StrategicInfluence6a

July 14, 2021

1 Experimentation with Strategic Influence Network Model, Part 6-2

1.0.1 Verification of steady-state vs optimal trajectories with extra results

James Yu
14 July 2021

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

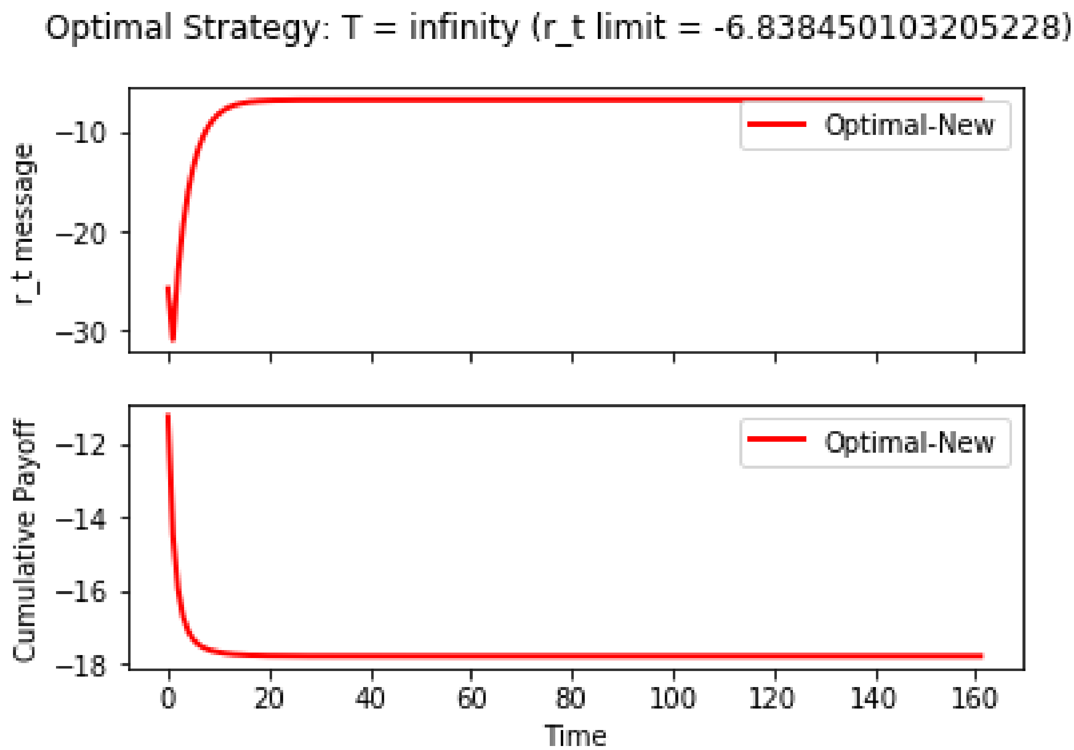
1.0.2 First, computation of necessary values:

[2]:
```
def optimal_K(z, delta = 0.8):
    A = np.array([
      [0.217,     0.2022,    0.2358,    0.1256,    0.1403],
      [0.8988*0.2497,    0.8988*0.0107,    0.8988*0.2334,    0.8988*0.1282,    0.
 ↪8988*0.378],
      [0.1285,    0.0907,    0.3185,    0.2507,    0.2116],
      [0.1975,    0.0629,    0.2863,    0.2396,    0.2137],
      [0.1256,    0.0711,    0.0253,    0.2244,    0.5536],
    ], ndmin = 2)
    c = np.array([0, 0.1012, 0, 0, 0,], ndmin = 2).T
    A_tilde = np.concatenate((np.concatenate((A, c), axis = 1), # A c
        np.concatenate((np.zeros((1, 5)), np.array([1], ndmin = 2)), axis =␣
 ↪1)), # 0 1
        axis = 0)
    B = np.array([0.0791, 0, 0, 0, 0,], ndmin = 2).T
    B_tilde = np.concatenate((B, np.array([0], ndmin = 2)), axis = 0)
    x = np.array([-0.98, -4.62, 2.74, 4.67, 2.15,], ndmin = 2).T
    w_0 = np.concatenate((x, np.array([z], ndmin = 2)), axis = 0)
    Q = 0.2 * np.identity(5)
    Q_tilde = 0.2 * np.identity(6)
    Q_tilde[5, :] = 0

    def L(K_entry):
        return -1 * np.linalg.inv(B_tilde.T @ K_entry @ B_tilde) @ B_tilde.T @␣
 ↪K_entry @ A_tilde
```

```python
    # first compute the sequence of optimal K_t matrices
    K = np.zeros((6, 6))
    K_t = [Q_tilde, K]
    K = Q_tilde
    current_difference = np.inf
    while abs(current_difference) != 0:
        K_new = delta * (A_tilde.T @ (K
                - (K @ B_tilde @ np.linalg.inv(B_tilde.T @ K @ B_tilde) @ B_tilde.T @ K))
                @ A_tilde) + Q_tilde
        K_t.insert(0, K_new)
        current_difference = np.max(np.abs(K - K_new))
        K = K_new

    # compute the Gamma matrix to use for later computations
    expr = A_tilde + B_tilde @ L(K_t[0])
    A_tilde_n = expr[:5, :5]
    c_nplus1 = np.array(expr[:5, 5], ndmin = 2).T
    x_t = x
    x_ts = [x]

    # compute the resulting sequence of x_t opinion vectors
    for K_ent in K_t:
        x_tp1 = A_tilde_n @ x_t + c_nplus1 * z
        x_ts.append(x_tp1)
        x_t = x_tp1

    # compute the sequence of r_t and cumulative costs
    payoff = 0
    payoffs = []
    r_ts = []
    i = 0
    for x_ent in x_ts:
        r_ts.append(L(K_t[0]) @ np.concatenate((x_ent, np.array([z], ndmin = 2)), axis = 0))
        payoff += (-1 * delta**i * (x_ent.T @ Q @ x_ent)).item() # account for discounting
        payoffs.append(payoff)
        i += 1

    return r_ts, A_tilde, B_tilde, w_0, K_t, x_ts, payoffs
```

[3]:
```python
rs, A_tilde_, B_tilde_, w_0_, Ks, xs, ps = optimal_K(10) # this means z = 10
fig, sub = plt.subplots(2, sharex=True)
fig.suptitle(f"Optimal Strategy: T = infinity (r_t limit = {rs[-1].item()})")
```

```
sub[0].plot(range(len(Ks)+1), [a.item() for a in rs], 'r', label =␣
↪"Optimal-New", linewidth=2)
sub[0].set(ylabel = "r_t message")
sub[1].plot(range(len(Ks)+1), ps, 'r', label = "Optimal-New", linewidth=2)
sub[1].set(xlabel = "Time", ylabel =  "Cumulative Payoff")

sub[0].legend()
sub[1].legend()
plt.show()
```



1.1 The Comparison:

[4]:
```
A_tilde_prime = np.concatenate((np.concatenate((A_tilde_, B_tilde_), axis = 1),␣
↪# A c
                               np.concatenate((np.zeros((1, 6)), np.array([1], ndmin =␣
↪2)), axis = 1)), # 0 1
                               axis = 0)

w_0_prime = A_tilde_prime @ np.concatenate((w_0_, np.array([rs[0].item()],␣
↪ndmin = 2)), axis = 0)
w_0_alt = A_tilde_prime @ np.concatenate((w_0_, np.array([rs[-1].item()], ndmin␣
↪= 2)), axis = 0)
```

```
i = 0
delta = 0.8
Q = 0.2 * np.identity(5)
payoff1 = 0
payoffs1 = []
payoff2 = 0
payoffs2 = []
diffs = []

x_init = np.array([-0.98, -4.62, 2.74, 4.67, 2.15,], ndmin = 2).T

payoff1 += (-1 * delta**i * (x_init.T @ Q @ x_init)).item()
payoffs1.append(payoff1)
payoff2 += (-1 * delta**i * (x_init.T @ Q @ x_init)).item()
payoffs2.append(payoff2)

i = 1

x_t = w_0_prime[:-2]
payoff1 += (-1 * delta**i * (x_t.T @ Q @ x_t)).item()
payoffs1.append(payoff1)
x_t_2 = w_0_alt[:-2]
payoff2 += (-1 * delta**i * (x_t_2.T @ Q @ x_t_2)).item()
payoffs2.append(payoff2)
diff = payoff2 - payoff1
print(f"ACTUAL CUMULATIVE PAYOFF (using r_t, t = 0):")
print(payoff1)
print("ESTIMATED CUMULATIVE PAYOFF (using r_ss):")
print(payoff2)
print("DIFFERENCE:")
print(diff)

i = 2

for r in rs[1:]:
    w_0_prime[6] = r.item()
    w_0_prime = A_tilde_prime @ w_0_prime
    w_0_alt = A_tilde_prime @ w_0_alt

    x_t = w_0_prime[:-2]
    payoff1 += (-1 * delta**i * (x_t.T @ Q @ x_t)).item()
    payoffs1.append(payoff1)
    x_t_2 = w_0_alt[:-2]
    payoff2 += (-1 * delta**i * (x_t_2.T @ Q @ x_t_2)).item()
```

```
    payoffs2.append(payoff2)
    diff = payoff2 - payoff1
    diffs.append(diff)
    if i < 5 or i > 155:
        print()
        print(f"ACTUAL CUMULATIVE PAYOFF (using r_t, t = {i - 1}):")
        print(payoff1)
        print("ESTIMATED CUMULATIVE PAYOFF (using r_ss):")
        print(payoff2)
        print("DIFFERENCE:")
        print(diff)
    i += 1

while np.any(np.round(w_0_prime - w_0_alt, 14)):
    w_0_prime = A_tilde_prime @ w_0_prime # by this point, the strategic agent␣
 ↪has reached r_ss
    w_0_alt = A_tilde_prime @ w_0_alt

    x_t = w_0_prime[:-2]
    payoff1 += (-1 * delta**i * (x_t.T @ Q @ x_t)).item()
    payoffs1.append(payoff1)
    x_t_2 = w_0_alt[:-2]
    payoff2 += (-1 * delta**i * (x_t_2.T @ Q @ x_t_2)).item()
    payoffs2.append(payoff2)
    diff = payoff2 - payoff1
    diffs.append(diff)
    i += 1
    if i % 100 == 0:
        print(diff)

print(len(diffs) - 1, "total iterations needed to match.")
print("ACTUAL PAYOFF:", payoff1, "\n", "ESTIMATED PAYOFF:", payoff2)
```

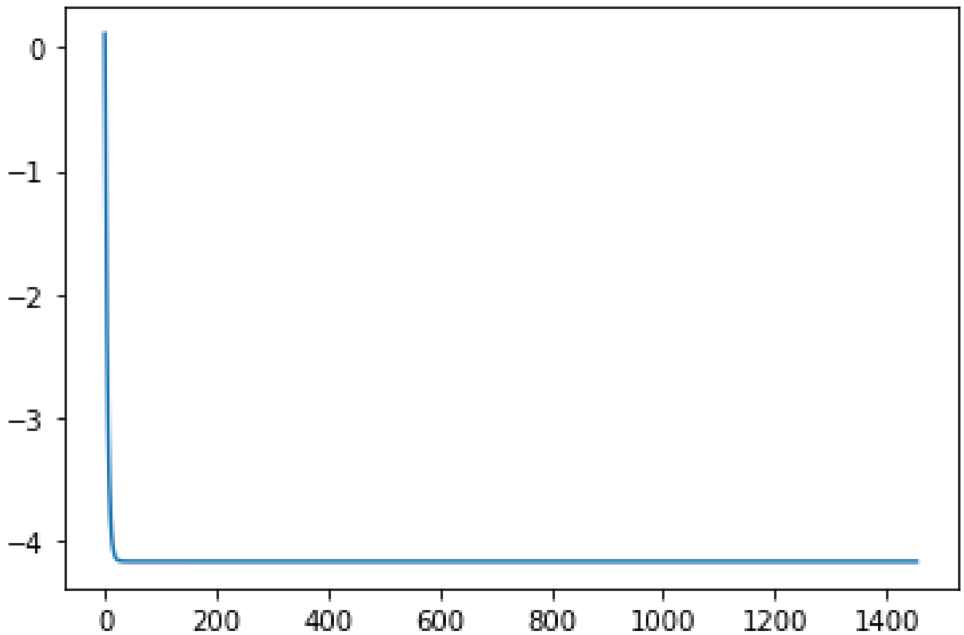
```
ACTUAL CUMULATIVE PAYOFF (using r_t, t = 0):
-14.487555975763826
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-14.053030484089382
DIFFERENCE:
0.4345254916744441

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 1):
-15.952499401192094
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-15.839244804633665
DIFFERENCE:
0.11325459655842884
```

```
ACTUAL CUMULATIVE PAYOFF (using r_t, t = 2):
-16.699866025617425
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-17.248565447762683
DIFFERENCE:
-0.5486994221452584

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 3):
-17.09819136703564
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-18.329907880382397
DIFFERENCE:
-1.2317165133467576

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 155):
-17.780409982367573
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-21.95420748988707
DIFFERENCE:
-4.173797507519499

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 156):
-17.780409982367573
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-21.95420748988707
DIFFERENCE:
-4.173797507519499

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 157):
-17.780409982367573
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-21.95420748988707
DIFFERENCE:
-4.173797507519499

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 158):
-17.780409982367573
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-21.95420748988707
DIFFERENCE:
-4.173797507519499

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 159):
-17.780409982367573
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-21.95420748988707
DIFFERENCE:
-4.173797507519499
```

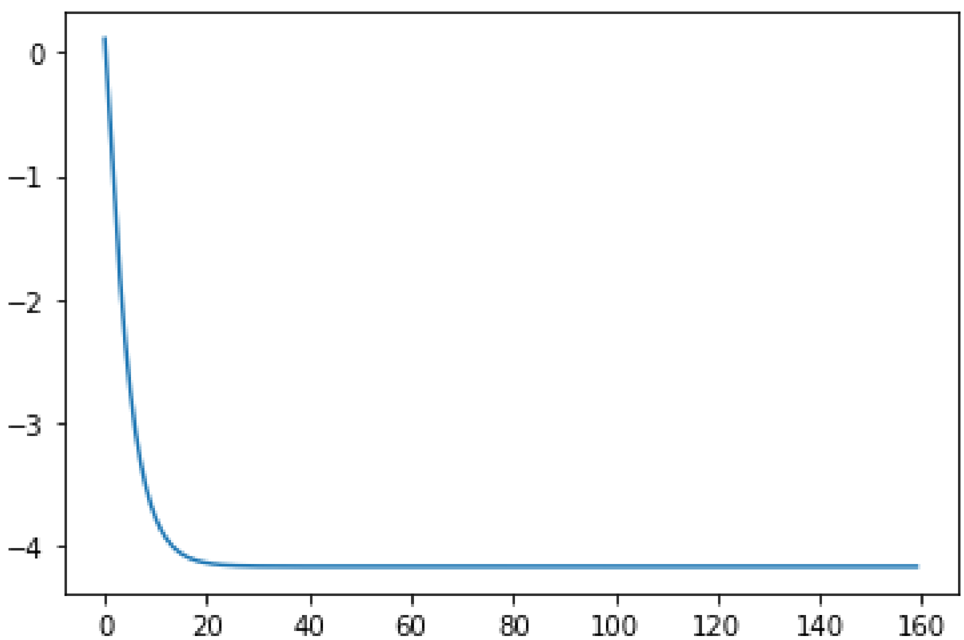
```
ACTUAL CUMULATIVE PAYOFF (using r_t, t = 160):
-17.780409982367573
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-21.95420748988707
DIFFERENCE:
-4.173797507519499

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 161):
-17.780409982367573
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-21.95420748988707
DIFFERENCE:
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
-4.173797507519499
1454 total iterations needed to match.
ACTUAL PAYOFF: -17.780409982367573
 ESTIMATED PAYOFF: -21.95420748988707
```

1.1.1 Difference between payoffs, time on horizontal:

[5]:
```
plt.plot(range(len(diffs)), diffs)
plt.show()
```

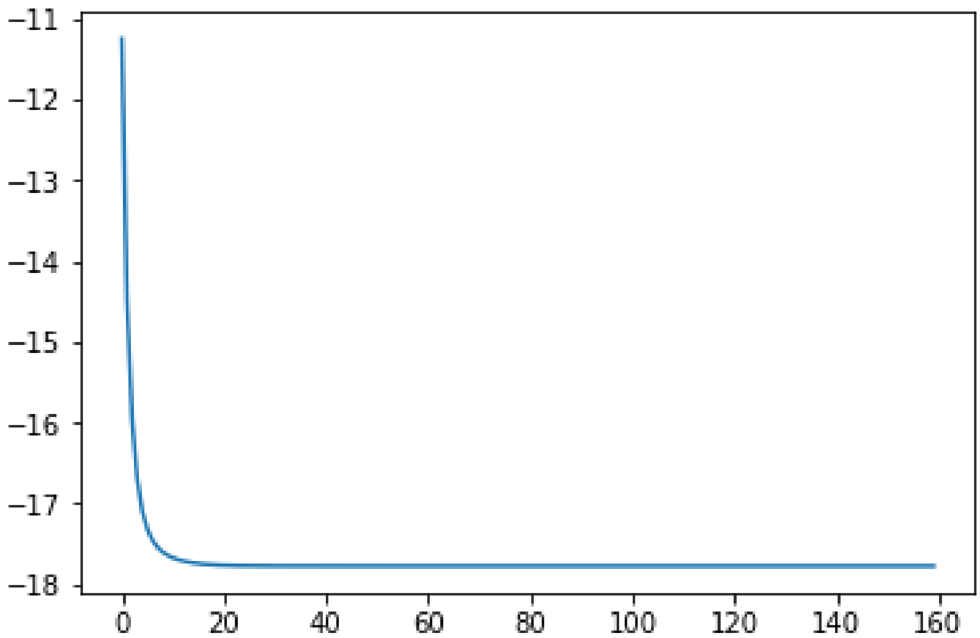


[6]:
```
plt.plot(range(160), diffs[:160])
plt.show()
```



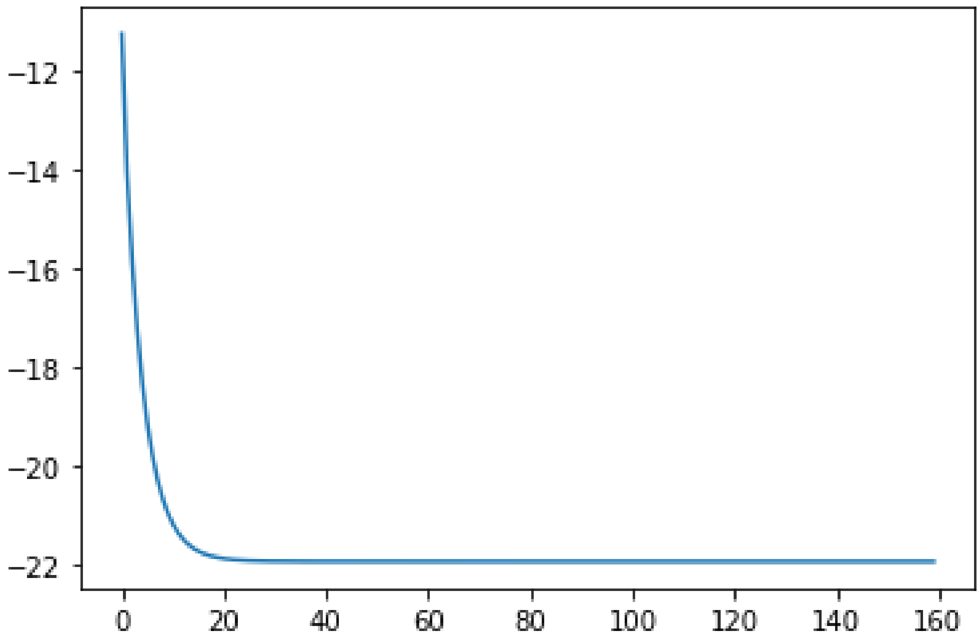
1.1.2 Actual payoffs:

[7]:
```
plt.plot(range(160), payoffs1[:160])
plt.show()
```



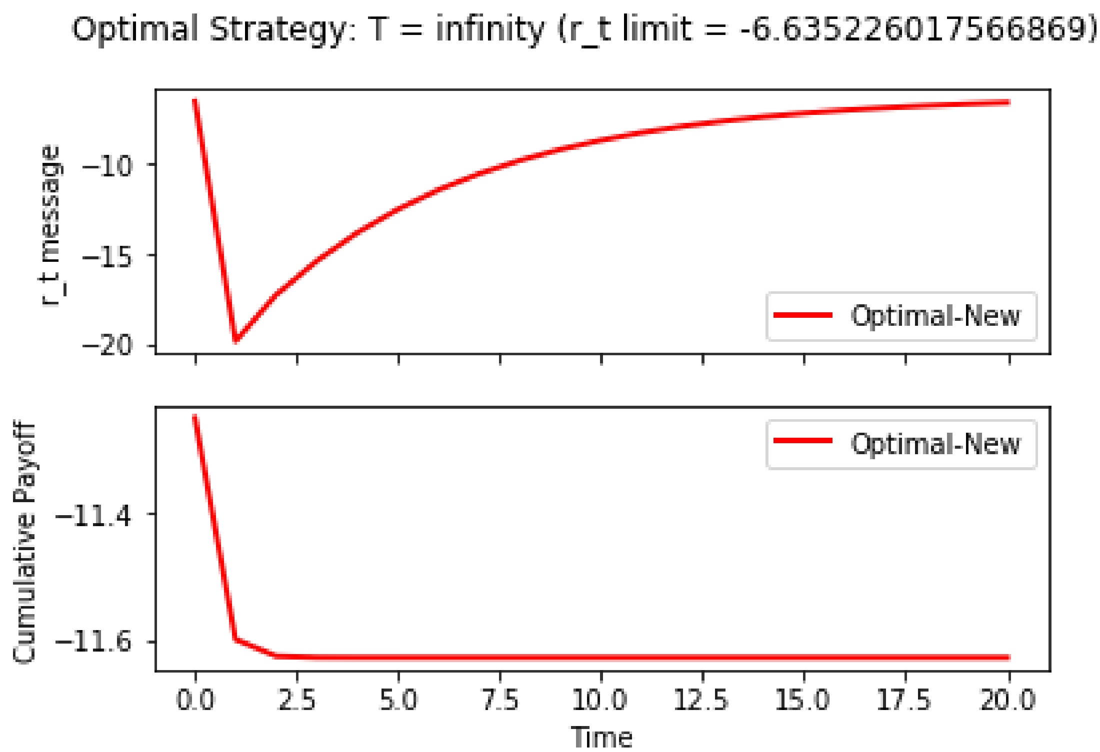
1.1.3 Estimated payoffs:

[8]:
```
plt.plot(range(160), payoffs2[:160])
plt.show()
```



2 Lower Discount: $\delta = 0.1$

[9]:
```
rs, A_tilde_, B_tilde_, w_0_, Ks, xs, ps = optimal_K(10, delta = 0.1)
fig, sub = plt.subplots(2, sharex=True)
fig.suptitle(f"Optimal Strategy: T = infinity (r_t limit = {rs[-1].item()})")
sub[0].plot(range(len(Ks)+1), [a.item() for a in rs], 'r', label =␣
 ↪"Optimal-New", linewidth=2)
sub[0].set(ylabel = "r_t message")
sub[1].plot(range(len(Ks)+1), ps, 'r', label = "Optimal-New", linewidth=2)
sub[1].set(xlabel = "Time", ylabel =  "Cumulative Payoff")

sub[0].legend()
sub[1].legend()
plt.show()
```



[10]:
```
A_tilde_prime = np.concatenate((np.concatenate((A_tilde_, B_tilde_), axis = 1),␣
 ↪# A c
                    np.concatenate((np.zeros((1, 6)), np.array([1], ndmin =␣
 ↪2)), axis = 1)), # 0 1
                    axis = 0)

w_0_prime = A_tilde_prime @ np.concatenate((w_0_, np.array([rs[0].item()],␣
 ↪ndmin = 2)), axis = 0)
w_0_alt = A_tilde_prime @ np.concatenate((w_0_, np.array([rs[-1].item()], ndmin␣
 ↪= 2)), axis = 0)

i = 0
delta = 0.1
Q = 0.2 * np.identity(5)
payoff1 = 0
payoffs1 = []
payoff2 = 0
payoffs2 = []
diffs = []


x_init = np.array([-0.98, -4.62, 2.74, 4.67, 2.15,], ndmin = 2).T
```

```python
payoff1 += (-1 * delta**i * (x_init.T @ Q @ x_init)).item()
payoffs1.append(payoff1)
payoff2 += (-1 * delta**i * (x_init.T @ Q @ x_init)).item()
payoffs2.append(payoff2)

i += 1

x_t = w_0_prime[:-2]
payoff1 += (-1 * delta**i * (x_t.T @ Q @ x_t)).item()
payoffs1.append(payoff1)
x_t_2 = w_0_alt[:-2]
payoff2 += (-1 * delta**i * (x_t_2.T @ Q @ x_t_2)).item()
payoffs2.append(payoff2)
diff = payoff2 - payoff1

print(f"ACTUAL CUMULATIVE PAYOFF (using r_t, t = 0):")
print(payoff1)
print("ESTIMATED CUMULATIVE PAYOFF (using r_ss):")
print(payoff2)
print("DIFFERENCE:")
print(diff)

i = 1

for r in rs[1:]:
    w_0_prime[6] = r.item()
    w_0_prime = A_tilde_prime @ w_0_prime
    w_0_alt = A_tilde_prime @ w_0_alt

    x_t = w_0_prime[:-2]
    payoff1 += (-1 * delta**i * (x_t.T @ Q @ x_t)).item()
    payoffs1.append(payoff1)
    x_t_2 = w_0_alt[:-2]
    payoff2 += (-1 * delta**i * (x_t_2.T @ Q @ x_t_2)).item()
    payoffs2.append(payoff2)
    diff = payoff2 - payoff1
    diffs.append(diff)
    if i < 5 or i > 155:
        print()
        print(f"ACTUAL CUMULATIVE PAYOFF (using r_t, t = {i - 1}):")
        print(payoff1)
        print("ESTIMATED CUMULATIVE PAYOFF (using r_ss):")
        print(payoff2)
        print("DIFFERENCE:")
        print(diff)
    i += 1
```

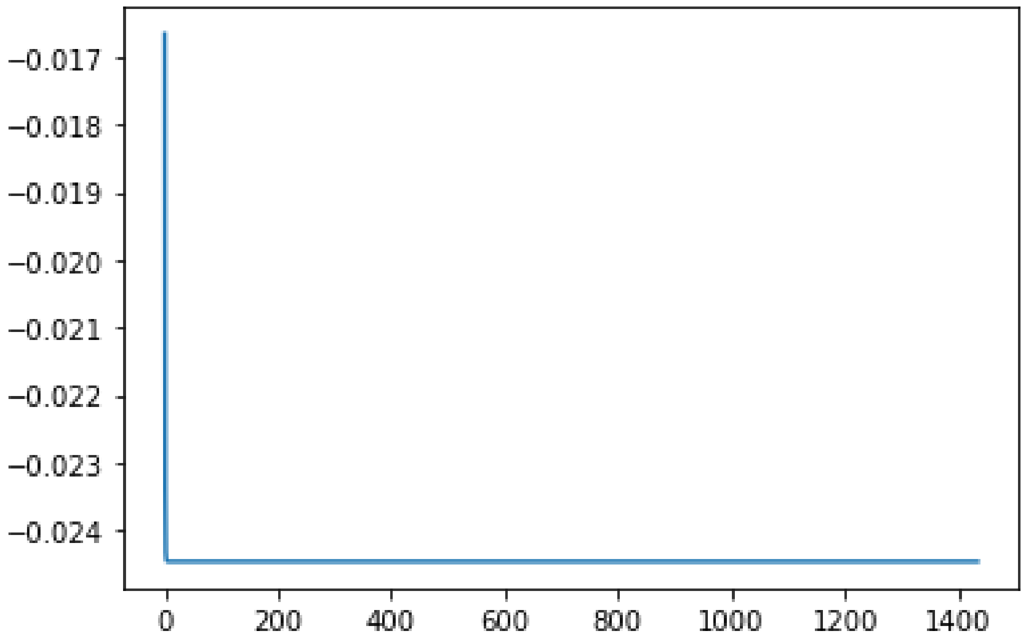
```
while np.any(np.round(w_0_prime - w_0_alt, 14)):
    w_0_prime = A_tilde_prime @ w_0_prime # by this point, the strategic agent␣
 ↪has reached r_ss
    w_0_alt = A_tilde_prime @ w_0_alt

    x_t = w_0_prime[:-2]
    payoff1 += (-1 * delta**i * (x_t.T @ Q @ x_t)).item()
    payoffs1.append(payoff1)
    x_t_2 = w_0_alt[:-2]
    payoff2 += (-1 * delta**i * (x_t_2.T @ Q @ x_t_2)).item()
    payoffs2.append(payoff2)
    diff = payoff2 - payoff1
    diffs.append(diff)
    i += 1
    if i % 100 == 0:
        print(diff)

print(len(diffs) - 1, "total iterations needed to match.")
print("ACTUAL PAYOFF:", payoff1, "\n", "ESTIMATED PAYOFF:", payoff2)
```

```
ACTUAL CUMULATIVE PAYOFF (using r_t, t = 0):
-11.599173199045536
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-11.599200306018101
DIFFERENCE:
-2.710697256524952e-05

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 0):
-11.86315480963553
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-11.87981039357735
DIFFERENCE:
-0.01665558394181943

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 1):
-11.884205153900089
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-11.907571579765584
DIFFERENCE:
-0.02336642586549509

ACTUAL CUMULATIVE PAYOFF (using r_t, t = 2):
-11.885910940365243
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-11.91024258986183
DIFFERENCE:
-0.02433164949658817
```

```
ACTUAL CUMULATIVE PAYOFF (using r_t, t = 3):
-11.886051981651095
ESTIMATED CUMULATIVE PAYOFF (using r_ss):
-11.910500451440585
DIFFERENCE:
-0.024448469789490446
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
-0.02446300389132361
1434 total iterations needed to match.
ACTUAL PAYOFF: -11.886064993276062
 ESTIMATED PAYOFF: -11.910527997167385
```

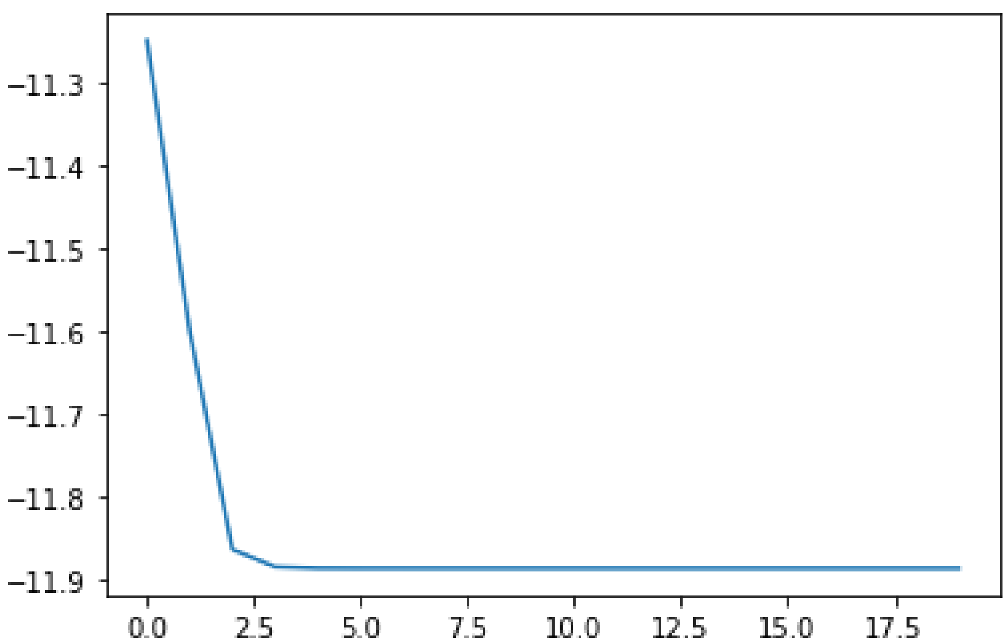
2.0.1 Difference, time on horizontal:

[11]:
```
plt.plot(range(len(diffs)), diffs)
plt.show()
```



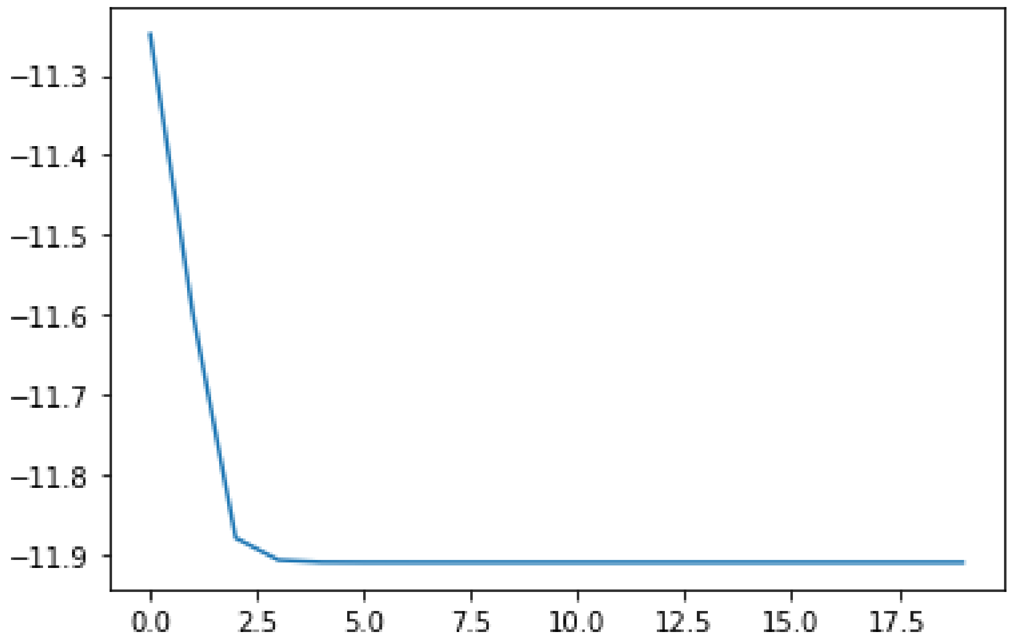
2.0.2 Actual:

[12]:
```
plt.plot(range(20), payoffs1[:20])
plt.show()
```



2.0.3 Estimate:

[13]:
```
plt.plot(range(20), payoffs2[:20])
plt.show()
```



[14]:
```
A_tilde_prime = np.concatenate((np.concatenate((A_tilde_, B_tilde_), axis = 1),␣
 ↪# A c
                    np.concatenate((np.zeros((1, 6)), np.array([1], ndmin =␣
 ↪2)), axis = 1)), # 0 1
                    axis = 0)

w_0_prime = A_tilde_prime @ np.concatenate((w_0_, np.array([rs[0].item()],␣
 ↪ndmin = 2)), axis = 0)
w_0_alt = A_tilde_prime @ np.concatenate((w_0_, np.array([rs[-1].item()], ndmin␣
 ↪= 2)), axis = 0)
print("ACTUAL (using r_t):")
print(w_0_prime)
print("ESTIMATE (using r_ss):")
print(w_0_alt)
diffs = []
i = 0
for r in rs[1:]:
    w_0_prime[6] = r.item()
    w_0_prime = A_tilde_prime @ w_0_prime
    w_0_alt = A_tilde_prime @ w_0_alt
    diff = w_0_prime - w_0_alt
```

```
    diffs.append(diff)
    if i < 5 or i > 15:
        print()
        print(f"ACTUAL (using r_t, t = {i}):")
        print(w_0_prime)
        print("ESTIMATE (using r_ss):")
        print(w_0_alt)
        print("DIFFERENCE:")
        print(diff)
    i += 1

while np.any(np.round(w_0_prime - w_0_alt, 14)):
    w_0_prime = A_tilde_prime @ w_0_prime # by this point, the strategic agent␣
 ↪has reached r_ss
    w_0_alt = A_tilde_prime @ w_0_alt
    diff = w_0_prime - w_0_alt
    diffs.append(diff)
    i += 1
    if i % 100 == 0:
        print("i = ", i, np.round(w_0_prime - w_0_alt, 14))

print(len(diffs), "total iterations needed to match.")
```

```
ACTUAL (using r_t):
[[-0.13235669]
 [ 2.59098488]
 [ 1.953435  ]
 [ 1.878701  ]
 [ 1.85594   ]
 [10.        ]
 [-6.57170283]]
ESTIMATE (using r_ss):
[[-0.13738138]
 [ 2.59098488]
 [ 1.953435  ]
 [ 1.878701  ]
 [ 1.85594   ]
 [10.        ]
 [-6.63522602]]

ACTUAL (using r_t, t = 0):
[[ -0.11822299]
 [  2.26402893]
 [  1.70387079]
 [  1.54285208]
 [  1.66604582]
 [ 10.        ]
```

```
 [-19.85299538]]
ESTIMATE (using r_ss):
[[ 0.92621221]
 [ 2.26290124]
 [ 1.70322511]
 [ 1.5418597 ]
 [ 1.66541472]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-1.04443520e+00]
 [ 1.12769168e-03]
 [ 6.45671920e-04]
 [ 9.92375130e-04]
 [ 6.31100336e-04]
 [ 0.00000000e+00]
 [-1.32177694e+01]]

ACTUAL (using r_t, t = 1):
[[ -0.10636595]
 [  2.10848857]
 [  1.47216693]
 [  1.33257793]
 [  1.45777055]
 [ 10.        ]
 [-17.29202763]]
ESTIMATE (using r_ss):
[[ 0.96263605]
 [ 2.34241648]
 [ 1.60568659]
 [ 1.53822546]
 [ 1.58828303]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[ -1.06900199]
 [ -0.23392791]
 [ -0.13351967]
 [ -0.20564753]
 [ -0.13051248]
 [  0.        ]
 [-10.65680161]]

ACTUAL (using r_t, t = 2):
[[ -0.09658276]
 [  1.96605712]
 [  1.28899859]
 [  1.16390929]
```

```
 [  1.27985206]
 [ 10.        ]
 [-15.40925021]]
ESTIMATE (using r_ss):
[[ 0.95234038]
 [ 2.30427021]
 [ 1.5692809 ]
 [ 1.50514159]
 [ 1.55252805]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-1.04892314]
 [-0.33821309]
 [-0.28028231]
 [-0.3412323 ]
 [-0.27267599]
 [ 0.        ]
 [-8.77402419]]

ACTUAL (using r_t, t = 3):
[[ -0.08847708]
 [  1.84857573]
 [  1.1390653 ]
 [  1.02600725]
 [  1.12997488]
 [ 10.        ]
 [-13.8400947 ]]
ESTIMATE (using r_ss):
[[ 0.92463683]
 [ 2.27799575]
 [ 1.53704294]
 [ 1.47471811]
 [ 1.52038367]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-1.01311391]
 [-0.42942002]
 [-0.39797764]
 [-0.44871087]
 [-0.39040879]
 [ 0.        ]
 [-7.20486868]]

ACTUAL (using r_t, t = 4):
[[ -0.08175607]
 [  1.75100191]
```

```
 [  1.01541151]
 [  0.91222255]
 [  1.00492948]
 [ 10.        ]
 [-12.54528619]]
ESTIMATE (using r_ss):
[[ 0.8973797 ]
 [ 2.25033616]
 [ 1.50640324]
 [ 1.44420555]
 [ 1.48959821]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-0.97913577]
 [-0.49933425]
 [-0.49099173]
 [-0.531983  ]
 [-0.48466873]
 [ 0.        ]
 [-5.91006018]]

ACTUAL (using r_t, t = 16):
[[-0.0525077 ]
 [ 1.32615424]
 [ 0.47895285]
 [ 0.41849264]
 [ 0.45878851]
 [10.        ]
 [-6.92230784]]
ESTIMATE (using r_ss):
[[ 0.61222234]
 [ 1.96587043]
 [ 1.18893755]
 [ 1.12808822]
 [ 1.17090133]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-0.66473004]
 [-0.6397162 ]
 [-0.7099847 ]
 [-0.70959558]
 [-0.71211281]
 [ 0.        ]
 [-0.28708183]]

ACTUAL (using r_t, t = 17):
```

```
[[-0.05191452]
 [ 1.31753725]
 [ 0.46807719]
 [ 0.40848297]
 [ 0.44770717]
 [10.        ]
 [-6.80829985]]
ESTIMATE (using r_ss):
[[ 0.59182168]
 [ 1.94551738]
 [ 1.16622606]
 [ 1.10547153]
 [ 1.1481026 ]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-0.6437362 ]
 [-0.62798013]
 [-0.69814888]
 [-0.69698856]
 [-0.70039543]
 [ 0.        ]
 [-0.17307383]]

ACTUAL (using r_t, t = 18):
[[-0.05142235]
 [ 1.31038779]
 [ 0.45905372]
 [ 0.40017802]
 [ 0.43851306]
 [10.        ]
 [-6.71370815]]
ESTIMATE (using r_ss):
[[ 0.57188467]
 [ 1.92562688]
 [ 1.14403074]
 [ 1.08336885]
 [ 1.12582202]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-0.62330702]
 [-0.61523909]
 [-0.68497702]
 [-0.68319083]
 [-0.68730896]
 [ 0.        ]
 [-0.07848213]]
```

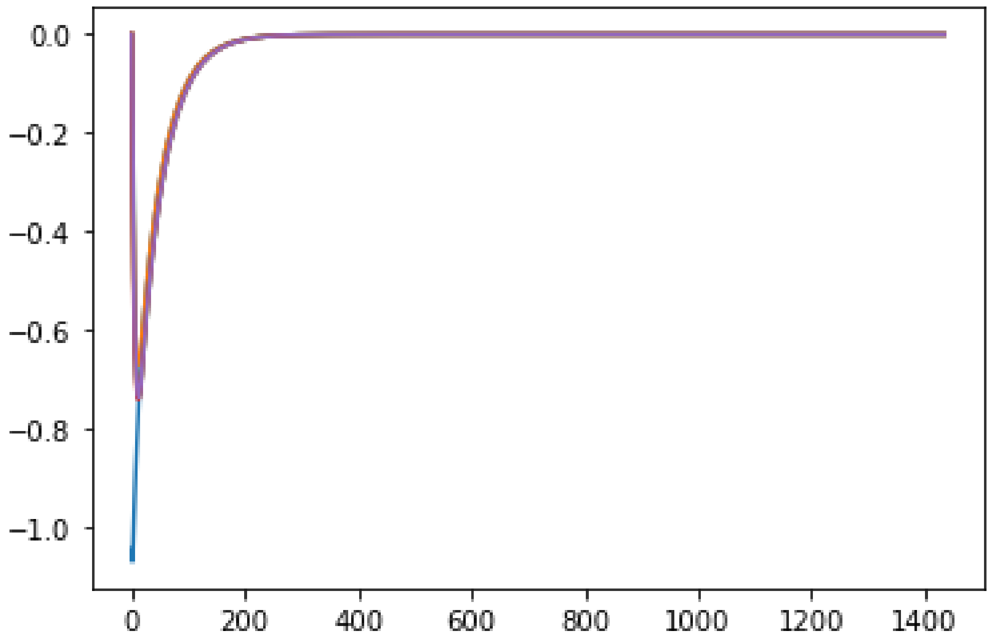
```
ACTUAL (using r_t, t = 19):
[[-0.05101401]
 [ 1.30445593]
 [ 0.451567  ]
 [ 0.39328745]
 [ 0.43088476]
 [10.        ]
 [-6.63522602]]
ESTIMATE (using r_ss):
[[ 0.55240076]
 [ 1.90618843]
 [ 1.12233984]
 [ 1.0617685 ]
 [ 1.1040478 ]
 [10.        ]
 [-6.63522602]]
DIFFERENCE:
[[-0.60341476]
 [-0.6017325 ]
 [-0.67077284]
 [-0.66848105]
 [-0.67316304]
 [ 0.        ]
 [ 0.        ]]
i =  100 [[-0.09581259]
 [-0.09558903]
 [-0.1066655 ]
 [-0.10622026]
 [-0.10707522]
 [ 0.        ]
 [ 0.        ]]
i =  200 [[-0.00961663]
 [-0.0095942 ]
 [-0.01070593]
 [-0.01066125]
 [-0.01074706]
 [ 0.        ]
 [ 0.        ]]
i =  300 [[-0.00096521]
 [-0.00096296]
 [-0.00107455]
 [-0.00107006]
 [-0.00107867]
 [ 0.        ]
 [ 0.        ]]
i =  400 [[-9.68778022e-05]
 [-9.66517611e-05]
```

```
 [-1.07851377e-04]
 [-1.07401186e-04]
 [-1.08265649e-04]
 [ 0.00000000e+00]
 [ 0.00000000e+00]]
i =  500 [[-9.72354998e-06]
 [-9.70086241e-06]
 [-1.08249592e-05]
 [-1.07797738e-05]
 [-1.08665394e-05]
 [ 0.00000000e+00]
 [ 0.00000000e+00]]
i =  600 [[-9.75945180e-07]
 [-9.73668050e-07]
 [-1.08649277e-06]
 [-1.08195755e-06]
 [-1.09066614e-06]
 [ 0.00000000e+00]
 [ 0.00000000e+00]]
i =  700 [[-9.7954860e-08]
 [-9.7726310e-08]
 [-1.0905044e-07]
 [-1.0859524e-07]
 [-1.0946932e-07]
 [ 0.0000000e+00]
 [ 0.0000000e+00]]
i =  800 [[-9.831650e-09]
 [-9.808710e-09]
 [-1.094531e-08]
 [-1.089962e-08]
 [-1.098735e-08]
 [ 0.000000e+00]
 [ 0.000000e+00]]
i =  900 [[-9.86800e-10]
 [-9.84490e-10]
 [-1.09857e-09]
 [-1.09399e-09]
 [-1.10279e-09]
 [ 0.00000e+00]
 [ 0.00000e+00]]
i =  1000 [[-9.9040e-11]
 [-9.8810e-11]
 [-1.1026e-10]
 [-1.0980e-10]
 [-1.1069e-10]
 [ 0.0000e+00]
 [ 0.0000e+00]]
i =  1100 [[-9.940e-12]
```

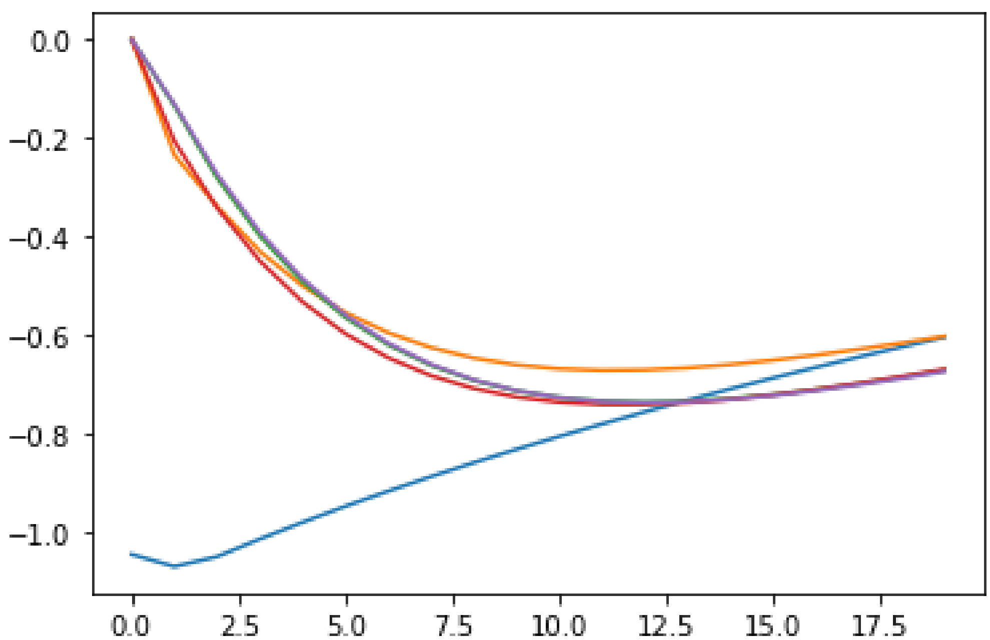
```
 [-9.920e-12]
 [-1.107e-11]
 [-1.102e-11]
 [-1.111e-11]
 [ 0.000e+00]
 [ 0.000e+00]]
i =  1200 [[-1.00e-12]
 [-1.00e-12]
 [-1.11e-12]
 [-1.11e-12]
 [-1.11e-12]
 [ 0.00e+00]
 [ 0.00e+00]]
i =  1300 [[-1.0e-13]
 [-1.0e-13]
 [-1.1e-13]
 [-1.1e-13]
 [-1.1e-13]
 [ 0.0e+00]
 [ 0.0e+00]]
i =  1400 [[-1.e-14]
 [-1.e-14]
 [-1.e-14]
 [-1.e-14]
 [-1.e-14]
 [ 0.e+00]
 [ 0.e+00]]
1435 total iterations needed to match.
```

2.0.4 Difference between opinions, time on horizontal:

[15]:
```
for i in range(5):
    plt.plot(range(len(diffs)), [d[i] for d in diffs])
plt.show()
```



[16]:
```
for i in range(5):
    plt.plot(range(20), [d[i] for d in diffs[:20]])
plt.show()
```



[17]:
```
for i in range(5):
    plt.plot(range(160), [d[i] for d in diffs[:160]])
plt.show()
```