# Analytical-Dual

September 24, 2021

## 1 Analytical Example 2: Two-Message Agent

James Yu, 24 September 2021

[1]:
```
from sympy import *

K, x, r_1, r_2, a, b_1, b_2, R, delta = symbols("K x r_1 r_2 a b_1 b_2 R delta")
```

First, the first-order conditions for each strategic agent:

[2]:
```
rt1 = -(a*b_1*K*x + b_1*b_2*K*r_2) / (b_1*b_1*K + R/delta)
rt1
```

[2]:
$$\frac{-Kab_1x - Kb_1b_2r_2}{Kb_1^2 + \frac{R}{\delta}}$$

[3]:
```
rt2 = -(a*b_2*K*x + b_1*b_2*K*r_1) / (b_2*b_2*K + R/delta)
rt2
```

[3]:
$$\frac{-Kab_2x - Kb_1b_2r_1}{Kb_2^2 + \frac{R}{\delta}}$$

We want to solve for $r_1$ and $r_2$, so we can substitute the equations into each other to get explicit forms:

[4]:
```
also_rt1 = simplify(rt1.subs(r_2, rt2))
Lx1 = solve(r_1 - also_rt1, r_1)[0]
Lx1
```

[4]:
$$-\frac{Kab_1\delta x}{Kb_1^2\delta + Kb_2^2\delta + R}$$

[5]:
```
also_rt2 = simplify(rt2.subs(r_1, rt1))
Lx2 = solve(r_2 - also_rt2, r_2)[0]
Lx2
```

[5]:
$$-\frac{Kab_2\delta x}{Kb_1^2\delta + Kb_2^2\delta + R}$$

This yields the expressions $r_1 = L_1x$ and $r_2 = L_2x$. Notice that $x$ is linearly related to each expression now.

[6]:
```
L_1 = Lx1 / x
L_2 = Lx2 / x
Ksub1 = simplify(1 + R*(L_1*L_1 + L_2*L_2) + delta * K * (a + b_1*L_1 +
↪b_2*L_2)**2)
Ksub1
```

[6]:
$$\frac{KRa^2\delta + Kb_1^2\delta + Kb_2^2\delta + R}{Kb_1^2\delta + Kb_2^2\delta + R}$$

[7]:
```
Ksub1_data = Ksub1.subs(a, 0.7).subs(b_1, 0.29).subs(b_2, 0.01).subs(R, 0.2).
↪subs(delta, 0.8)
Ksub1_data
```

[7]:
$$\frac{0.14576K + 0.2}{0.06736K + 0.2}$$

Now simply iterate this from $K = 1$ upwards to get the steady-state matrix.

[8]:
```
Kunit = 1
while True:
    K_prime = Ksub1_data.subs(K, Kunit)
    if abs(K_prime - Kunit) < 10**(-12):
        break
    print("K =", Kunit)
    Kunit = K_prime
```

```
K = 1
K = 1.29323758228606
K = 1.35313625238492
K = 1.36437190901285
K = 1.36644487239033
K = 1.36682615730550
K = 1.36689624822010
K = 1.36690913156456
K = 1.36691149959455
K = 1.36691193485004
K = 1.36691201485207
K = 1.36691202955682
K = 1.36691203225963
K = 1.36691203275641
K = 1.36691203284773
K = 1.36691203286451
```

From here, we can compute the steady-state message.

[9]:
```
print(L_1.subs(a, 0.7).subs(b_1, 0.29).subs(b_2, 0.01).subs(R, 0.2).subs(delta,
↪0.8).subs(K, Kunit) * 4)
```

```
-3.04012827233619
```

[10]:
```
print(L_2.subs(a, 0.7).subs(b_1, 0.29).subs(b_2, 0.01).subs(R, 0.2).subs(delta,␣
↪0.8).subs(K, Kunit) * 4)
```

```
-0.104832009390903
```

[11]:
```
-0.104832009390903+ 0.01084864
```

[11]:
```
-0.09398336939090299
```

This indicates that the discrepancy in the graph is expected.