StrategicInfluence4

June 27, 2021

1 Experimentation with Strategic Influence Network Model, Part 4

James Yu

26 June 2021

In this notebook, I break down the behaviour of the limit of the Γ matrix to determine with what proportion the strategic agent influences the naive agents in the long run when a static-agenda (bot) agent is also present.

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

1.1 Standard setup using the augmented model:

[2]:
```
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

z = 10
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
```

[3]:
```
print(A_tilde)
```

```
[[0.217      0.2022     0.2358     0.1256     0.1403     0.        ]
 [0.22443036 0.00961716 0.20977992 0.11522616 0.3397464  0.1012    ]
 [0.1285     0.0907     0.3185     0.2507     0.2116     0.        ]
 [0.1975     0.0629     0.2863     0.2396     0.2137     0.        ]
 [0.1256     0.0711     0.0253     0.2244     0.5536     0.        ]
 [0.         0.         0.         0.         0.         1.        ]]
```

[4]:
```
print(B_tilde)
```

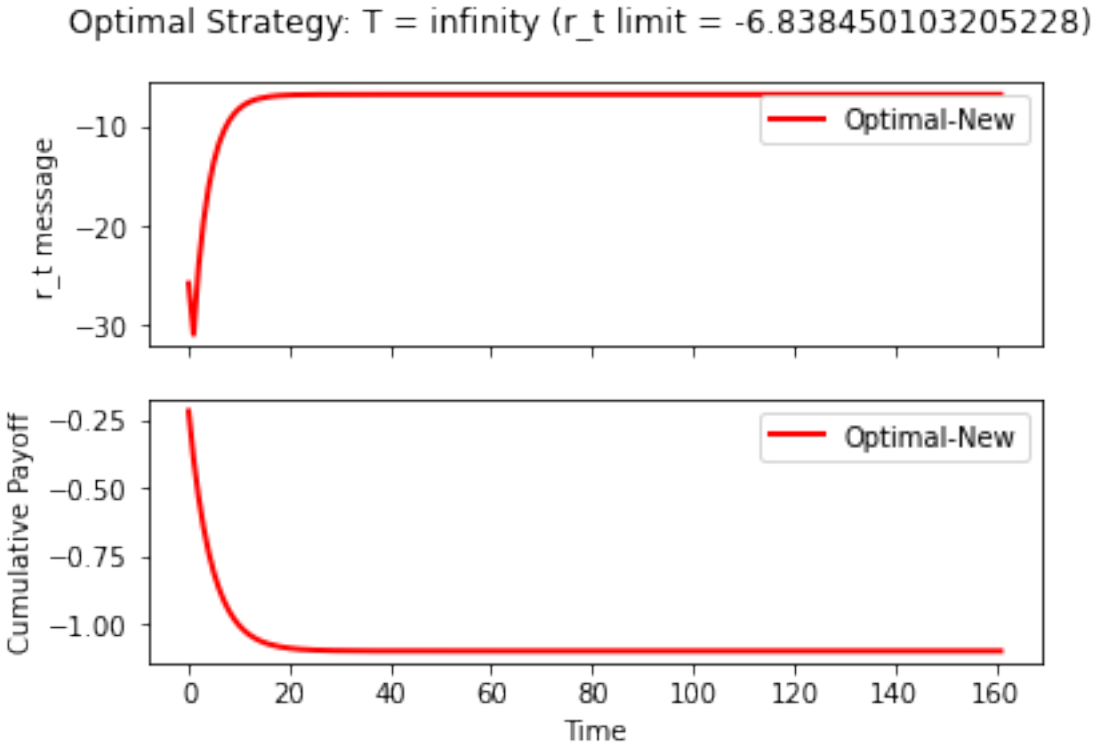
```
[[0.0791]
 [0.    ]
 [0.    ]
 [0.    ]
 [0.    ]
 [0.    ]]
```

1.2 Creation of K matrices in optimal solution for $\delta = 0.8$:

[5]:
```
def L(K_entry):
    return -1 * np.linalg.inv(B_tilde.T @ K_entry @ B_tilde) @ B_tilde.T @␣
 ↪K_entry @ A_tilde

K = np.zeros((6, 6)) # initial K
```

```
K_t = [Q_tilde, K] # saved K
K = Q_tilde
delta = 0.8
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

expr = A_tilde + B_tilde @ L(K_t[0])
print("GAMMA MATRIX:")
print(expr)
A_tilde_n = expr[:5, :5]
c_nplus1 = np.array(expr[:5, 5], ndmin = 2).T
x_t = x
x_ts = [x]
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

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_t.T @ Q @ x_t)).item() # account for␣
 ↪discounting
    payoffs.append(payoff)
    i += 1

fig, sub = plt.subplots(2, sharex=True)
fig.suptitle(f"Optimal Strategy: T = infinity (r_t limit = {r_ts[-1].item()})")
sub[0].plot(range(len(K_t)+1), [a.item() for a in r_ts], 'r', label =␣
 ↪"Optimal-New", linewidth=2)
sub[0].set(ylabel = "r_t message")
sub[1].plot(range(len(K_t)+1), payoffs, 'r', label = "Optimal-New", linewidth=2)
sub[1].set(xlabel = "Time", ylabel =  "Cumulative Payoff")

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

```
GAMMA MATRIX:
[[-0.10354903 -0.04714439 -0.12185954 -0.15429751 -0.24828892 -0.03860925]
 [ 0.22443036  0.00961716  0.20977992  0.11522616  0.3397464   0.1012    ]
 [ 0.1285      0.0907      0.3185      0.2507      0.2116      0.        ]
 [ 0.1975      0.0629      0.2863      0.2396      0.2137      0.        ]
 [ 0.1256      0.0711      0.0253      0.2244      0.5536      0.        ]
 [ 0.          0.          0.          0.          0.          1.        ]]
```



[6]:
```
print(L(K_t[0]))
```

```
[[-4.05245291 -3.15226791 -4.52161234 -3.53852736 -4.91262861 -0.48810687]]
```

As $t \to \infty$, we see:

[7]:
```
Gamma = A_tilde + B_tilde @ L(K_t[0])
print("ORIGINAL GAMMA MATRIX:")
print(Gamma)
print()
print("LIMIT GAMMA MATRIX:")
print(np.linalg.matrix_power(Gamma, 1000000000000))
```

```
ORIGINAL GAMMA MATRIX:
[[-0.10354903 -0.04714439 -0.12185954 -0.15429751 -0.24828892 -0.03860925]
 [ 0.22443036  0.00961716  0.20977992  0.11522616  0.3397464   0.1012    ]
```

```
 [ 0.1285      0.0907      0.3185      0.2507      0.2116      0.        ]
 [ 0.1975      0.0629      0.2863      0.2396      0.2137      0.        ]
 [ 0.1256      0.0711      0.0253      0.2244      0.5536      0.        ]
 [ 0.          0.          0.          0.          0.          1.        ]]

LIMIT GAMMA MATRIX:
[[ 0.          0.          0.          0.          0.         -0.04110785]
 [ 0.          0.          0.          0.          0.          0.09612265]
 [ 0.          0.          0.          0.          0.          0.00697612]
 [ 0.          0.          0.          0.          0.          0.00123909]
 [ 0.          0.          0.          0.          0.          0.00476192]
 [ 0.          0.          0.          0.          0.          1.        ]]
```

This second matrix represents the opinion retention when going from $w_0 = \{-0.98, -4.62, 2.74, 4.67, 2.15, 10\}'$ to w_{ss}.

In other words, given that each row of Γ represents a set of fractions of each agent's opinions through which a single agent is influenced (as in the equation $w_{t+1} = \Gamma w_t$), it is therefore the case that in Γ^∞, each row corresponds to the amount by which each agent influences a particular agent in the "long run".

E.g. in the above limit matrix, we see that the long-run w_{ss} is dependent solely on the initial $w_0[6]$, or the 10 that is the bot's static agenda (because all other columns are zero). Thus, each agent is affected by some specific fraction of this 10 in the long run. Most of them are biased in the positive direction, but the one agent that is influenced by the strategic agent is biased in the negative direction. The bot's entry in the limit matrix is 1, corresponding to the fact that in the long run their opinion never diminishes or changes (that is, 1 * 10 = 10).

What is missing, however, is the fact that Γ implicitly contains a term representing the amount by which each agent is influenced by the strategic agent, which creates information propagation similar to that of the bot. This is seen in the fact that $\Gamma = \tilde{A} + \tilde{B}L_{ss}$.

In a singleton interval, the precise impact of each component of Γ is more clear. The equation is $w_{t+1} = \tilde{A}w_t + \tilde{B}L_{ss}w_t$.

$\tilde{A}$ represents the "raw" amount by which each naive agent (and the bot) manipulates the opinions of the other naive agents over time (as opposed to the "adjusted" amount represented by the combined Γ matrix).

Recall that at optimality, $r_t^* = L_{ss}w_t$. Here for $t = 0$ this is:

[8]:
```
print(L(K_t[0]) @ w_0)
```

```
[[-25.82247922]]
```

Therefore $\tilde{B}$ represents how much the perceived opinion of the strategic agent is imparted onto the naive agents. In this case, as only one entry of $\tilde{B}$ is nonzero, only one agent listens to the strategic agent.

What we see, then, is that the "adjusted" matrix Γ is the composition of the raw agent to agent interactions and the "new" influence from the strategic agent given a specific opinion deployed each period.

[9]:
```
print(B_tilde @ L(K_t[0])) # this is \tilde B L_ss
```

```
[[-0.32054903 -0.24934439 -0.35765954 -0.27989751 -0.38858892 -0.03860925]
 [ 0.          0.          0.          0.          0.          0.        ]
```

```
[ 0.          0.          0.          0.          0.          0.        ]
[ 0.          0.          0.          0.          0.          0.        ]
[ 0.          0.          0.          0.          0.          0.        ]
[ 0.          0.          0.          0.          0.          0.        ]]
```

This matrix, when composited with w_t, provides this "new" influence factor. Observe that there are numbers in each column on the top row, which is merged with $\tilde{A}$ to achieve Γ and is proof of an implicit term.

Here is an alternative approach to looking at Γ. Recall the original time-series transition equation is:

$$w_{t+1} = \tilde{A}w_t + \tilde{B}L_{ss}w_t$$

This can be rewritten as:

$$w_{t+1} = \tilde{A}w_t + \tilde{B}r_t^*$$

Thus, we would see:

$$w_{t+2} = \tilde{A}[\tilde{A}w_t + \tilde{B}r_t^*] + \tilde{B}r_{t+1}^* = \tilde{A}^2w_t + \tilde{A}\tilde{B}r_t^* + \tilde{B}r_{t+1}$$

Notice the polynomial structure of the equation. The next entry in the sequence would be:

$$w_{t+3} = \tilde{A}^3w_t + \tilde{A}^2\tilde{B}r_t^* + \tilde{A}\tilde{B}r_{t+1} + \tilde{B}r_{t+2}$$

This goes on to infinity. Note the fact that there are multiple intermediate terms.

There is strong intuition for this. At time $t = 3$ for example, we have: - $\tilde{A}^3$ representing three periods of the naive agents interacting with themselves - $\tilde{A}^2\tilde{B}$ representing the original r_t message being first sent in period t and then propagated by the naive agents for the next two periods (hence A^2) - $\tilde{A}\tilde{B}$ representing the r_{t+1} message sent in period $t+1$ which had one period of initializing and one period of propagating among the naive agents - $\tilde{B}$, the last term, representing the new r_{t+2} message.

This logic continues until infinity and makes for a very complex matrix polynomial, but there are patterns that can be used to achieve results.

For example, given w_0, as $i \to \infty$ for w_{0+i}, the first term approaches $\tilde{A}^\infty w_0$. This is:

[10]:
```
print(np.linalg.matrix_power(A_tilde, 1000000000000) @ w_0)
```

```
[[ 3.81708031]
 [ 4.63206325]
 [ 4.10264085]
 [ 4.06856981]
 [ 4.08949118]
 [10.        ]]
```

These are the components of the opinions derived from *purely* the bot's (and others') opinion is imposed on the naive agents. We saw before that only the bot matters in the long run, so lets check:

[11]:
```
gamma_builder = np.linalg.matrix_power(A_tilde, 1000000000000)
print(gamma_builder)
```

```
[[0.         0.         0.         0.         0.         0.38170803]
 [0.         0.         0.         0.         0.         0.46320633]
 [0.         0.         0.         0.         0.         0.41026409]
 [0.         0.         0.         0.         0.         0.40685698]
 [0.         0.         0.         0.         0.         0.40894912]
 [0.         0.         0.         0.         0.         1.        ]]
```

Indeed this is the case. Also observe now that the fraction for agent one is biased positively. The limit gamma matrix was:

[12]:
```
print(np.linalg.matrix_power(Gamma, 1000000000000))
```

```
[[ 0.         0.         0.         0.         0.        -0.04110785]
 [ 0.         0.         0.         0.         0.         0.09612265]
 [ 0.         0.         0.         0.         0.         0.00697612]
 [ 0.         0.         0.         0.         0.         0.00123909]
 [ 0.         0.         0.         0.         0.         0.00476192]
 [ 0.         0.         0.         0.         0.         1.        ]]
```

There are unaccounted changes when going from the pure propagation matrix $\tilde{A}^\infty$ to the Γ^∞ matrix, which must come from the strategic agent as they are all negative changes.

In general, the equation that we have is:

$$w_t = \tilde{A}^t w_0 + \tilde{A}^{t-1}\tilde{B}r_0^* + \tilde{A}^{t-2}\tilde{B}r_1 + ... + \tilde{B}r_{t-1}$$

The first term we already saw as $t \to \infty$. The intuition with the middle terms is that, given any particular broadcast that the strategic agent makes in a particular period, the opinions produced by that broadcast will propagate using the agent to agent channels of the $\tilde{A}$ matrix in future periods. Essentially there is a form of "saturation" which in general looks like $\tilde{A}^k\tilde{B}r_i^*$. $\tilde{B}$ is the period in which the broadcast was first imparted on the naive agents, and $\tilde{A}^k$ is k periods of "saturation".

The last term, $\tilde{B}r_{t-1}$, follows the same intuition. r_{t-1} is the last message broadcasted, so there has not been any time for the message to "saturate" among all the naive agents. Thus, there is no A term.

In summary, any particular network has two sets of components. There is the propagation of the initial opinions (described by $\tilde{A}w_0$) and then separate terms for the propagation of each of the strategic agent's broadcasts.

For example, the last message r_{t-1} is:

[13]:
```
r_ts[-1].item()
```

[13]:
```
-6.838450103205228
```

And so the last term is:

[14]:
```
last_term = B_tilde @ r_ts[-1]
print(last_term)
```

```
[[-0.5409214]
 [ 0.       ]
 [ 0.       ]
 [ 0.       ]
```

```
 [ 0.        ]
 [ 0.        ]]
```

Note that so far, we have accounted for the following final opinions:

[15]:
```
gamma_base = gamma_builder @ w_0
print(gamma_base)
```

```
[[ 3.81708031]
 [ 4.63206325]
 [ 4.10264085]
 [ 4.06856981]
 [ 4.08949118]
 [10.        ]]
```

We are looking to achieve the final opinions of the Γ matrix, which are:

[16]:
```
print(Gamma @ w_0)
```

```
[[-1.65509311]
 [ 2.59098488]
 [ 1.953435  ]
 [ 1.878701  ]
 [ 1.85594   ]
 [10.        ]]
```

So there is still some ways to go. We can, however, add the change in opinions described by the $\tilde{B}r^*_{t-1}$ term representing the most recent broadcast:

[17]:
```
gamma_base = gamma_base + last_term
print(gamma_base)
```

```
[[ 3.27615891]
 [ 4.63206325]
 [ 4.10264085]
 [ 4.06856981]
 [ 4.08949118]
 [10.        ]]
```

(Note that the only change is a small decrease in the first entry - this is Agent 1 picking up the strategic agent's final message).

Recall that we are looking for the behaviour of the limit matrices themselves, not just obtain the final numbers. So far we have the $\tilde{A}^\infty$ component:

[18]:
```
print(gamma_builder)
```

```
[[0.         0.         0.         0.         0.         0.38170803]
 [0.         0.         0.         0.         0.         0.46320633]
 [0.         0.         0.         0.         0.         0.41026409]
 [0.         0.         0.         0.         0.         0.40685698]
 [0.         0.         0.         0.         0.         0.40894912]
 [0.         0.         0.         0.         0.         1.        ]]
```

and we have $\tilde{B}$ as before conveying the strategic agent's broadcasts.

Note that so far, the unaccounted part of Γ is:

[19]:
```
print(np.linalg.matrix_power(Gamma, 1000000000000) - gamma_builder)
```

```
[[ 0.          0.          0.          0.          0.         -0.42281588]
 [ 0.          0.          0.          0.          0.         -0.36708368]
 [ 0.          0.          0.          0.          0.         -0.40328796]
 [ 0.          0.          0.          0.          0.         -0.40561789]
 [ 0.          0.          0.          0.          0.         -0.4041872 ]
 [ 0.          0.          0.          0.          0.          0.        ]]
```

which are all strategic agent shifts (as they are in the negative direction). This matrix alone actually explains the fraction of opinions that end up starting from one of the strategic agent broadcasts and not the initial agent opinions. In other words, let the above matrix be Φ. Then we have:

$$\Gamma^{\infty} = \tilde{A}^{\infty} + \Phi$$

This means:

$$w_{ss} = \Gamma^{\infty} w_0 = (\tilde{A}^{\infty} + \Phi) w_0 = \tilde{A}^{\infty} w_0 + \Phi w_0$$

where $\tilde{A}^{\infty}$ represents the long-run influence fractions of the initial opinions of the naive agent, and Φ represents (indirectly) the fractions which appear via strategic broadcasts. The question now, however, is why these are present in the column of the bot.

Note that these are just fractions - really, the $\tilde{A}^{\infty}$ matrix says that some "positive" fraction of the bot's opinion is what is imparted onto the naive agents in the long run, and the Φ matrix says some "negative" fraction of the bot's opinion is also imparted onto the naive agents in the long run.

This would make more sense if we knew how Φ arises (because r_t^* is dependent on w_t, there are still implicit terms). We can try looking at the other terms in the original equation for more info.

[20]:
```
print(A_tilde @ B_tilde @ r_ts[-2])
```

```
[[-0.11737994]
 [-0.12139919]
 [-0.0695084 ]
 [-0.10683198]
 [-0.06793973]
 [ 0.        ]]
```

This for example is what we get from one period of saturation with the second-to-last message (note the -2 for second-to-last, and the single A_tilde). These are raw opinions, and all of them are in the negative direction (but diminished compared to Agent 1's original opinion modification).

[21]:
```
print(A_tilde @ A_tilde @ B_tilde @ r_ts[-3])
```

```
[[-0.08935848]
 [-0.07748472]
 [-0.08939148]
 [-0.09083446]
 [-0.08671749]
 [ 0.        ]]
```

There is a rough trend towards zero. If we keep going:

[22]:
```
print(np.linalg.matrix_power(A_tilde, 1000000000000) @ B_tilde @ r_ts[0])
```

```
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]]
```

Towards infinity, the effect of the original message is zero. That is, the original broadcast is essentially lost.

An analog of this is like people speaking. If you are close by to the person talking, you can hear them very well. However, if you are far away, you cannot hear them at all. The information "diffuses", or turns into background noise.

The network analog/intuition of this is how if you only inform people about something once, they will eventually forget it (unless, for example, $\tilde{A}$ has extremely high or perfect self-retention i.e. perfect memory). Whereas if you inform someone twice, people will forget it less. This was observed recently in an email for the upcoming VSE Workathon where an email asking for volunteers was sent twice and the second round picked up almost as many volunteers than the first.

What we have, therefore, is that in the short-run the broadcasts serve to "keep up" or "ping" the naive agents, whereas in the long-run any one broadcast dissolves into noise.

If we do this operation over all the r_t broadcasts, we should see some tangible results.

[23]:
```
all_results = [np.linalg.matrix_power(A_tilde, i) @ B_tilde @ r_ts[-(i+1)] for␣
↪i in range(len(r_ts))]
```

[24]:
```
[str(list(k.item() for k in a)) for a in all_results]
```

[24]:
```
['[-0.5409214031635335, 0.0, 0.0, 0.0, 0.0, 0.0]',
 '[-0.1173799444864868, -0.121399185243697, -0.06950840030651406,
-0.10683197712479789, -0.0679397282373398, 0.0]',
 '[-0.08935848420069256, -0.07748472181678187, -0.08939147762594955,
-0.09083446443908574, -0.08671749484508037, 0.0]',
 '[-0.07971198550741644, -0.07948093999414374, -0.08810323725653384,
-0.08841033600422034, -0.08738425268708404, 0.0]',
 '[-0.07750763912114388, -0.07701212937323822, -0.0861677715662229,
-0.08582355639573307, -0.08610713380282219, 0.0]',
 '[-0.07556964033971236, -0.07535568111142296, -0.08412540210614916,
-0.08378607326949297, -0.0843182818211232, 0.0]',
 '[-0.07382568623322919, -0.07363382953969988, -0.08218631663327916,
-0.08184393891253673, -0.08245790408482363, 0.0]',
```

'[-0.07213691037818065, -0.07196325145162098, -0.08030789885764181,
-0.07997314522758833, -0.08060166087531981, 0.0]',
'[-0.07049432160760737, -0.07032790201062557, -0.07847930462619121,
-0.07815182028452648, -0.07877582621245593, 0.0]',
'[-0.06889110665159927, -0.06872974041969326, -0.07669454573426966,
-0.07637444867022374, -0.07698749289697696, 0.0]',
'[-0.06732507354682578, -0.06716777444472338, -0.07495113525578698,
-0.07463828781328055, -0.07523848189572271, 0.0]',
'[-0.06579487060501678, -0.06564128354392366, -0.07324760719579604,
-0.07294186240298096, -0.0735287770852448, 0.0]',
'[-0.06429952557351297, -0.06414947441926842, -0.07158288231770471,
-0.07128408351443107, -0.07185778038763718, 0.0]',
'[-0.06283819190534119, -0.06269156616885377, -0.0699560204513049,
-0.0696640115281954, -0.0702247105291154, 0.0]',
'[-0.06141007868039519, -0.06126679035925312, -0.0683661416631714,
-0.06808076887076198, -0.06862873851117969, 0.0]',
'[-0.06001442487174877, -0.059874394732789674, -0.06681239894060079,
-0.06653351165241471, -0.06706903223526683, 0.0]',
'[-0.05865049076848445, -0.058513643630414114, -0.06529396885310794,
-0.06502141973815194, -0.06554477118283576, 0.0]',
'[-0.05731755481045746, -0.05718381795517945, -0.06381004813138642,
-0.06354369316480674, -0.06405515103068689, 0.0]',
'[-0.05601491228489256, -0.05588421489904603, -0.062359852246035526,
-0.06209955066200756, -0.06259938491530169, 0.0]',
'[-0.05474187464478894, -0.05461414760870716, -0.06094261465935762,
-0.060688228886473225, -0.06117670358179489, 0.0]',
'[-0.05349776904174317, -0.05337294483482359, -0.05955758630871717,
-0.059308981900536135, -0.059786355166252546, 0.0]',
'[-0.052281937935787684, -0.052159950581382716, -0.05820403517335236,
-0.057961080738437046, -0.058427604861527156, 0.0]',
'[-0.05109373873631795, -0.050974523760369206, -0.05688124587501817,
-0.056643813007855696, -0.05709973455000376, 0.0]',
'[-0.04993254345860914, -0.04981603785339936, -0.055588519295725856,
-0.05535648250898575, -0.05580204243112664, 0.0]',
'[-0.04879773839062337, -0.04868388058074944, -0.05432517220685241,
-0.05409840886510519, -0.05453384265284797, 0.0]',
'[-0.047688723768220385, -0.04757745357780668, -0.05309053690757607,
-0.052868927162491926, -0.05329446494993319, 0.0]',
'[-0.04660491345802738, -0.046496172078840575, -0.051883960871828956,
-0.051667387598840375, -0.052083254289978405, 0.0]',
'[-0.045545734647609096, -0.0454394646079498, -0.05070480640337359,
-0.05049315513977464, -0.0508995705273029, 0.0]',
'[-0.04451062754271041, -0.04440677267702934, -0.0495524502987521,
-0.049345609183203326, -0.0497427880646506, 0.0]',
'[-0.04349904507139005, -0.0433975504906005, -0.04842628351790653,
-0.048224143231314534, -0.04861229552256104, 0.0]',
'[-0.04251045259488184, -0.04241126465734883, -0.04732571086228878,

```
-0.047128164570029205, -0.04750749541624808, 0.0]',
 '[-0.04154432762502826, -0.041447393908217715, -0.04625015066028726,
-0.046057093955740896, -0.046427803839820064, 0.0]',
 '[-0.0406001595481363, -0.04050542882090868, -0.04519903445980314,
-0.045010365309175286, -0.045372650157675734, 0.0]',
 '[-0.03967744935510922, -0.03958487155064295, -0.04417180672781366,
-0.04398742541620742, -0.04434147670291376, 0.0]',
 '[-0.03877570937771162, -0.03868523556704193, -0.04316792455676325,
-0.04298773363547854, -0.04333373848259612, 0.0]',
 '[-0.03789446303082841, -0.03780604539698749, -0.04218685737762755,
-0.04201076161265775, -0.042348902889709966, 0.0]',
 '[-0.03703324456058133, -0.03694683637332624, -0.041228086679498524,
-0.041055993001197714, -0.04138644942167567, 0.0]',
 '[-0.036191598798169994, -0.036107154389284954, -0.040291105735542516,
-0.04012292318943653, -0.04044586940525223, 0.0]',
 '[-0.03536908091930735, -0.035286555658467285, -0.03937541933518639,
-0.03921105903390182, -0.039526665727694726, 0.0]',
 '[-0.034565256209122375, -0.034484606480305005, -0.03848054352239024,
-0.0383199185986759, -0.03862835257402162, 0.0]',
 '[-0.03377969983240577, -0.03370088301083982, -0.03760600533986821,
-0.03744903090068435, -0.03775045517025316, 0.0]',
 '[-0.03301199660907727, -0.032934971038714525, -0.03675134257912248,
-0.03659793566077331, -0.036892509532485254, 0.0]',
 '[-0.03226174079475574, -0.03218646576625511, -0.0359161035361581,
-0.035766183060444066, -0.03605406222166588, 0.0]',
 '[-0.031528535866316355, -0.031454971595528285, -0.03509984677274964,
-0.03495333350411616, -0.03523467010394744, 0.0]',
 '[-0.030811994312321275, -0.030740101919261124, -0.03430214088313347,
-0.034158957386793656, -0.03443390011646944, 0.0]',
 '[-0.03011173742821315, -0.030041478916512484, -0.03352256426600239,
-0.03338263486701138, -0.03365132903850429, 0.0]',
 '[-0.029427395116163234, -0.029358733352988057, -0.032760704901682106,
-0.03262395564494152, -0.0328865432677508, 0.0]',
 '[-0.02875860568946836, -0.02869150438589379, -0.03201616013437168,
-0.03188251874554313, -0.032139138601751854, 0.0]',
 '[-0.028105015681393227, -0.028039439373224152, -0.03128853645933301,
-0.03115793230663986, -0.031408720024263606, 0.0]',
 '[-0.027466279658357157, -0.027402193687384842, -0.030577449314916846,
-0.030449813371814082, -0.0306949014964827, 0.0]',
 '[-0.026842060037366527, -0.026779430533051096, -0.029882522879315292,
-0.029757787688007794, -0.02997305753018063, 0.0]',
 '[-0.026232026907596274, -0.026170820769165597, -0.02920338987193349,
-0.029081489507723456, -0.02931556410249965, 0.0]',
 '[-0.02563585785602643, -0.02557604273498175, -0.028539691359275532,
-0.028420561395720127, -0.02864931623271871, 0.0]',
 '[-0.025053237797041186, -0.024994782080060413, -0.027891076565241913,
-0.027774654040102792, -0.027998210020196562, 0.0]',
```

```
 '[-0.02448385880590081, -0.024426731598130327, -0.02725720268573832,
-0.027143426067704965, -0.02736190134408131, 0.0]',
 '[-0.023927419955998043, -0.02387159106472423, -0.026637734707497758,
-0.02652654386366716, -0.02674005390427398, 0.0]',
 '[-0.02338362715981324, -0.02332906707850513, -0.026032345231020243,
-0.025923681395115614, -0.026132339043688165, 0.0]',
 '[-0.02285219301348405, -0.022798872906198597, -0.025440714297536592,
-0.025334520038848375, -0.025538435574549108, 0.0]',
 '[-0.022332836644907504, -0.022280728331049295, -0.024862529219904627,
-0.024758748412937362, -0.024958029608640384, 0.0]',
 '[-0.021825283565294302, -0.021774359504721633, -0.024297484417348685,
-0.02419606221215774, -0.02439081439140861, 0.0]',
 '[-0.02132926552409673, -0.021279498802566164, -0.023745281253954902,
-0.023646164047157324, -0.02383649013983837, 0.0]',
 '[-0.02084452036723363, -0.020795884682175362, -0.02320562788083698,
-0.023108763287281175, -0.02329476388401171, 0.0]',
 '[-0.020370791898537497, -0.02032326154515396, -0.022678239081889037,
-0.022583575906968332, -0.02276534931226853, 0.0]',
 '[-0.01990782974435033, -0.01986137960203082, -0.022162836123043993,
-0.02207032433563938, -0.022247966619885925, 0.0]',
 '[-0.019455389221196894, -0.01940999474024085, -0.021659146604957793,
-0.021568737310995643, -0.02174234236119655, 0.0]',
 '[-0.01901323120646529, -0.018968868395107377, -0.021166904319041717,
-0.021078549735652416, -0.02124820930506795, 0.0]',
 '[-0.01858112201202654, -0.01853776742375658, -0.02068584910676654,
-0.0205995025370304, -0.02076530629366625, 0.0]',
 '[-0.018158833260726412, -0.018116463981897493, -0.020215726722164283,
-0.02013134253043144, -0.020293378104429806, 0.0]',
 '[-0.017746141765684165, -0.0177047354034024, -0.01975628869745489,
-0.019673822285226048, -0.019832175315179692, 0.0]',
 '[-0.01734282941233449, -0.017302364082623974, -0.019307292211726716,
-0.019226699994082115, -0.019381454172295823, 0.0]',
 '[-0.01694868304315025, -0.016909137359387007, -0.01886849996260161,
-0.0187897393451656, -0.018940976461888998, 0.0]',
 '[-0.016563494344985087, -0.016524847406593855, -0.018439680040816524,
-0.018362709397245717, -0.01851050938390086, 0.0]',
 '[-0.016187059738976436, -0.016149291120384383, -0.018020605807655576,
-0.017945384457638606, -0.018089825429065093, 0.0]',
 '[-0.01581918027295068, -0.015782270012792105, -0.017611055775167698,
-0.017537543962924992, -0.01767870225866501, 0.0]',
 '[-0.015459661516273641, -0.015423590106840064, -0.01721081348910661,
-0.01713897236237874, -0.017276922587023825, 0.0]',
 '[-0.01510831345709082, -0.015073061834020816, -0.016819667414531194,
-0.016749459004044794, -0.016884274066665678, 0.0]',
 '[-0.014764950401903098, -0.014730499934106446, -0.016437410824005905,
-0.01636879802340627, -0.01650054917608656, 0.0]',
 '[-0.014429390877424848, -0.01439572335723569, -0.016063841688342156,
```

```
-0.015996788234581876, -0.016125545110076023, 0.0]',
 '[-0.014101457534672615, -0.014068555168226367, -0.015698762569822854,
-0.015633233023996145, -0.015759063672531593, 0.0]',
 '[-0.013780977055233684, -0.013748822453062682, -0.015341980517853812,
-0.015277940246466448, -0.015400911171709355, 0.0]',
 '[-0.013467780059665082, -0.013436356227507945, -0.014993306966986898,
-0.014930722123651808, -0.015050898317855393, 0.0]',
 '[-0.013161701017974663, -0.0131309913477945, -0.014652557637261074,
-0.01459139514480991, -0.014708840123164047, 0.0]',
 '[-0.012862578162136977, -0.012832566423343768, -0.014319552436808781,
-0.014259779969810036, -0.014374555804010156, 0.0]',
 '[-0.012570253400597926, -0.01254092373147034, -0.013994115366676288,
-0.013935701334350726, -0.014047868685403841, 0.0]',
 '[-0.012284572234723038, -0.012255909134025227, -0.013676074427807898,
-0.013618987957332237, -0.013728606107617412, 0.0]',
 '[-0.012005383677145503, -0.011977371995934393, -0.013365261530145056,
-0.013309472450335088, -0.013416599334935329, 0.0]',
 '[-0.011732540171971008, -0.011705165105589797, -0.013061512403792634,
-0.013006991229157137, -0.013111683466479227, 0.0]',
 '[-0.011465897516797621, -0.011439144597051192, -0.012764666512205775,
-0.012711384427362844, -0.012813697349061341, 0.0]',
 '[-0.011205314786509908, -0.011179169874018078, -0.012474566967352026,
-0.012422495811799494, -0.012522483492020745, 0.0]',
 '[-0.010950654258807688, -0.0109251035355322, -0.012191060446804555,
-0.012140172700036492, -0.01223788798399811, 0.0]',
 '[-0.010701781341430898, -0.010676811303372237, -0.011913997112723602,
-0.011864265879684997, -0.011959760411605956, 0.0]',
 '[-0.010458564501043184, -0.010434161951103296, -0.011643230532684548,
-0.011594629529556495, -0.011687953779952619, 0.0]',
 '[-0.010220875193738162, -0.01019702723474535, -0.011378617602312466,
-0.011331121142620243, -0.01142232443497961, 0.0]',
 '[-0.009988587797133556, -0.009965281825025799, -0.01112001846968439,
-0.011073601450721135, -0.011162731987573529, 0.0]',
 '[-0.009761579544019999, -0.009738803241183088, -0.010867296461462372,
-0.010821934351021084, -0.010909039239415368, 0.0]',
 '[-0.009539730457532905, -0.009517471786289849, -0.010620318010722114,
-0.010575986834128892, -0.010661112110531931, 0.0]',
 '[-0.009322923287817813, -0.009301170484066016, -0.010378952586444251,
-0.010335628913885876, -0.010418819568516285, 0.0]',
 '[-0.009111043450162011, -0.0090897850171548, -0.010143072624638002,
-0.010100733558777004, -0.010182033559386844, 0.0]',
 '[-0.008903978964567976, -0.00888320366837118, -0.009912553461069938,
-0.009871176624940475, -0.009950628940057743, 0.0]',
 '[-0.008701620396747742, -0.008681317254163651, -0.009687273265574673,
-0.009646836790752596, -0.009724483412397195, 0.0]',
 '[-0.00850386080052178, -0.008484019082488116, -0.009467112977929095,
-0.009427595492969714, -0.009503477458855426, 0.0]',
```

'[-0.008310595661611466, -0.008291204881390866, -0.009251956245278104,
-0.00921333686441511, -0.009287494279650041, 0.0]',
'[-0.008121722842821581, -0.008102772751989316, -0.00904168936110779,
-0.009003947673206955, -0.009076419731504798, 0.0]',
'[-0.007937142530618661, -0.00791862311364092, -0.008836201205772599,
-0.008799317263533689, -0.008870142267948328, 0.0]',
'[-0.007756757183123213, -0.007738658652056751, -0.008635383188596496,
-0.008599337497996897, -0.00866855288119289, 0.0]',
'[-0.007580471479550093, -0.007562784268859837, -0.00843912919158633,
-0.008403902701559619, -0.008471545045631565, 0.0]',
'[-0.007408192271152435, -0.00739090703264356, -0.008247335514819105,
-0.008212909607161551, -0.008279014663015699, 0.0]',
'[-0.007239828533752775, -0.007222936131613507, -0.008059900823596124,
-0.008026257303093821, -0.008090860009406104, 0.0]',
'[-0.007075291321981945, -0.007058782827933198, -0.007876726097498498,
-0.007843847182267135, -0.007906981684032869, 0.0]',
'[-0.006914493725396032, -0.006898360413943377, -0.007697714581533313,
-0.007665582893561855, -0.007727282560253869, 0.0]',
'[-0.006757350826706767, -0.006741584170489845, -0.007522771739632695 5,
-0.007491370295521187, -0.007551667738875208, 0.0]',
'[-0.006603779662448018, -0.006588371327681921, -0.007351805210864907,
-0.007321117412745039, -0.007380044504194124 5, 0.0]',
'[-0.006453699186516718, -0.00643864102851805, -0.007184724768845468,
-0.0071547343954706, -0.007212322283254211, 0.0]',
'[-0.006307030237180514, -0.006292314295970817, -0.007021442285007677,
-0.006992133482996143, -0.00704841260897484, 0.0]',
'[-0.006163695508349332, -0.006149314004325358, -0.006861871696620062,
-0.006833228971831993, -0.00688822908804577, 0.0]',
'[-0.006023619526180832, -0.006009564855839425, -0.006705928980741858,
-0.006677937189764719, -0.006731687374782578, 0.0]',
'[-0.005886728632452826, -0.005872993364154614, -0.006553532135711988,
-0.006526176477423385, -0.006578705152544551 4, 0.0]',
'[-0.005752950976619313, -0.005739527846370925, -0.006404601172305261,
-0.006377867179472659, -0.006429202124856901, 0.0]',
'[-0.005622216519110668, -0.005609098426339259, -0.006259058117406401,
-0.006232931648271499, -0.006283100019098906, 0.0]',
'[-0.005494457049296137, -0.005481637052581996, -0.006116827034007169 5,
-0.006091294263786823, -0.006140322606577855, 0.0]',
'[-0.005369606222668912, -0.005357077535391309, -0.00597783406260349,
-0.005952881474817822, -0.006000795744085172, 0.0]',
'[-0.00524759962333531, -0.005235355609172545, -0.005842007490762937,
-0.005817621868273058, -0.005864447443731134, 0.0]',
'[-0.00512837485991585, -0.005116409028121528, -0.00570927785988873,
-0.005685446275488813, -0.005731207980118949, 0.0]',
'[-0.005011871705664795, -0.005000177706017178, -0.005579578121210957,
-0.005556287927569168, -0.005601010046935174, 0.0]',
'[-0.004898032297209708, -0.004886603914497347, -0.005452843857037817,

```
-0.005430082675713721, -0.00547378897905081, 0.0]',
 '[-0.004786801411100891, -0.004775632558963005, -0.005329013588630471,
-0.005306769297807282, -0.005349483061578738, 0.0]',
 '[-0.004678126843738881, -0.004667211557619255, -0.005208029199165794,
-0.005186289919617123, -0.005228033954461139, 0.0]',
 '[-0.004571959928744093, -0.004561292357637829, -0.00508983650970971,
-0.005068590588362083, -0.005109387270655203, 0.0]',
 '[-0.004468256237149774, -0.004457830633716304, -0.004974386058722612,
-0.004953622048964198 6, -0.004993493358632 7675, 0.0]',
 '[-0.004366976520873904, -0.004356787229348636, -0.004861634152400 5515,
-0.004841340790005553, -0.004880308356755978, 0.0]',
 '[-0.004268087980005731, -0.004258129421154799, -0.004751544275510269,
-0.004731710448674245, -0.004769795609531551, 0.0]',
 '[-0.004171565961190068, -0.004161832613302285, -0.004644088982153442,
-0.004624703693636222, -0.004661927565637701, 0.0]',
 '[-0.004077396230021104, -0.004067882604597857, -0.004539252425559868,
-0.004520304744268635, -0.004556688317434603, 0.0]',
 '[-0.003985578007816763, -0.003976278618176304, -0.004437033738844327,
-0.004418512737304805, -0.004454076994707212, 0.0]',
 '[-0.003896128026362 2986, -0.003887037346783 2786, -0.004337451549040478,
-0.004319346222025699, -0.004354112296038182, 0.0]',
 '[-0.003809085938421051, -0.003800198350661912, -0.004240550000472 7885,
-0.004222849158489 19, -0.004256838535316393, 0.0]',
 '[-0.003724521533980 7946, -0.003715831256961795, -0.004146406788403975,
-0.004129098917643561, -0.004162333706242733, 0.0]',
 '[-0.003642544361620 9274, -0.003634045358657 0084, -0.004055143870236921,
-0.004038216947818882, -0.004070720234675218, 0.0]',
 '[-0.003563316553416 1146, -0.003555002409525 9386, -0.003966941743125122,
-0.003950382992740 0634, -0.003982179311081695, 0.0]',
 '[-0.003487069916910 5247, -0.003478933676245 49, -0.003882058471994766,
-0.003865854040122436, -0.003896969991651 1663, 0.0]',
 '[-0.003414128710848172, -0.003406162660979726, -0.003800855045135247,
-0.003784989571423 3465, -0.003815545465127 86185, 0.0]',
 '[-0.003344939991761 5443, -0.003337135377162 0227, -0.003723829158216788,
-0.003708285204839 0042, -0.003738132897351 6838, 0.0]',
 '[-0.003280114045143 7843, -0.003272460686330 8073, -0.003651660225195 9315,
-0.003636417518326121, -0.003665686753549 2066, 0.0]',
 '[-0.003220478249636 8764, -0.003212964036638597, -0.003585269343826 1514,
-0.003570303764805 3566, -0.003599040855687 6097, 0.0]',
 '[-0.003167148834582 4546, -0.003159759052973 7797, -0.003525899181354053,
-0.003511181424399817, -0.003539442644269 0334, 0.0]',
 '[-0.003121626472534 1548, -0.003114342906430787, -0.003475220395018909,
-0.003460714180715683, -0.003488569193756611, 0.0]',
 '[-0.003085923621812 5584, -0.003078723359741 75, -0.003435473398996261,
-0.003421133096024 3103, -0.003448669524064 3745, 0.0]',
 '[-0.003062734164094 7934, -0.003055558800905 75066, -0.003409657217233 5592,
-0.003395424675791115, -0.003422754178802 5696, 0.0]',
```

```
 '[-0.003055659388587127, -0.0030485297408403603, -0.0034017810653778685,
-0.003387581400453664, -0.0034148477735660857, 0.0]',
 '[-0.0030695090553270934, -0.0030623470927072624, -0.0034171995162214992,
-0.003402935491824589, -0.0034303254487965946, 0.0]',
 '[-0.003110702535575477, -0.003103444457856978, -0.0034630590782032305,
-0.003448603627491341, -0.00347636116365311, 0.0]',
 '[-0.003187803462796539, -0.0031803654885705656, -0.003548893407553976,
-0.00353407966843599 8, -0.003562525194449246, 0.0]',
 '[-0.003312232832141503, -0.003304504531848399, -0.0036874171822245316,
-0.0036720252191860376, -0.0037015810579597245, 0.0]',
 '[-0.0034992216037690806, -0.0034910570100594777, -0.0038955866088094347,
-0.003879325707985 6574, -0.00391055009189 6959, 0.0]',
 '[-0.0037690877782348884, -0.0037602935165819613, -0.004196021155248926,
-0.0041785061849218185, -0.004212138648683541, 0.0]',
 '[-0.004148960950541566, -0.00413928034602035, -0.004618923449144773,
-0.0045996432062285165, -0.004636665368361137, 0.0]',
 '[-0.004675161673493448, -0.004664253305886872, -0.005204728157160194,
-0.005183002656772931, -0.00522472023269991, 0.0]',
 '[-0.00539644258865189, -0.005383851285155682, -0.006007710246449577,
-0.005982632949932901, -0.006030786686455865, 0.0]',
 '[-0.006380839572731083, -0.006365951415153012, -0.007103612176410444,
-0.0070739603820362765, -0.007130898126213658, 0.0]',
 '[-0.007697155437933922, -0.007679195973233913, -0.008569030214503347,
-0.008533261493520765, -0.008601945036214982, 0.0]',
 '[-0.00968569720891878, -0.009663097959297125, -0.010782818757007444,
-0.010737809272168383, -0.010824237044497557, 0.0]',
 '[-0.0079001026396536, -0.007881669646370906, -0.00879496571983486,
-0.008758253902160127, -0.008828748390846845, 0.0]']
```

[25]:
```
term = sum(all_results)
term
```

[25]:
```
array([[-4.17363297],
       [-3.61643818],
       [-3.97217756],
       [-3.99573025],
       [-3.98093673],
       [ 0.        ]])
```

The big set of lists is the collection of raw changes to opinions produced by each broadcast from the most recent to the most early. Observe the general trend towards zero corresponding with "forgetfulness".

The `term` variable is the sum of all these opinion changes. Since we are stopping at about 162 entries, this does not cover the effects of going off to infinity but we do not necessarily need this because w_{ss} is achieveable with the 162 entries we have.

Note that with the limit Γ matrix we expect something like:

[26]:
```
print(np.linalg.matrix_power(Gamma, 162) @ w_0)
```

```
[[-0.41107847]
```

```
 [ 0.96122649]
 [ 0.06976123]
 [ 0.01239086]
 [ 0.04761921]
 [10.        ]]
```

Under the new setup, we have $\tilde{A}^{162} w_0$ (because we are truncating) plus the new term. The former of these is:

[27]:
```
print(np.linalg.matrix_power(A_tilde, 162) @ w_0)
```

```
[[ 3.76255451]
 [ 4.57766467]
 [ 4.04193878]
 [ 4.00812112]
 [ 4.02855594]
 [10.        ]]
```

Thus, we have:

[28]:
```
(np.linalg.matrix_power(A_tilde, 162) @ w_0) + term
```

[28]:
```
array([[-0.41107847],
       [ 0.96122649],
       [ 0.06976123],
       [ 0.01239086],
       [ 0.04761921],
       [10.        ]])
```

This is exactly equal to the expected result, which confirms that the equation is correct.

[29]:
```
x_ts[-1]
```

[29]:
```
array([[-0.41107847],
       [ 0.96122649],
       [ 0.06976123],
       [ 0.01239086],
       [ 0.04761921]])
```

Not only is this equal to the expected result, this is equal to the empirically found steady-state w_{ss} from the graph. So indeed this is correct.

1.3 Alternate method: pretending strategic agent is a bot

[30]:
```
print(A_tilde)
```

```
[[0.217      0.2022     0.2358     0.1256     0.1403     0.        ]
 [0.22443036 0.00961716 0.20977992 0.11522616 0.3397464  0.1012    ]
 [0.1285     0.0907     0.3185     0.2507     0.2116     0.        ]
 [0.1975     0.0629     0.2863     0.2396     0.2137     0.        ]
 [0.1256     0.0711     0.0253     0.2244     0.5536     0.        ]
 [0.         0.         0.         0.         0.         1.        ]]
```

[31]:
```
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)
print("\n".join([str(a) for a in A_tilde_prime]))
```

```
[0.217  0.2022 0.2358 0.1256 0.1403 0.     0.0791]
[0.22443036 0.00961716 0.20977992 0.11522616 0.3397464  0.1012
 0.        ]
[0.1285 0.0907 0.3185 0.2507 0.2116 0.     0.    ]
[0.1975 0.0629 0.2863 0.2396 0.2137 0.     0.    ]
[0.1256 0.0711 0.0253 0.2244 0.5536 0.     0.    ]
[0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 1.]
```

So now the strategic agent is like another bot, with their own diagonal for pushing a static agenda and also the 0.0791 influence factor for Agent 1.

[32]:
```
w_0
```

[32]:
```
array([[-0.98],
       [-4.62],
       [ 2.74],
       [ 4.67],
       [ 2.15],
       [10.  ]])
```

[33]:
```
w_0_prime = np.concatenate((w_0, np.array([r_ts[-1].item()], ndmin = 2)), axis␣
 ↪= 0)
w_0_prime
```

[33]:
```
array([[-0.98     ],
       [-4.62     ],
       [ 2.74     ],
       [ 4.67     ],
       [ 2.15     ],
       [10.       ],
       [-6.8384501]])
```

Here we've added the optimal strategy for the strategic agent as the initial opinion of the strategic agent bot. We can now observe the behaviour of the limit matrix in the long-run.

[34]:
```
print("\n".join([str([round(i, 1) for i in a]) for a in np.linalg.
 ↪matrix_power(A_tilde_prime, 1000000000000)]))
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.6]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.5, 0.5]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.6]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.6]
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.4, 0.6]
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

This is a rounded version in order to get it to fit in the screen. A more precise version:

[35]:
```
print("\n".join([str([round(i, 5) for i in a]) for a in np.linalg.
 ↪matrix_power(A_tilde_prime, 1000000000000)]))
```

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.38171, 0.61829]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.46321, 0.53679]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.41026, 0.58974]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.40686, 0.59314]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.40895, 0.59105]
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
```

The bot is the second to last column, and the strategic agent is the last column. Indeed the strategic agents' is higher.

Note originally the long-run $\tilde{A}^{\infty}$ was:

[36]:
```
print(np.linalg.matrix_power(A_tilde, 1000000000000))
```

```
[[0.         0.         0.         0.         0.         0.38170803]
 [0.         0.         0.         0.         0.         0.46320633]
 [0.         0.         0.         0.         0.         0.41026409]
 [0.         0.         0.         0.         0.         0.40685698]
 [0.         0.         0.         0.         0.         0.40894912]
 [0.         0.         0.         0.         0.         1.        ]]
```

These exact numbers form the second to last column, which means this correctly isolates the "non-strategic-agent" opinion effect of the real bot.

[37]:
```
np.linalg.matrix_power(A_tilde_prime, 1000000000000)[:, 5] + np.linalg.
 ↪matrix_power(A_tilde_prime, 1000000000000)[:, 6]
```

[37]:
```
array([1., 1., 1., 1., 1., 1., 1.])
```

Observe that the second to last and the last columns sum up to one. This means there no longer is any hidden information; we correctly have a distribution over the agents for the opinions in the long run, and more weight is given to the strategic agent.

If we then multiply this by w_0', we get:

[38]:
```
print(np.linalg.matrix_power(A_tilde_prime, 1000000000000) @ w_0_prime)
```

```
[[-0.41107847]
 [ 0.96122649]
 [ 0.06976123]
 [ 0.01239086]
 [ 0.04761921]
 [10.        ]
 [-6.8384501 ]]
```

Recall originally we had the optimal long-run opinions of:

[39]:
```
print(x_ts[-1])
```

```
[[-0.41107847]
 [ 0.96122649]
 [ 0.06976123]
 [ 0.01239086]
 [ 0.04761921]]
```

Once again they are exactly the same, which means this worked.