i1 : initial condition P1 (float)

i2 : initial condition P2 (float)

u1 : matrix of actions for P1 (array of floats)

u2 : matrix of actions for P2 (array of floats)

J1 : cost function for P1 (Python function)

J2 : cost function for P2 (Python function)

alpha : learning rate (float)

e : convergence error (float)

BR\_play(i1, i2, u1, u2, J1, J2, alpha, error):

u1k = i1

u2k = i2

while(True):

u1k+1 = u1k + alpha\*(BR(u1, u2k, J1)- u1k)

u2k+1 = u2k + alpha\*(BR(u1k, u2, J2)- u2k)

if (abs(u1k+1 - u1k) < error) and (abs(u2k+1 - u2k) < error):

return (u1k+1, u2k+1)

else:

u1k , u2k = u1k+1, u2k+1

Gradient\_play(i1, i2, u1, u2, J1, J2, alpha, error):

u1k = i1

u2k = i2

while(True):

u1k+1 = u1k - alpha\*grad(u1k, u2k, J1)

u2k+1 = u2k - alpha\*grad(u2k, u1k, J2)

if (abs(u1k+1 - u1k) < error) and (abs(u2k+1 - u2k) < error):

return (u1k+1, u2k+1)

else:

u1k , u2k = u1k+1, u2k+1