Q a. Recall the Canonical form

$$A(\theta) X_t = B(\theta) E_t X_{t+1} + C(\theta) X_{t-1} + D(\theta) \eta_t$$

put $X_t = [x_t, \pi_t, i_t, g_t, u_t]'$ $\quad \eta_t = [\varepsilon_{it}, \varepsilon_{gt}, \varepsilon_{ut}]'$

$$A(\theta) = \begin{bmatrix} 1 & 0 & \sigma & -1 & 0 \\ -K & 1 & 0 & 0 & -1 \\ (\rho_i - 1)\phi_x & (\rho_i - 1)\phi_\pi & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B(\theta) = \begin{bmatrix} 1 & \sigma & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C(\theta) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \rho_i & 0 & 0 \\ 0 & 0 & 0 & \rho_g & 0 \\ 0 & 0 & 0 & 0 & \rho_u \end{bmatrix}$$

$$D(\theta) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \sigma_i & 0 & 0 \\ 0 & \sigma_g & 0 \\ 0 & 0 & \sigma_u \end{bmatrix}$$

X_t, η_t
$A(\theta), B(\theta), C(\theta), D(\theta)$ constitute a system of expectational DE.

Q. 6 conjecture the solution form as $X_t = F(\theta) X_{t-1} + G(\theta) \eta_t$

Then $E_t X_{t+1} = F(\theta) X_t$

plug it into the expectational DE we derived in a.

$A(\theta) X_t = B(\theta) E_t X_{t+1} + C(\theta) X_{t-1} + D(\theta) \eta_t$

for convenience drop θ hereon

$\Rightarrow A(\theta) X_t - B(\theta) \cdot F(\theta) X_t = C(\theta) X_{t-1} + D(\theta) \eta_t$

$\Rightarrow (A - BF) X_t = C X_{t-1} + D \eta_t$

$\Rightarrow X_t = (A - BF)^{-1} C X_{t-1} + (A - BF)^{-1} D \eta_t$

$\therefore$ we have $F = (A - BF)^{-1} C$

$G = (A - BF)^{-1} D$

since we know A, B, C, D, it suffices to solve for F.

solve for F.

STEP 1. stack the expectational DE in first order form.

$$\underbrace{\begin{bmatrix} 0_{5\times5} & I_5 \\ -C & A \end{bmatrix}}_{K} \begin{bmatrix} X_{t-1} \\ X_t \end{bmatrix} = \underbrace{\begin{bmatrix} I_5 & 0_{5\times5} \\ 0_{5\times5} & B \end{bmatrix}}_{L} E_t \begin{bmatrix} X_t \\ X_{t+1} \end{bmatrix} + \begin{bmatrix} \underset{5\times3}{0} & \underset{5\times3}{0} \\ \underset{5\times3}{0} & \underset{5\times3}{D} \end{bmatrix} \begin{bmatrix} \eta_{t-1} \\ \eta_t \end{bmatrix}$$

STEP 2. schur decomposition of K & L

$K = QTZ'$

$L = QSZ'$

where T, S are upper triangular

then $F = Z_{21} Z_{11}^{-1}$

will compute this using a computer.

```
# Calculate F
'''
Z_21: from row6, upto column 5
'''
Z_21 = Z[5:,:5]
Z_11 = Z[:5,:5]
F = np.matmul(Z_21, inv(Z_11))
F
```

← calculated result. code is attached below

```
array([[ 0.        ,  0.        , -3.12895561,  1.98531639, -1.16139252],
       [ 0.        ,  0.        , -1.13918312,  0.55776102,  1.51891083],
       [ 0.        ,  0.        ,  0.59393949,  0.16118511,  0.27474735],
       [ 0.        ,  0.        ,  0.        ,  0.8       ,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.        ,  0.8       ]])
```

```
# Calculate G
G = np.matmul(inv(A-B@F), D)
G
```

```
array([[-1.73830867,  2.48164548, -1.45174065],
       [-0.63287951,  0.69720127,  1.89863853],
       [ 0.32996638,  0.20148139,  0.34343419],
       [ 0.        ,  1.        ,  0.        ],
       [ 0.        ,  0.        ,  1.        ]])
```

In [4]:
```
# Parameters
sigma, kappa, beta = 1, 0.15, 0.99
pi_pi, pi_x = 2, 0.25
roh_i, roh_g, roh_u = 0.9, 0.8, 0.8
sigma_i, sigma_g, sigma_u = 0.5, 1, 1
```

In [7]:
```
# Matrices
A = np.array([
    [1, 0, sigma, -1, 0],
    [-kappa, 1, 0, 0, -1],
    [(roh_i-1)*pi_x, (roh_i-1)*pi_pi, 1, 0, 0],
    [0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1]
    ])

B = np.array([
    [1, sigma, 0, 0, 0],
    [0, beta, 0, 0, 0],
    [0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0]
    ])

C = np.array([
    [0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0],
    [0, 0, roh_i, 0, 0],
    [0, 0, 0, roh_g, 0],
    [0, 0, 0, 0, roh_u]
    ])

D = np.array([
    [0, 0, 0],
    [0, 0, 0],
    [sigma_i, 0, 0],
    [0, sigma_g, 0],
    [0, 0, sigma_u]
    ])
```

In [42]:
```
# Schur Decomposition Using Ordqz
'''
input: K and L --> first order stacked
output: Q and Z --> left and right Schur Vectors
'''
zero_55 = np.zeros((5,5))
zero_53 = np.zeros((5,3))

K = np.vstack(( np.hstack((zero_55, np.identity(5))),
                np.hstack((-C, A))
              ))

L = np.vstack(( np.hstack((np.identity(5), zero_55)),
                np.hstack((zero_55, B))
              ))

T, S, a, b, Q, Z =  ordqz(K,L, sort='iuc')
```

Q. C

Transition Eqn : $X_t = F X_{t-1} + G \eta_t$

Measurement Eqn : $y_t = H X_t + J \cdot v_t$

$y_t = [\, x_t^{obs}, \ \pi_t^{obs}, \ i_t^{obs} \,]'$

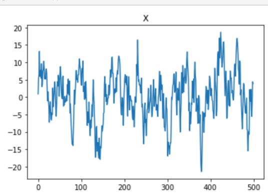
$$H = \begin{bmatrix} 1, & 0, & 0, & 0, & 0 \\ 0, & 1, & 0, & 0, & 0 \\ 0, & 0, & 1, & 0, & 0 \end{bmatrix}$$

$J = \Delta_m \cdot M$

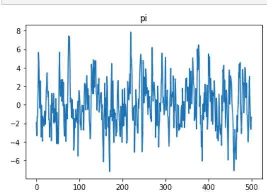
$v_t = [\, v_{xt}, \ v_{\pi t}, \ v_{it} \,]'$

In [105]:
```python
# Generate Data Using State Space Representation
Simulated = []
Simulated_x = []
Simulated_pi = []
Simulated_i = []


X_old = np.array([0, 0, 0, 0, 0])
Simulated.append(X_old)
for i in range(500):
    X_new = F@X_old.reshape(5,1) + G@np.random.normal(0,1, size = (3,1))

    Simulated.append(X_new)
    Simulated_x.append(X_new[0][0])
    Simulated_pi.append(X_new[1][0])
    Simulated_i.append(X_new[2][0])

    X_old = X_new
```

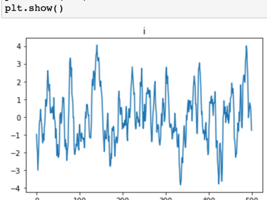
]:
```python
simulated_data['x'].plot()
plt.title("x")
plt.show()
```



]:
```python
simulated_data['pi'].plot()
plt.title("pi")
plt.show()
```



]:
```python
simulated_data['i'].plot()
plt.title("i")
plt.show()
```



Q.d

```
Roadmap
1. generate measurement data
2. using Kalman Filter, generate y_t|t-1 and V_t|t-1 recursively
   --> in the same code update the likelihood
```

In [167]:

```python
# STEP1: Generate Mearsurement Data
H = np.array([
    [1, 0, 0, 0, 0],
    [0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0],
    ])
J = 0.2*np.identity(3)

Simulated_y = []
for i in range(501):
    target = H@Simulated[i].reshape(5,1)+J@np.random.normal(0,1, size = (3,1))
    Simulated_y.append(target)
```

In [177]:

```python
# STEP2: Kalman Filter

# initial value
X_update = np.zeros((5,1))
P_update = np.identity(5)


first_moment = []
second_moment = []
likelihood = 0

# update
for i in range(500):
    X_estimate = F@X_update
    P_estimate = F@P_update@F.T + G@G.T
    Y_estimate = H@X_estimate
    V_estimate = H@P_estimate@H.T + J@J.T

    likelihood = likelihood - (1/2)*(ln(np.absolute(V_estimate))
                                 + (Simulated_y[i+1]-Y_estimate).T@inv(V_estimate)@(Simulated_y[i+1]-Y_estimate))
    X_update = X_estimate + P_estimate@H.T@inv(V_estimate)@(Simulated_y[i+1]-Y_estimate)
    P_update = P_estimate - P_estimate@H.T@inv(V_estimate)@H@P_estimate

    first_moment.append(Y_estimate)
    second_moment.append(V_estimate)
```

In [178]:

```python
likelihood
```

Out[178]:

```
array([[-1349.84903358,   -46.61469171,  -608.32951811],
       [  -46.61469171, -1120.6457738 ,  -606.72756591],
       [ -608.32951811,  -606.72756591,  -467.09374145]])
```