In [48]:
```
!pip install --upgrade cvxpy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: cvxpy in /usr/local/lib/python3.7/dist-packages (1.2.1)
Requirement already satisfied: osqp>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from cvxpy) (0.6.2.post0)
Requirement already satisfied: ecos>=2 in /usr/local/lib/python3.7/dist-packages (from cvxpy) (2.0.10)
Requirement already satisfied: scs>=1.1.6 in /usr/local/lib/python3.7/dist-packages (from cvxpy) (3.2.0)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages (from cvxpy) (1.21.6)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from cvxpy) (1.4.1)
Requirement already satisfied: qdldl in /usr/local/lib/python3.7/dist-packages (from osqp>=0.4.1->cvxpy) (0.1.5.
post2)
```

In [59]:
```
import numpy as np
import cvxpy as cvx
import matplotlib.pyplot as plt
from scipy.linalg import solve_discrete_are
from tqdm.auto import tqdm
from itertools import product
import matplotlib.pyplot as plt
```

In [60]:
```
def generate_ellipsoid_points(M, num_points=100):
    L = np.linalg.cholesky(M)
    θ = np.linspace(0, 2*np.pi, num_points)
    u = np.column_stack([np.cos(θ), np.sin(θ)])
    x = u @ L.T
    return x
```

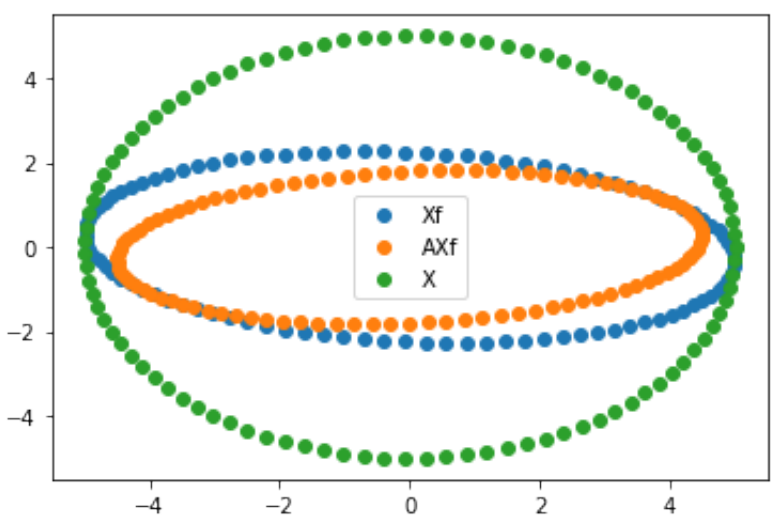
In [61]:
```
def generate_circle_points(num_points=100, rx = 5):
    θ = np.linspace(0, 2*np.pi, num_points)
    x = rx * np.column_stack([np.cos(θ), np.sin(θ)])
    return x
```

In [62]:
```
n,  m = 2, 1
N = 4
A = np.array([[0.9, 0.6],[0, 0.8]])
B = np.array([[0],[1]])

Q = np.eye(n)
R = np.eye(m)

rx = 5
ru = 1
```

In [63]:
```
# P5 (d) Solve the SDP for M

M_cvx = cvx.Variable((n,n), symmetric = True)

objective = cvx.log_det(M_cvx)

constraints = [M_cvx >> 0,
               M_cvx - rx**2 * np.eye(2) << 0,
               cvx.bmat([[M_cvx, A@M_cvx],[M_cvx@A.T, M_cvx]]) >> 0]

prob = cvx.Problem(cvx.Maximize(objective), constraints)
prob.solve()


M = M_cvx.value
status = prob.status
print(M)
P = np.linalg.inv(M)
print(P)
```

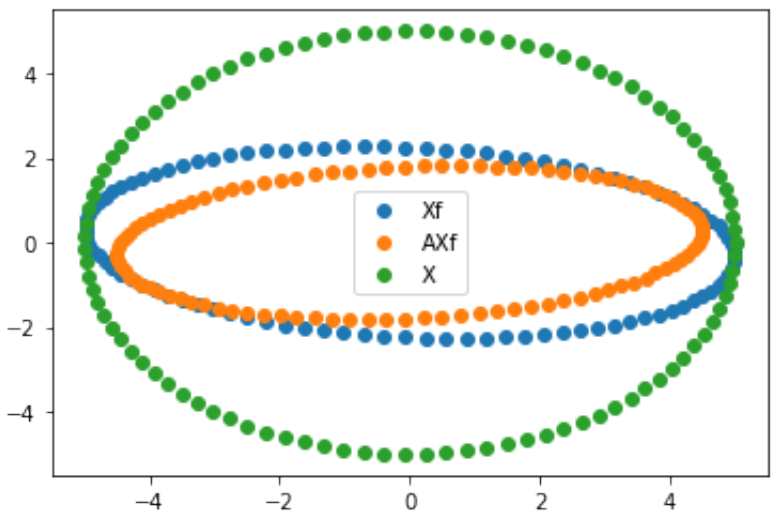
```
[[24.8640275  -1.64239115]
 [-1.64239115  5.15785901]]
[[0.04108286 0.01308181]
 [0.01308181 0.19804447]]
```

In [64]:
```
Xf = generate_ellipsoid_points(M, num_points=100)
AXf = A @ Xf.T
X = generate_circle_points()
plot1 = plt.scatter(Xf[:,0], Xf[:,1])
plot2 = plt.scatter(AXf[0,:], AXf[1,:])
plot3 = plt.scatter(X[:,0], X[:,1])
plt.legend((plot1, plot2, plot3), ('Xf', 'AXf', 'X'))
plt.show()
plt.savefig('P5_d.png', bbox_inches='tight')
```




```
<Figure size 432x288 with 0 Axes>
```

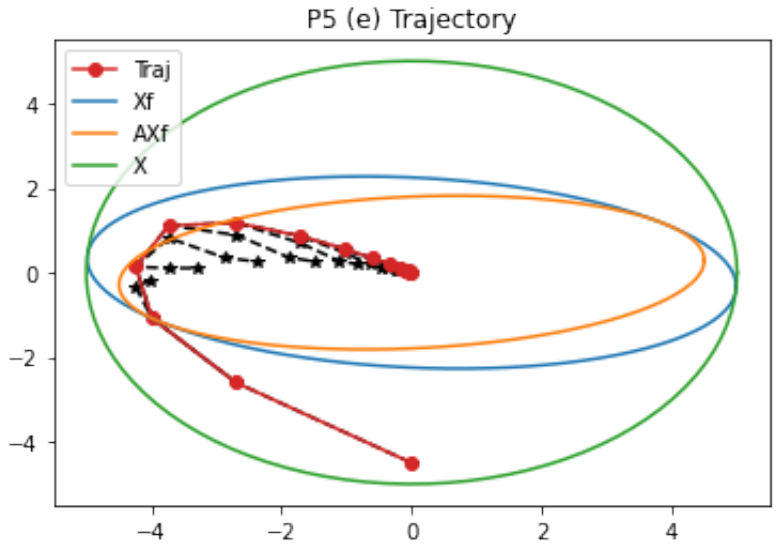
In [65]:
```
AXf = generate_ellipsoid_points(A @ M @ A.T, num_points=100)
plot1 = plt.scatter(Xf[:,0], Xf[:,1])
plot2 = plt.scatter(AXf[:,0], AXf[:,1])
plot3 = plt.scatter(X[:,0], X[:,1])
plt.legend((plot1, plot2, plot3), ('Xf', 'AXf', 'X'))
plt.show()
plt.savefig('P5_d.png', bbox_inches='tight')
```




```
<Figure size 432x288 with 0 Axes>
```

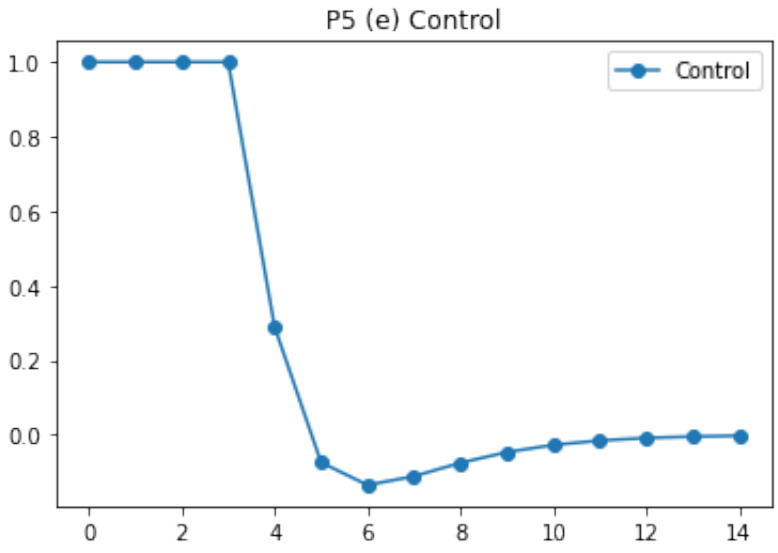
In [66]:
```
def do_mpc(N, x0, A, B, P, Q, R, rx, ru):

  x_cvx = cvx.Variable((N + 1, n))
  u_cvx = cvx.Variable((N, m))

  cost = cvx.quad_form(x_cvx[N],P)

  constraints = [x_cvx[0] == x0]

  for k in range(N):
      cost += cvx.quad_form(x_cvx[k],Q) + cvx.quad_form(u_cvx[k],R)
      constraints += [x_cvx[k+1] == A@x_cvx[k] + B@u_cvx[k]]
      constraints += [cvx.norm(x_cvx[k], 2) <= rx, cvx.norm(u_cvx[k], 2) <= ru]

  constraints += [cvx.norm(x_cvx[N], 2) <= rx]


  prob = cvx.Problem(cvx.Minimize(cost), constraints)
  prob.solve()


  x = x_cvx.value
  u = u_cvx.value
  status = prob.status

  return x, u, status
```

In [67]:
```
T = 15
x0 = np.array([0, -4.5])
x = np.copy(x0)
x_mpc = np.zeros((T, N + 1, n))
u_mpc = np.zeros((T, N, m))
for t in range(T):
  x_mpc[t], u_mpc[t], status = do_mpc(N, x, A, B, P, Q, R, rx, ru)
  if status == 'infeasible':
      x_mpc = x_mpc[:t]
      u_mpc = u_mpc[:t]
      break
  x = A@x + B@u_mpc[t, 0, :]
  plt.plot(x_mpc[t, :, 0], x_mpc[t, :, 1], '--*', color='k')
plt.plot(x_mpc[:, 0, 0], x_mpc[:, 0, 1], '-o', label = 'Traj', color = 'C3')
plt.title('P5 (e) Trajectory')

Xf = generate_ellipsoid_points(M, num_points=100)
AXf = generate_ellipsoid_points(A @ M @ A.T, num_points=100)
X = generate_circle_points()
plt.plot(Xf[:,0], Xf[:,1],'-', label = 'Xf', color = 'C0')
plt.plot(AXf[:,0], AXf[:,1],'-', label = 'AXf',color = 'C1')
plt.plot(X[:,0], X[:,1],'-', label = 'X',color = 'C2')

plt.legend()
plt.show()
plt.savefig('P5_e_Traj.png', bbox_inches='tight')
```



<Figure size 432x288 with 0 Axes>

In [68]:
```
plt.plot(u_mpc[:,0], '-o', label = 'Control', color = 'C0')
plt.title('P5 (e) Control')

plt.legend()
plt.show()

plt.savefig('P5_e_Control.png', bbox_inches='tight')
```



<Figure size 432x288 with 0 Axes>