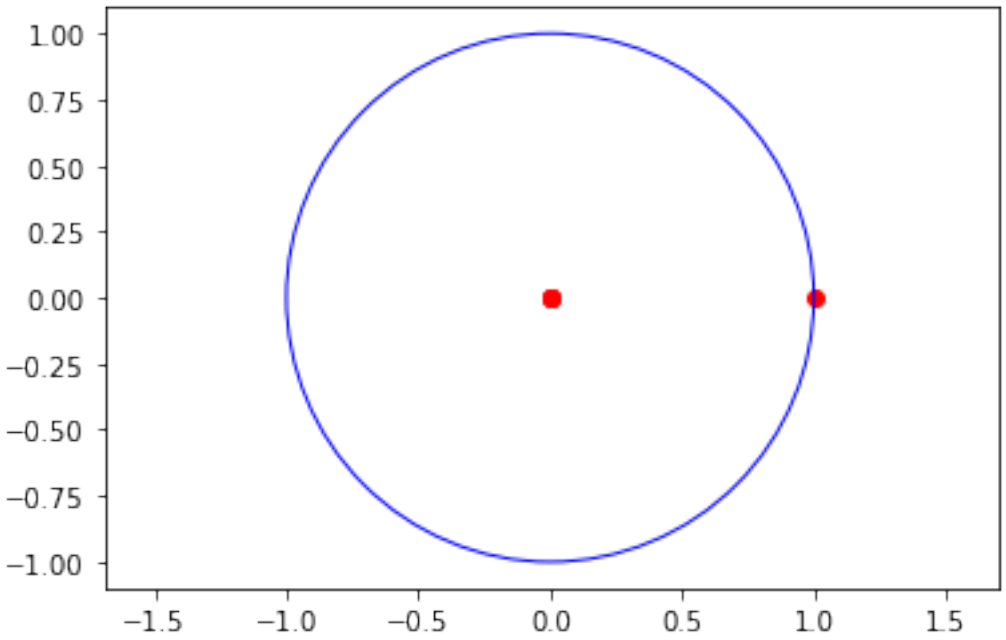
5. We define a new Markov chain with transition kernel $P = K^n$. Then draw the 5 eigenvalues of P on a 2D complex plane as you did in (1). Show how these eigenvalues move on the plane at three stages: n = 10, 100, 1000. You'd better draw the traces of the 5 dots (link the movements of the 5 dots to show their trajectories).

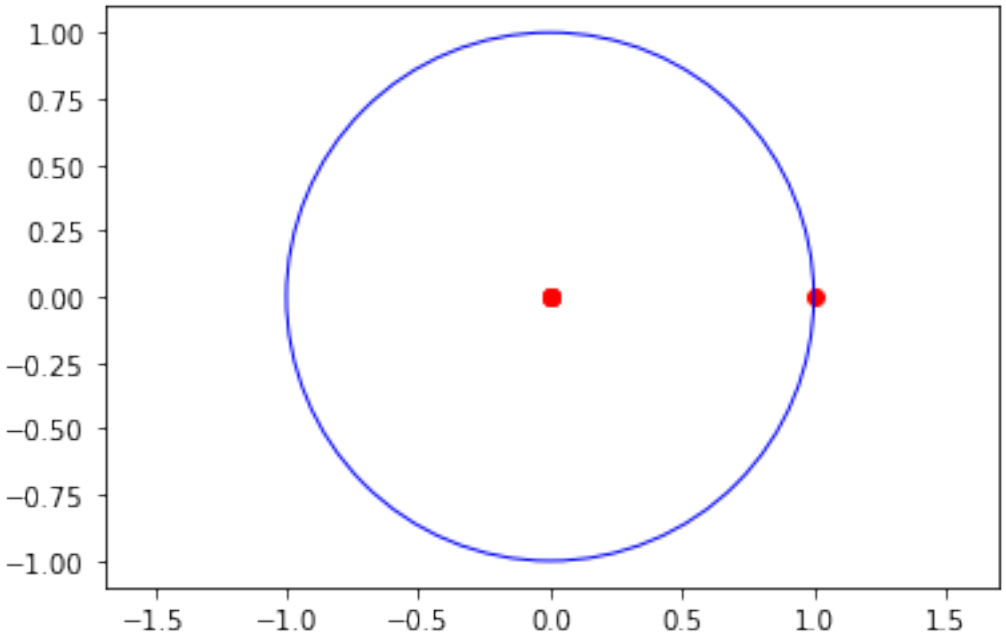
[133]:
```
n_list=[10, 100, 1000]
X_list=[]
Y_list=[]

for n in n_list:
    P=np.linalg.matrix_power(K, n)
    P_1000=P
    w,v=np.linalg.eig(P)
    X = [x.real for x in w]
    Y = [x.imag for x in w]
    X_list.append(X)
    Y_list.append(Y)
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    circ = plt.Circle((0, 0), radius=1, edgecolor='b', facecolor='None')
    ax.add_patch(circ)
    plt.scatter(X, Y, color='red')
    ax.set_aspect('equal', 'datalim')
```

```
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5



Printout the matrix P for n = 1000 and see whether it becomes the "ideal" transition kernel.

[134]:
```
print(P_1000)
```

```
[[0.1488 0.2353 0.2635 0.2098 0.1427]
 [0.1488 0.2353 0.2635 0.2098 0.1427]
 [0.1488 0.2353 0.2635 0.2098 0.1427]
 [0.1488 0.2353 0.2635 0.2098 0.1427]
 [0.1488 0.2353 0.2635 0.2098 0.1427]]
```

Problem2 Now we consider two more transition matrices

[141]:
```
K1 = np.matrix([[0.1, 0.4, 0.3, 0.0, 0.2], [0.5, 0.3, 0.2, 0.0, 0.0 ], [0.0, 0.
↪4, 0.5, 0.1, 0.0 ], [ 0.0, 0.0, 0.0, 0.5, 0.5 ], [0.0, 0.0, 0.0, 0.7, 0.3] ])
```

[142]:
```
K2 = np.matrix([[0.0, 0.0, 0.0, 0.4, 0.6], [0.0, 0.0, 0.0, 0.5, 0.5], [0.0, 0.
↪0, 0.0, 0.9, 0.1 ], [0.0, 0.2, 0.8, 0.0, 0.0], [0.3, 0.0, 0.7, 0.0, 0.0] ])
```

1. Are K1 and K2 irreducibly, aperiodic?

K1 and K2 are not irreducibly, aperiodic.

K1 is not irreducible since once we reach states 4 or 5, it's impossible to go from state 4 and state 5 to any other states. Hence, K1 is reducible.

K2 is not aperiodic since it has a period is 2. From any one of the state of this markov chain, it takes a multiple of 2 steps to return to itself.

2. Printout the 5 eigenvalues and 5 eigenvectors of the two matrices.

[150]:
```
w1,v1= np.linalg.eig(K1.T)
np.set_printoptions(precision=4)
print("The eigenvalues of K1:")
w1
```

The eigenvalues of K1:

[150]:
```
array([ 1.    ,  0.9152, -0.2   , -0.1977,  0.1824])
```

[151]:
```
print("The left eigenvectors of K1:")
v1
```

The left eigenvectors of K1:

[151]:
```
matrix([[-2.4241e-16, -2.5991e-01,  1.4754e-14,  1.3002e-02, -6.3229e-01],
        [-8.1742e-16, -4.2377e-01, -8.4463e-15, -7.7402e-03, -1.0423e-01],
        [-1.1547e-15, -3.9189e-01, -3.3423e-15, -3.3721e-03,  6.6293e-01],
        [-8.1373e-01,  6.3996e-01,  7.0711e-01, -7.0797e-01,  3.0805e-01],
        [-5.8124e-01,  4.3560e-01, -7.0711e-01,  7.0608e-01, -2.3446e-01]])
```

[152]:
```
w2,v2= np.linalg.eig(K2.T)
np.set_printoptions(precision=4)
print("The eigenvalues of K2:")
w2
```

The eigenvalues of K2:

[152]:
```
array([ 1.0000e+00, -1.0000e+00, -2.6458e-01,  2.6458e-01,  1.0372e-16])
```

[153]:
```
print("The left eigenvectors of K2:")
v2
```

The left eigenvectors of K2:

[153]:
```
matrix([[-5.0412e-02, -5.0412e-02,  5.6695e-01, -5.6695e-01,  6.1721e-01],
        [-1.4003e-01, -1.4003e-01, -3.7796e-01,  3.7796e-01, -7.7152e-01],
        [-6.7777e-01, -6.7777e-01, -1.8898e-01,  1.8898e-01,  1.5430e-01],
        [-7.0017e-01,  7.0017e-01,  5.0000e-01,  5.0000e-01, -1.1186e-16],
        [-1.6804e-01,  1.6804e-01, -5.0000e-01, -5.0000e-01,  2.4490e-16]])
```

3. How many and what are the invariant probabilities for each matrix?

[160]:
```
pi_K1 = v1[:, 0] / np.linalg.norm(v1[:, 0], ord=1)
left0_K1 = np.matrix(pi_K1).getH()
print('left0_K1:', left0_K1*(-1))
```

```
left0_K1: [[1.7377e-16 5.8598e-16 8.2777e-16 5.8333e-01 4.1667e-01]]
```

[161]:
```
pi_K2 = v2[:, 0] / np.linalg.norm(v2[:, 0], ord=1)
left0_K2 = np.matrix(pi_K2).getH()
print('left0_K2:', left0_K2*(-1))
```

```
left0_K2: [[0.029  0.0806 0.3903 0.4032 0.0968]]
```

The invariant probabilities for each matrix is unique. K2 is irreducible but not aperiodic. When P is irreducible (but not necessarily aperiodic), then π still exists and is unique, but the Markov chain does not necessarily converge to π from every starting state. Notice K2 has eigenvalues of 1 and -1. So there is another eigenvalue of magnitute 1.

Problem 3. A Markov chain returning time $\tau_{ret}(i)$ is the minimum steps that a Markov chain returns to state i after leaving i. Calculate the probability for returning to state 0 in finite step.

$$\begin{aligned} Prob(\tau_{ret}(0) < \infty) &= \sum_{\tau(0)=1}^{\infty} Prob(\tau(0)) \\ &= (1-\alpha) + \alpha(1-\alpha) + \alpha^2(1-\alpha) + \dots \\ &= (1-\alpha) \times \frac{1}{1-\alpha} \\ &= 1 \end{aligned}$$

The markov chain is finite, irreducible and aperiodic. So it has a unique stationary distribution. Since from each state, the probability of going to the state 0 is $1-\alpha$. We have the following relationship,

$$\sum_{i=0}^{\infty} \pi(i)(1-\alpha) = \pi(0)$$

Since $\sum_{i=0}^{\infty} \pi(i) = 1$, we get $\pi(0) = 1-\alpha$

We find from π that the mean recurrence time (i.e. the expected time to return) for the state 0, $E[\tau_{ret}(0)]$, is $1/\pi(0) = \frac{1}{1-\alpha}$

Problem 4. Prove that $KL(\pi||v) \leq KL(\pi||\mu)$

First, show $KL_{X,Y}(\pi(X)P(X,Y)||\eta(X)P(X,Y)) = KL_X(\pi(X)||\eta(X))$ for any distribution η.

$$KL_{X,Y}(\pi(X)P(X,Y)||\eta(X)P(X,Y)) = \sum_X \sum_Y \pi(X)P(X,Y)\log\frac{\pi(X)P(X,Y)}{\eta(X)P(X,Y)}$$
$$= \sum_X \pi(X)\log\frac{\pi(X)}{\eta(X)}$$
$$= KL_X(\pi(X)||\eta(X))$$

Next, show that $KL(\pi||\mu) - KL(\pi||v) = E_{Y\,\pi}[KL_X(P(Y,X)||Q(Y,X))]$ where

$$Q(Y,X) = \frac{P(X,Y)\mu(X)}{v(Y)}$$

Using the above proved result, we have

$$KL_Y(\pi(Y)||v(Y)) = KL_{X,Y}(\pi(Y)P(Y,X)||v(Y)P(Y,X))$$
$$KL_X(\pi(X)||\mu(X)) = KL_{X,Y}(\pi(X)P(X,Y)||\mu(X)P(X,Y))$$

Subtracting the two equations, we get

$$\begin{aligned}
&KL_X(\pi(X)||\mu(X)) - KL_Y(\pi(Y)||v(Y)) \\
=&KL_{X,Y}(\pi(X)P(X,Y)||\mu(X)P(X,Y)) - KL_{X,Y}(\pi(Y)P(Y,X)||v(Y)P(Y,X)) \\
=&\sum_X\sum_Y \pi(X)P(X,Y)\log\frac{\pi(X)P(X,Y)}{\mu(X)P(X,Y)} - (\sum_X\sum_Y \pi(Y)P(Y,X)\log\frac{\pi(Y)P(Y,X)}{v(Y)P(Y,X)}) \\
=&\sum_X\sum_Y \pi(Y)P(Y,X)\log\frac{\pi(Y)P(Y,X)}{\mu(X)P(X,Y)} - (\sum_X\sum_Y \pi(Y)P(Y,X)\log\frac{\pi(Y)P(Y,X)}{v(Y)P(Y,X)}) \\
=&\sum_X\sum_Y \pi(Y)P(Y,X)\log\frac{v(Y)P(Y,X)}{\mu(X)P(X,Y)} \\
=&\sum_Y\sum_X \pi(Y)P(Y,X)\log\frac{P(Y,X)}{Q(Y,X)} \\
=&E_{Y\sim\pi}[KL_X(P(Y,X)||Q(Y,X))]
\end{aligned}$$

Here we used the fact that $\pi(X)P(X,Y) = \pi(Y)P(Y,X)$ since P satisfies detailed balance. Also $Q(Y,X) = \frac{P(X,Y)\mu(X)}{v(Y)}$.

Now we will show, $KL_{X,Y}(P(Y,X)||Q(Y,X)) \geq 0$

$$\begin{aligned}
-KL_{X,Y}(P(Y,X)||Q(Y,X)) &= -\sum_{X,Y} P(Y,X)\log\frac{P(Y,X)}{Q(Y,X)} \\
&= \sum_{X,Y} P(Y,X)\log\frac{Q(Y,X)}{P(Y,X)} \\
&\leq \log\sum_{X,Y} P(Y,X)\frac{Q(Y,X)}{P(Y,X)} \\
&\leq \log(1) \\
&\leq 0
\end{aligned}$$

We have used Jensen's inequality and the fact that log is a concave function.

Since $KL_{X,Y}(P(Y,X)||Q(Y,X)) \geq 0$, the expected value $E_{Y\sim\pi}[KL_X(P(Y,X)||Q(Y,X))] \geq 0$

Thus, $KL_X(\pi(X)||\mu(X)) - KL_Y(\pi(Y)||v(Y)) = E_{Y\sim\pi}[KL_X(P(Y,X)||Q(Y,X))] \geq 0$

Hence, $KL(\pi||v) \leq KL(\pi||\mu)$

[ ]: