opensource_vs_private

April 24, 2024

1 Open Source vs Closed

These are just the libraries used and a bunch of symbols defined

[380]:
```python
from sympy import *
init_printing()
import numpy as np
import matplotlib.pyplot as plt
```

[381]:
```python
l1 = Symbol("\\lambda_1")
l2 = Symbol("\\lambda_2")
l3 = Symbol("\\lambda_3")

phi = Symbol("\\phi")

p1 = Symbol("p_1")
p2 = Symbol("p_2")
p3 = Symbol("p_3")

q1 = Symbol("q_1")
q2 = Symbol("q_2")
q3 = Symbol("q_3")

a1 = Symbol("a_1")
a2 = Symbol("a_2")
a3 = Symbol("a_3")


x1 = Symbol("x_1")
x2 = Symbol("x_2")
x3 = Symbol("x_3")

l = Symbol("\\lambda")
a = Symbol("a")
b = Symbol("b")
c = Symbol("c")
d = Symbol("d")
f = Symbol("f")
```

```
g = Symbol("g")
h = Symbol("h")
w = Symbol("w")
z = Symbol("z")

xa = Symbol("x_A")
temp = Symbol("temp")

gamma = Symbol("\\gamma")
init_printing(True)
```

[382]:
```
f =0.25 # adjustment to get interior solutions (cost of investing)
```

1.1 Model

2 firms downstream (1,2), competing in quantities (symmetric eq); 1 monopolist providing AI called (A).

Downstream

Demand Dowstream

$p_i = 1 - q_i + a_i + bq_j - ba_j$

where a_i is quality of good i, b subs. degree (homogeneous in quality and price)

[383]:
```
p1 = 1-q1+a1+b*q2-b*a2
```

Profits downstream

$\Pi^1 = q_1(p_1 - w) - fx_1^2$;

where - w is the price set by the monopolist, - x_1 the investment of firm i in the development of the AI and - f just the adjustment of the cost.

Quality of good i, a_i is given by:

$a_i = \phi(\lambda x_i + x_j) + (1 - \phi)(\lambda q_i + q_j)$

where:

- ϕ is the degree of open-sourceness of the AI
- $\lambda(> 1)$ is the extent to which the AI learns more with own resources than other's.

Here, I solve the dowsntream stage for given w, ϕ, letting the firms choose x_i and q_i

[384]:
```
Pi1 = q1*(p1-w) - f*x1**2
a1_e  = phi*(l*x1 + x2) + (1-phi)*(l*q1+q2)
a2_e  = phi*(l*x2 + x1) + (1-phi)*(l*q2+q1)
```

[385]:
```
Pi1 = Pi1.subs(a1,a1_e).subs(a2,a2_e)
```

[386]:
```
focq1sym = simplify(diff(Pi1,q1).subs(q2,q1).subs(x2,x1))
```

[387]:
```
focx1sym = diff(Pi1,x1).subs(x2,x1).subs(q2,q1)
```

[388]:
```
sol = solve([focq1sym,focx1sym],[q1,x1])
```

[389]:
```
q1sol = sol[q1]
q2sol = sol[q1]
x1sol = sol[x1]
x2sol = sol[x1]
```

Upstream

Profit of firm A:

$\Pi^A = (1-\phi)w(q_1+q_2) + \gamma\left(\phi(x_1+x_2) + (1-\phi)(q_1+q_2)\right)$

where

- γ is the profits that a complementary (independent) product owned by the monopolist would get from the quality of AI.
- Profits are Π = selling AI + Complementary Product. The complemenary product is there independently of whether AI is open source or not (if AI is a key input of such product, having it open source does not seem a great idea if barriers to entry are low). On the other hand, the AI can be sold only to the extent that AI is private, i.e.: $1-\phi$.

I solve for the optimal w and conduce comparative statics wrt ϕ

[390]:
```
# first stage
```

[391]:
```
PiA = (1-phi)*w*(q1+q2) + gamma*(phi*(x1 + x2) + (1-phi)*(q1+q2))
```

[392]:
```
PiAext = simplify(PiA.subs(q1,q1sol).subs(q2,q1sol).subs(x2,x1sol).
 ↪subs(x1,x1sol))
```

[393]:
```
PiAext
```

[393]:

$$\frac{2\left(-\gamma\left(2.0\phi^2\left(\lambda w-\lambda-bw+b\right)-\left(\phi-1\right)\left(w-1.0\right)\right)+w\left(\phi-1\right)\left(w-1.0\right)\right)}{2.0\lambda^2\phi^2b-2.0\lambda^2\phi^2-2.0\lambda\phi^2b^2+4.0\lambda\phi^2b-2.0\lambda\phi^2-\lambda\phi b+2.0\lambda\phi+\lambda b-2.0\lambda-2.0\phi^2b^2+2.0\phi^2b-2.0\phi b+\phi+b}$$

[394]:
```
focw =   simplify(diff(PiAext, w))
```

[395]:
```
sol_1stage = solve([focw],[w])
```

[396]:
```
sol_1stage
```

[396]:

$$\left\{w:\frac{2.0\gamma\lambda\phi^2-2.0\gamma\phi^2b-\gamma\phi+\gamma+\phi-1.0}{2.0\phi-2.0}\right\}$$

[397]:
```
wsol = sol_1stage[w]
```

Comparative Statics

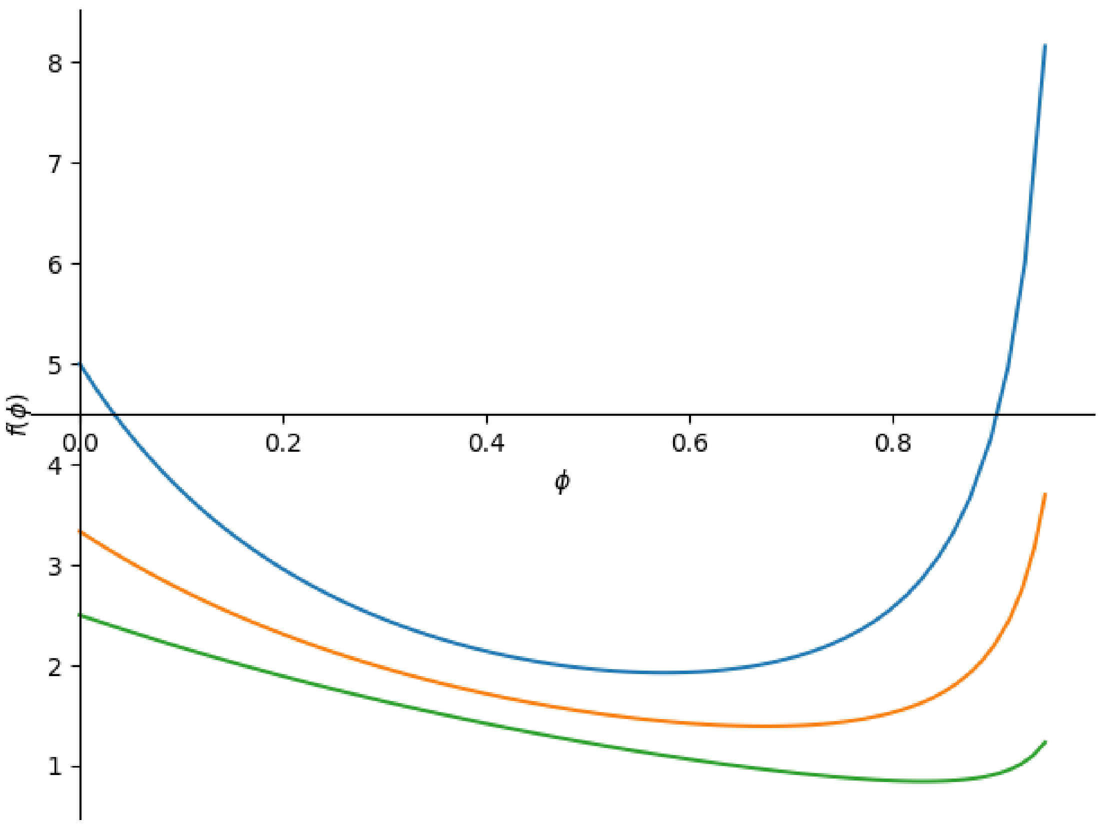
WRT Quality a

- for different b (downstream subst. degree)
- for different γ (complementary good importnace)
- for different λ (premium to own-investment)

[398]:
```
# plot quality in terms of phi
```

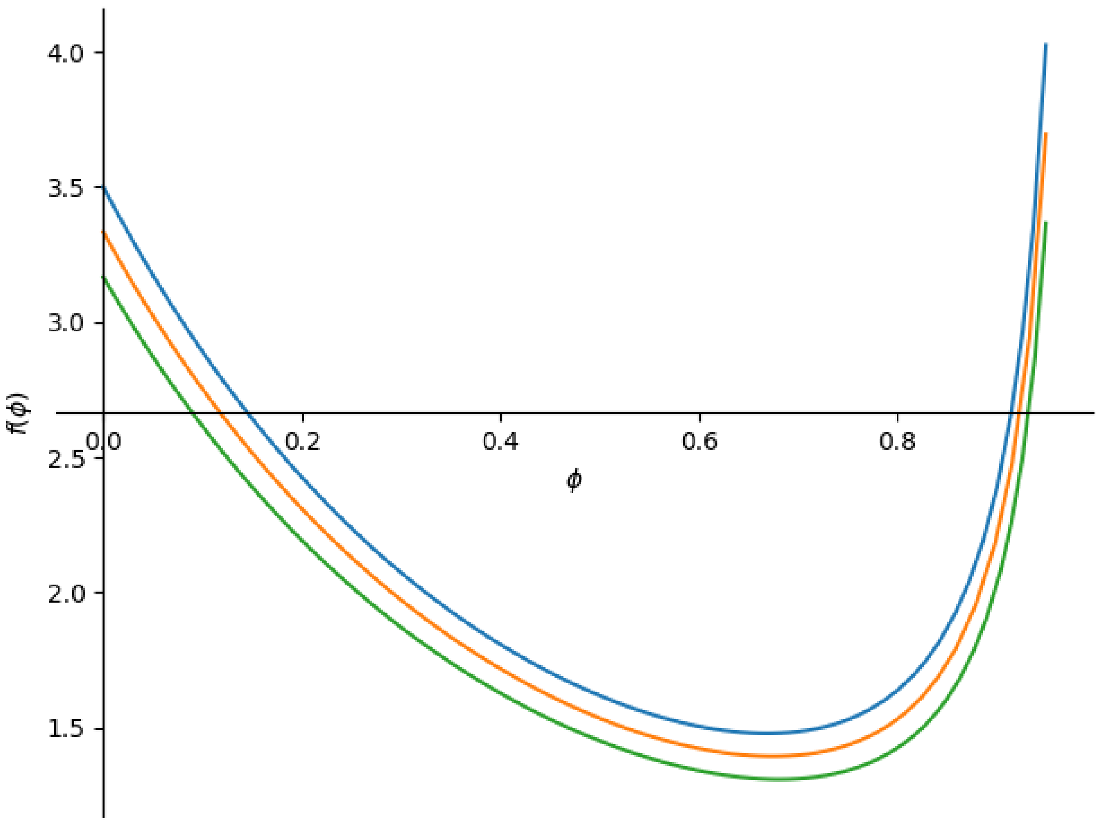
[399]:
```
a = phi*(x1 + x2) + (1-phi)*(q1+q2)
a = simplify(a.subs(x1,x1sol).subs(x2,x1sol).subs(q1,q1sol).subs(q2,q1sol).
 ↪subs(w,wsol))
```

[400]:
```
# for different b - subst. degree
plot(a.subs(b,0.7).subs(l,1).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.9).subs(l,1).subs(gamma,1),(phi,0,0.95))
```



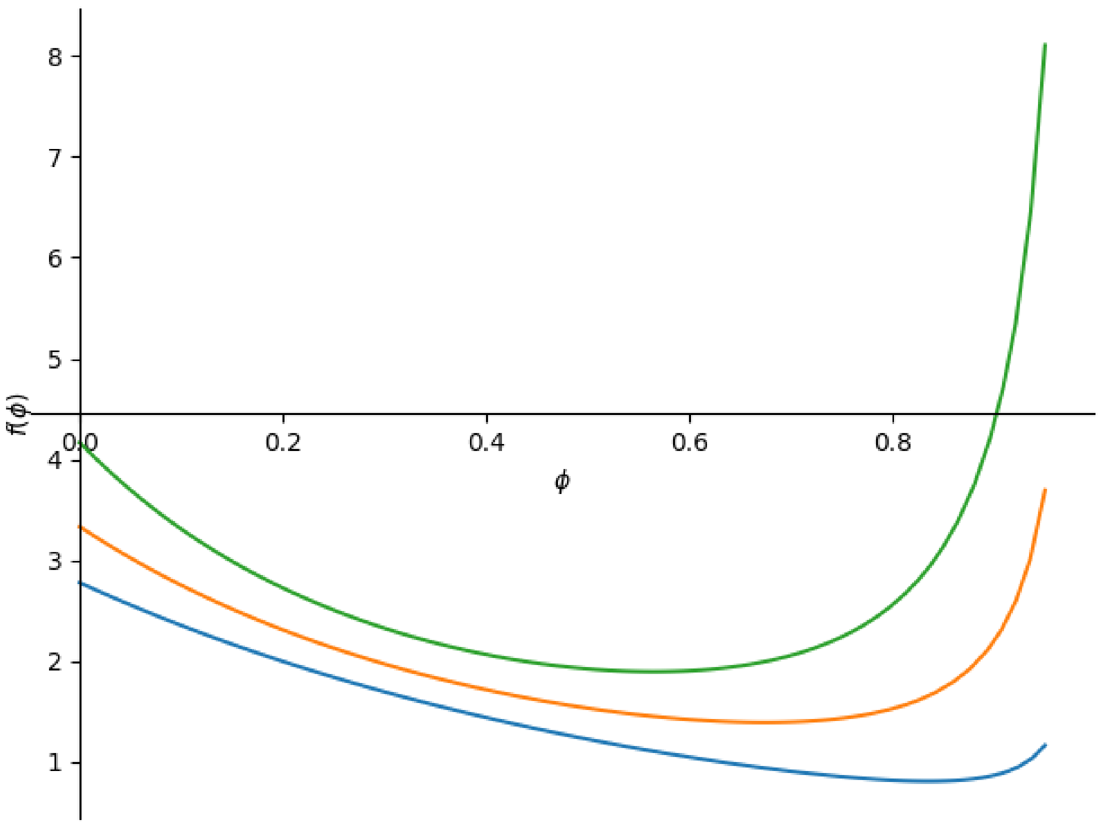
[400]: <sympy.plotting.plot.Plot at 0x7f3ba34fa740>

[401]:
```
# for different  gamma-complementary good importnace
plot(a.subs(b,0.8).subs(l,1).subs(gamma,1.1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1).subs(gamma,0.9),(phi,0,0.95))
```



[401]: <sympy.plotting.plot.Plot at 0x7f3bb49710c0>

[402]:
```
# for different  l-own learning improvement
plot(a.subs(b,0.8).subs(l,0.9).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1.1).subs(gamma,1),(phi,0,0.95))
```



[402]: <sympy.plotting.plot.Plot at 0x7f3ba32ae980>

Comparative Statics

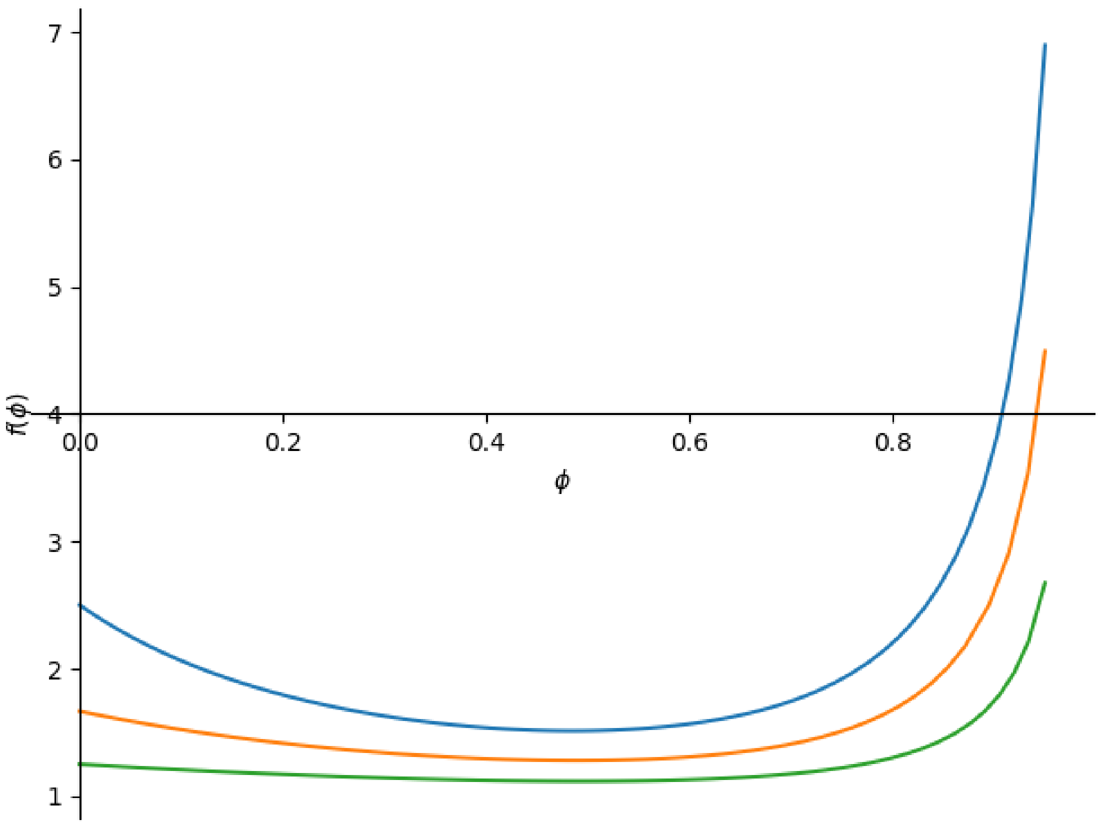
WRT Quantity $q_1 (= q_2)$

- for different b (downstream subst. degree)
- for different γ (complementary good importnace)
- for different λ (premium to own-investment)

[403]:
```
# plot quanitity in terms of phi
```

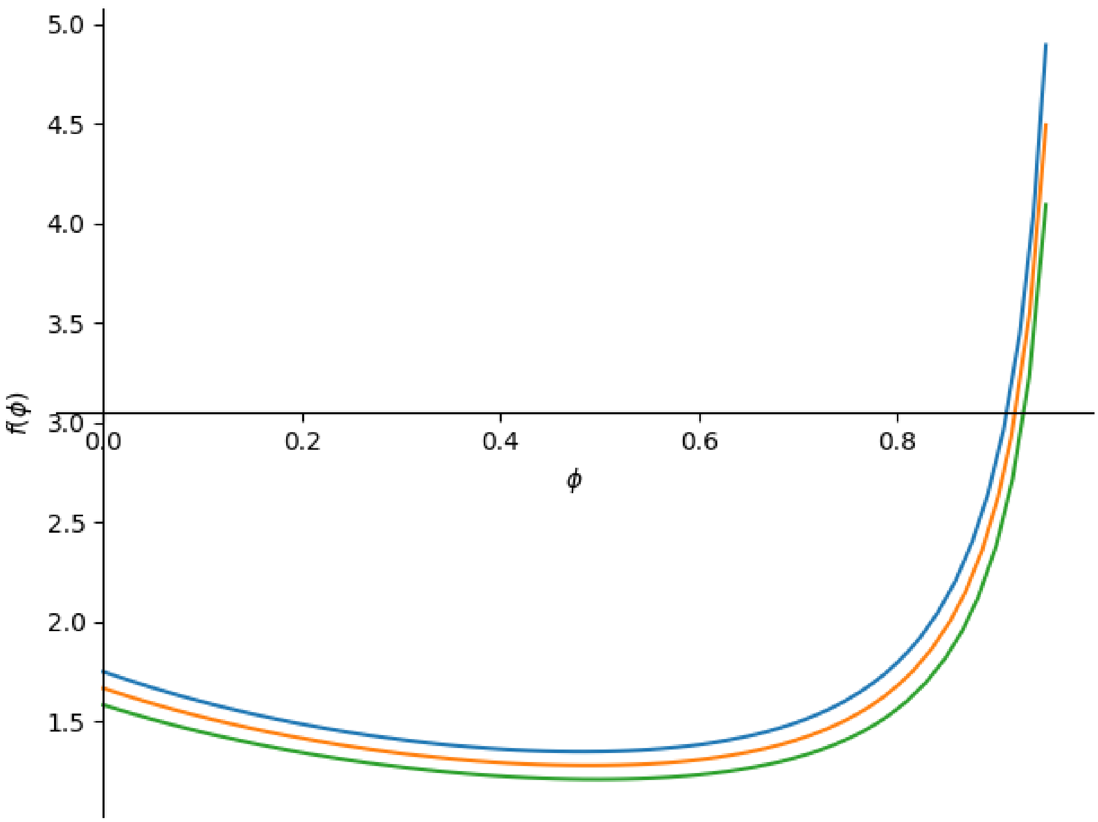
[404]:
```
a = simplify(q1sol.subs(w,wsol))
```

[405]:
```
# for different b - subst. degree
plot(a.subs(b,0.7).subs(l,1).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.9).subs(l,1).subs(gamma,1),(phi,0,0.95))
```



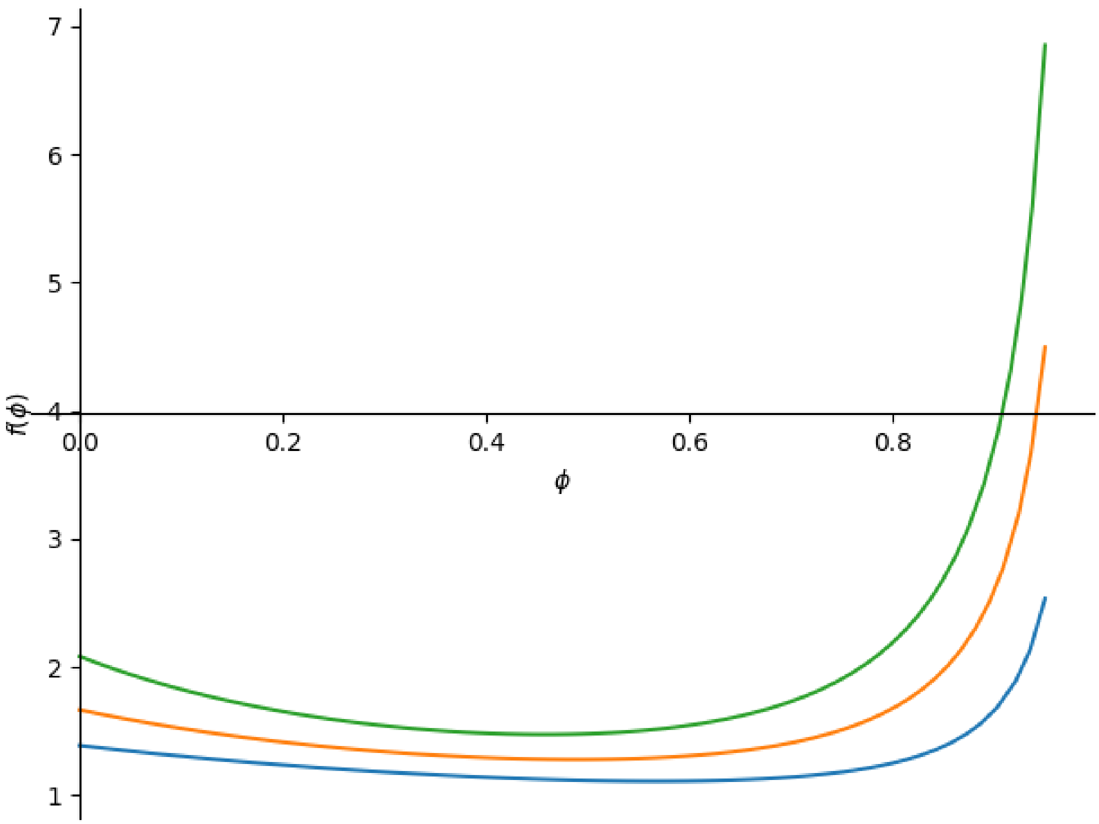
[405]: <sympy.plotting.plot.Plot at 0x7f3ba332c580>

[406]:
```
# for different  gamma-complementary good importnace
plot(a.subs(b,0.8).subs(l,1).subs(gamma,1.1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1).subs(gamma,0.9),(phi,0,0.95))
```



[406]: <sympy.plotting.plot.Plot at 0x7f3ba32533a0>

[407]:
```
# in terms of lambda - own improvement
plot(a.subs(b,0.8).subs(l,0.9).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1.1).subs(gamma,1),(phi,0,0.95))
```



[407]: <sympy.plotting.plot.Plot at 0x7f3ba2e20be0>

Comparative Statics

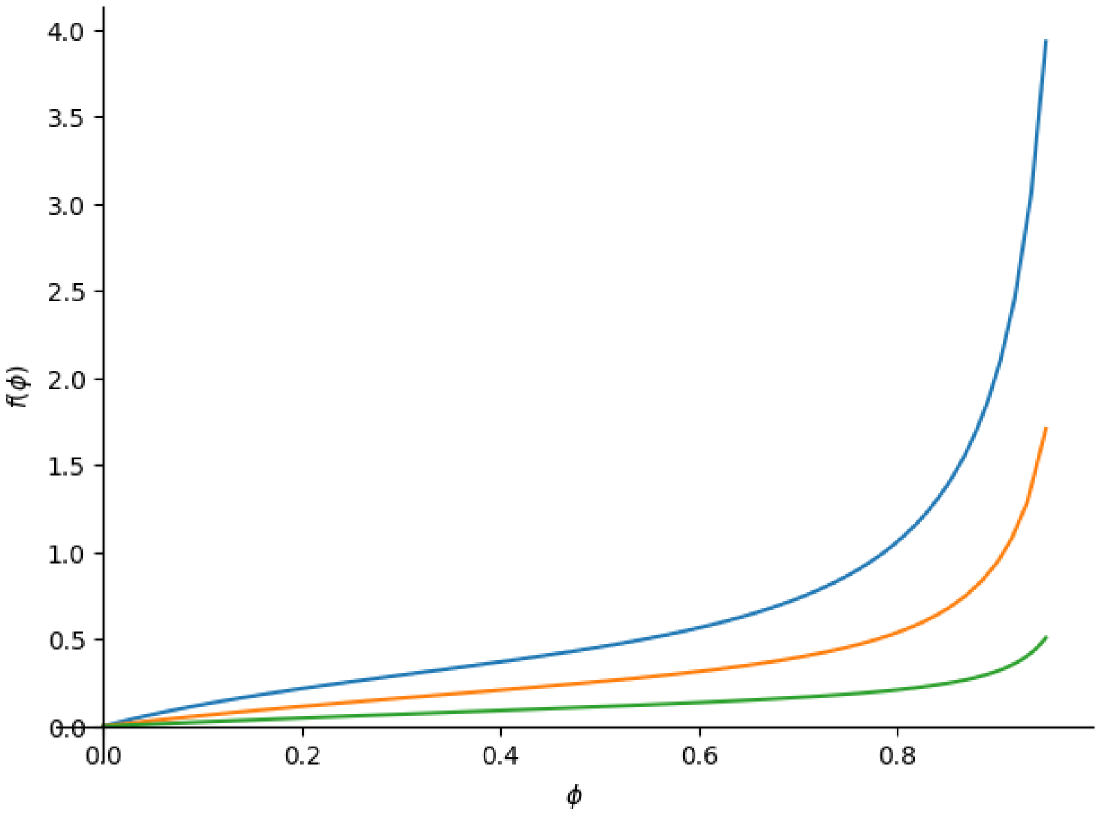
WRT Investing $x_1 (= x_2)$

- for different b (downstream subst. degree)
- for different γ (complementary good importnace)
- for different λ (premium to own-investment)

[408]:
```
# investing
```

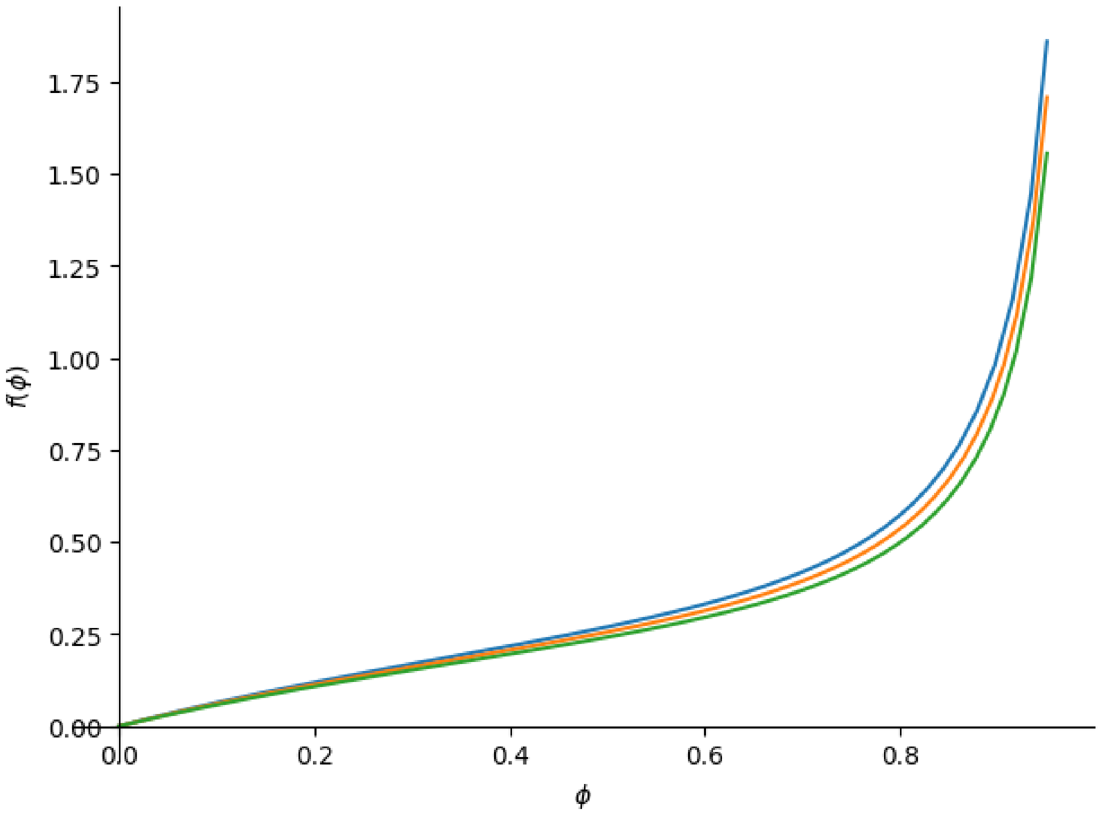
[409]:
```
a = simplify(x1sol.subs(w,wsol))
```

[410]:
```
plot(a.subs(b,0.7).subs(l,1).subs(gamma,1),a.subs(b,0.8).subs(l,1).
  ↪subs(gamma,1),a.subs(b,0.9).subs(l,1).subs(gamma,1),(phi,0,0.95))
```



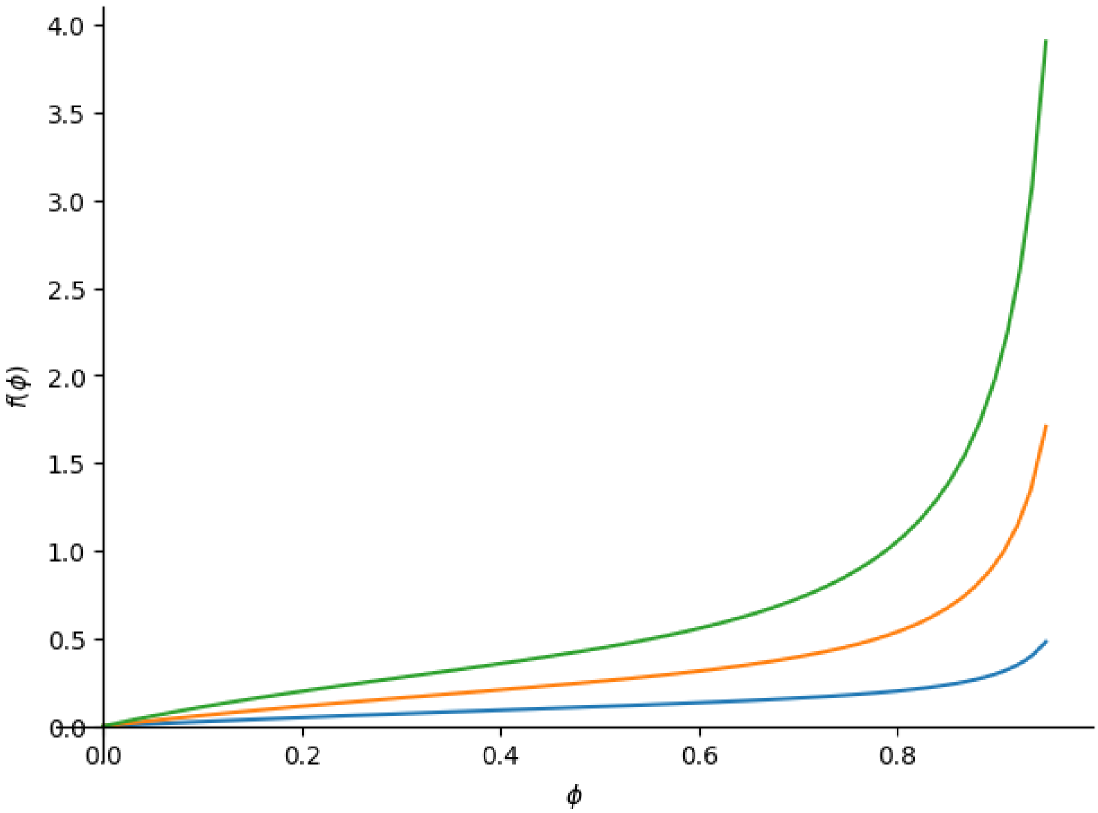
[410]: <sympy.plotting.plot.Plot at 0x7f3ba3132c50>

[411]:
```
plot(a.subs(b,0.8).subs(l,1).subs(gamma,1.1),a.subs(b,0.8).subs(l,1).
↪subs(gamma,1),a.subs(b,0.8).subs(l,1).subs(gamma,0.9),(phi,0,0.95))
```



[411]: <sympy.plotting.plot.Plot at 0x7f3ba3239de0>

[412]:
```
# in terms of lambda - own improvement
plot(a.subs(b,0.8).subs(l,0.9).subs(gamma,1),a.subs(b,0.8).subs(l,1).
    ↪subs(gamma,1),a.subs(b,0.8).subs(l,1.1).subs(gamma,1),(phi,0,0.95))
```



[412]: <sympy.plotting.plot.Plot at 0x7f3ba313f6a0>

Comparative Statics

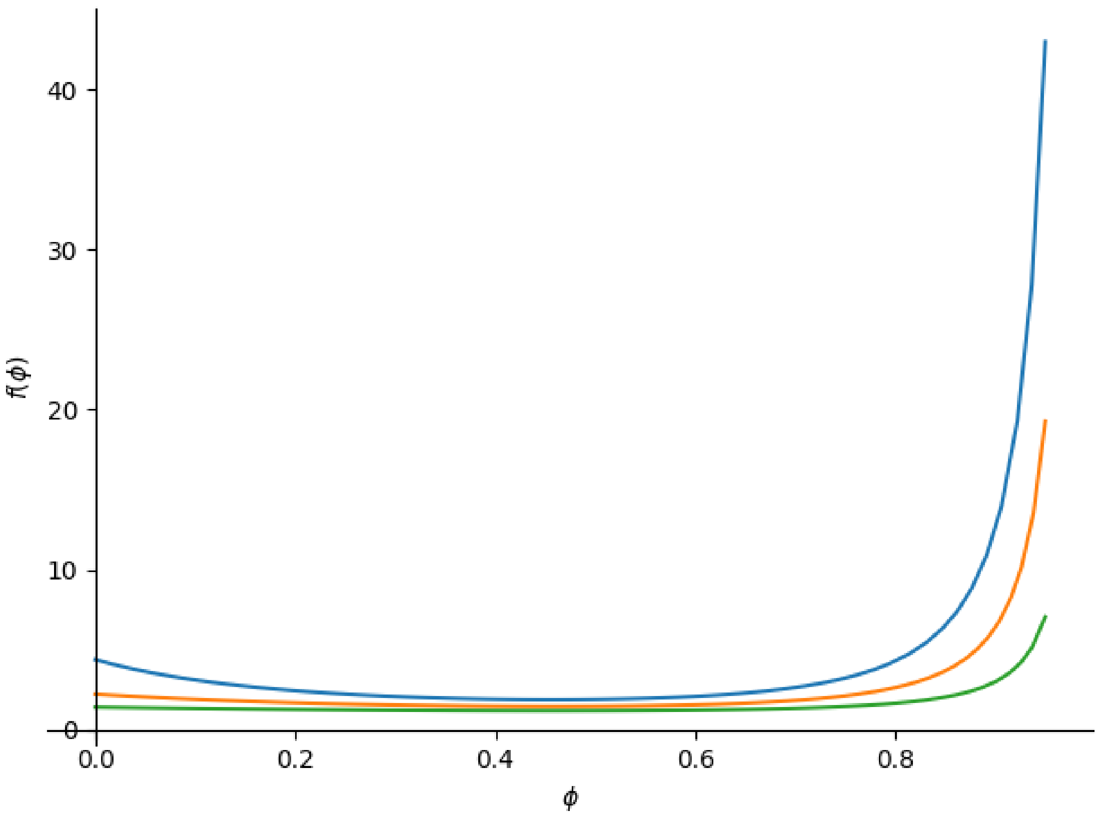
WRT Downstream Profits $\Pi_1 (= \Pi_2)$

- for different b (downstream subst. degree)
- for different γ (complementary good importnace)
- for different λ (premium to own-investment)

[413]:
```
# profits downstream firms
```

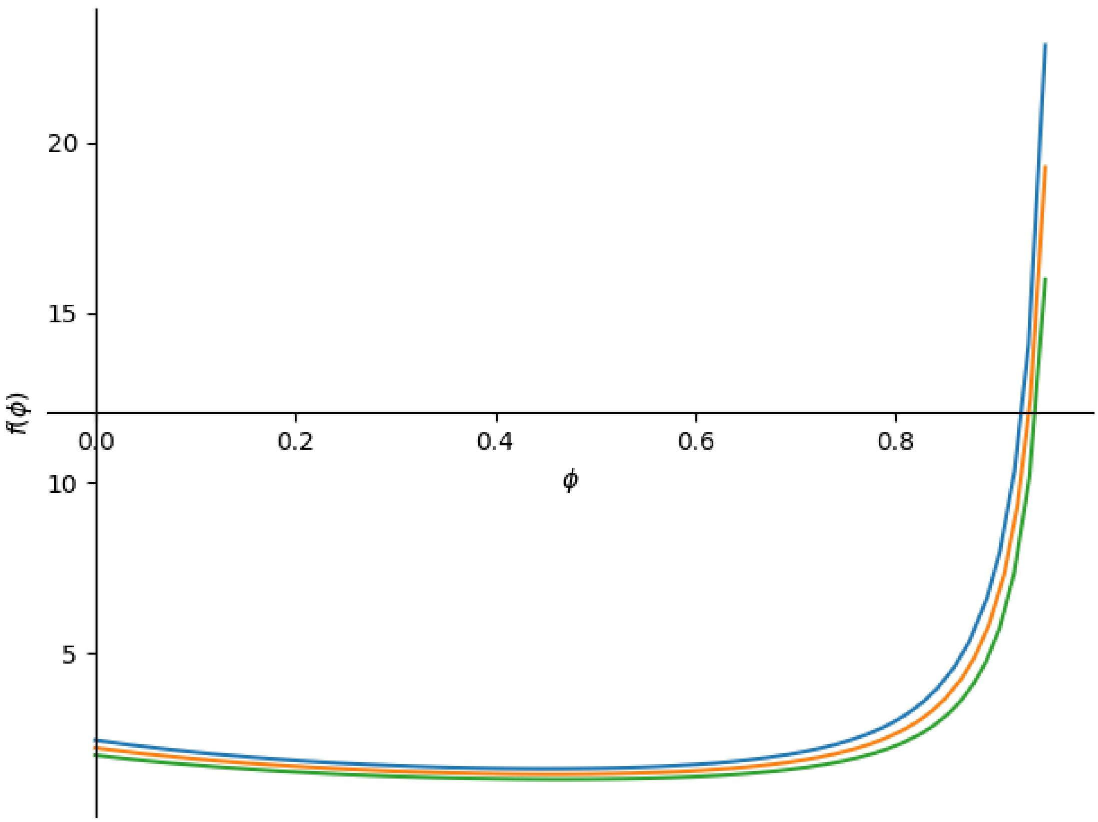
[414]:
```
a = simplify(Pi1.subs(x1,x1sol).subs(x2,x1sol).subs(q1,q1sol).subs(q2,q1sol).
 ↪subs(w,wsol))
u=a
```

[415]:
```
plot(a.subs(b,0.7).subs(l,1).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.9).subs(l,1).subs(gamma,1),(phi,0,0.95))
```



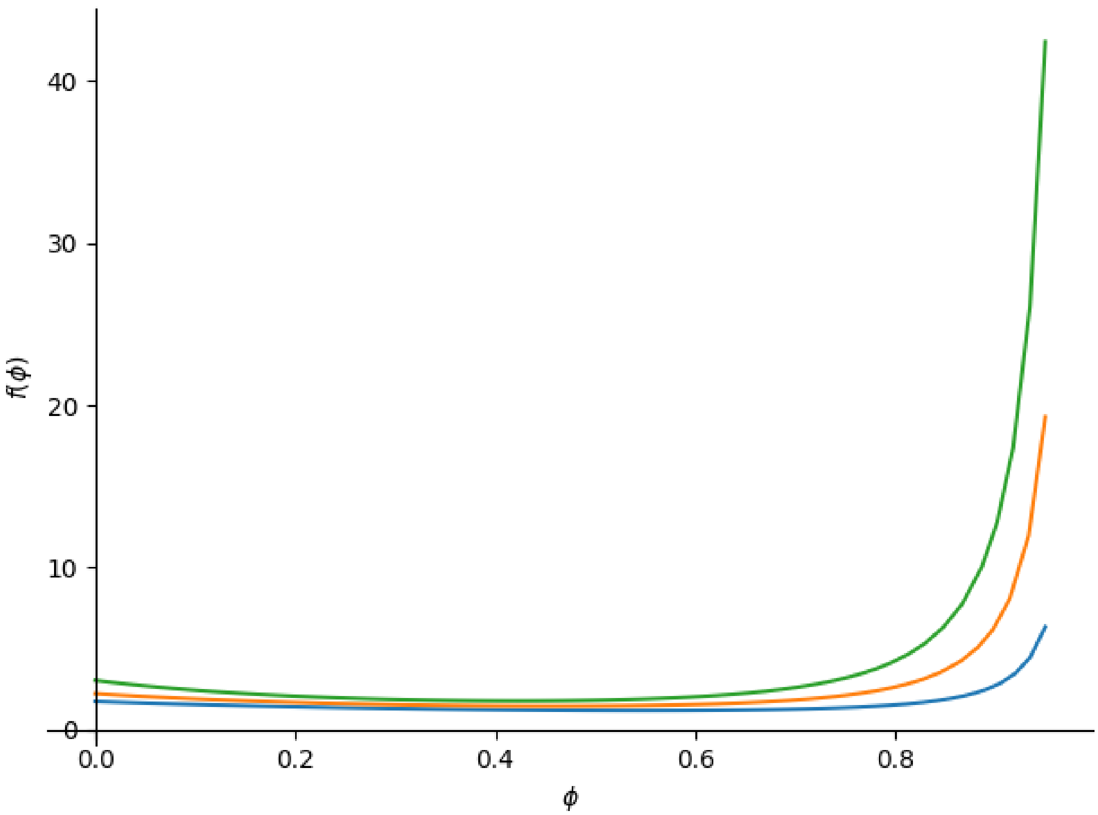
[415]: <sympy.plotting.plot.Plot at 0x7f3ba35acf10>

[416]:
```
plot(a.subs(b,0.8).subs(l,1).subs(gamma,1.1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1).subs(gamma,0.9),(phi,0,0.95))
```



[416]: <sympy.plotting.plot.Plot at 0x7f3ba2fc16c0>

[417]:
```
# in terms of lambda - own improvement
plot(a.subs(b,0.8).subs(l,0.9).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1.1).subs(gamma,1),(phi,0,0.95))
```



[417]: <sympy.plotting.plot.Plot at 0x7f3ba35acb80>

Comparative Statics

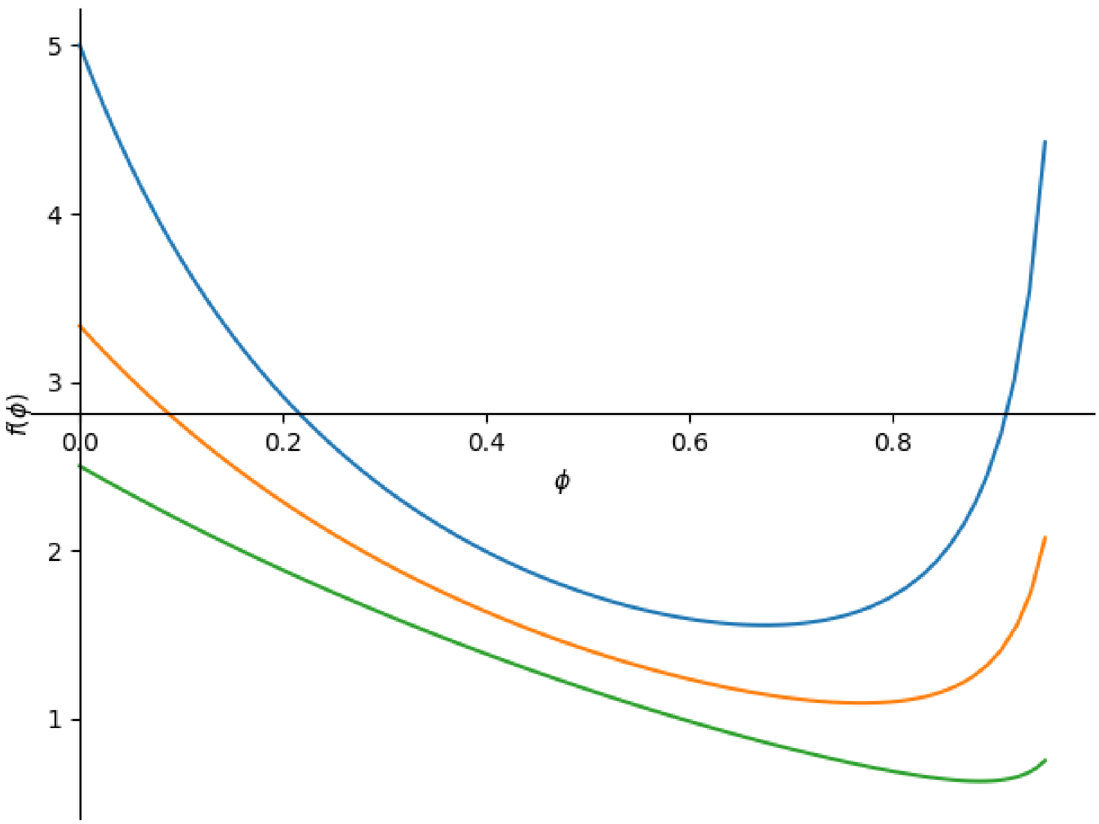
WRT Upstream Profits Π_A

- for different b (downstream subst. degree)
- for different γ (complementary good importnace)
- for different λ (premium to own-investment)

[418]:
```
# profits AI
```

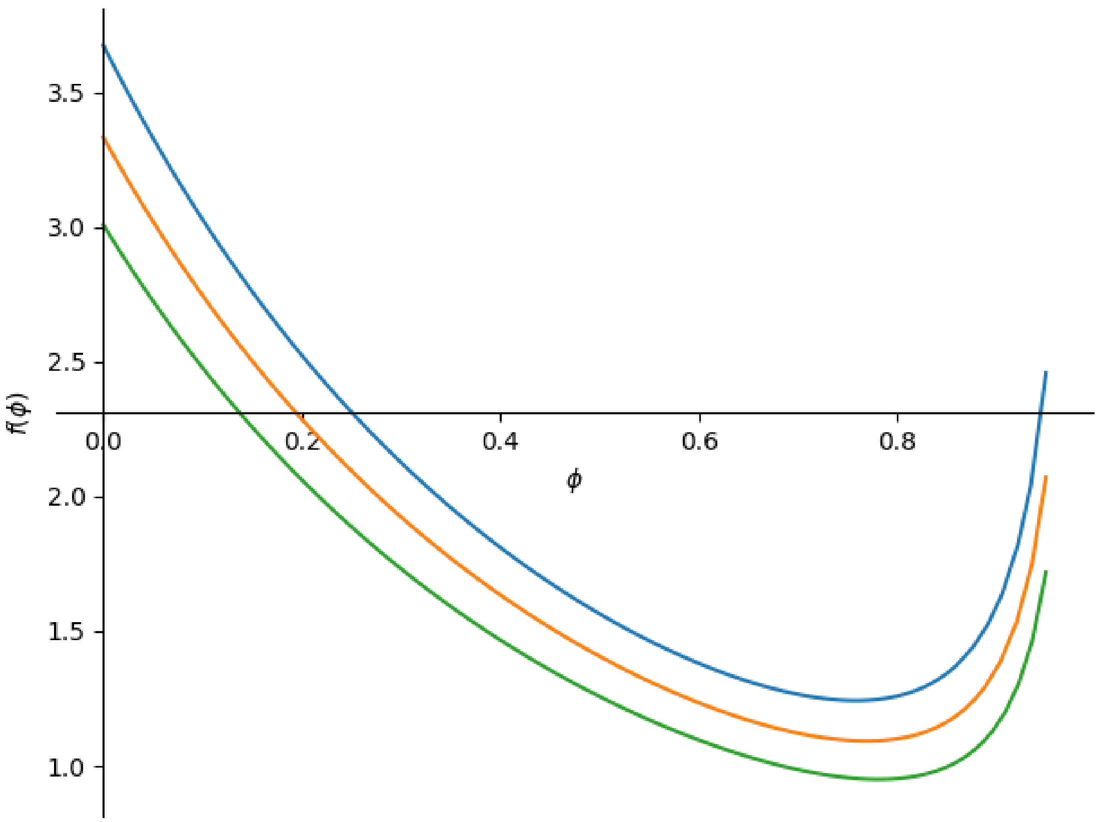
[419]:
```
a = simplify(PiA.subs(x1,x1sol).subs(x2,x1sol).subs(q1,q1sol).subs(q2,q1sol).
 ↪subs(w,wsol))
```

[420]:
```
# increasing b
plot(a.subs(b,0.7).subs(l,1).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.9).subs(l,1).subs(gamma,1),(phi,0,0.95))
```



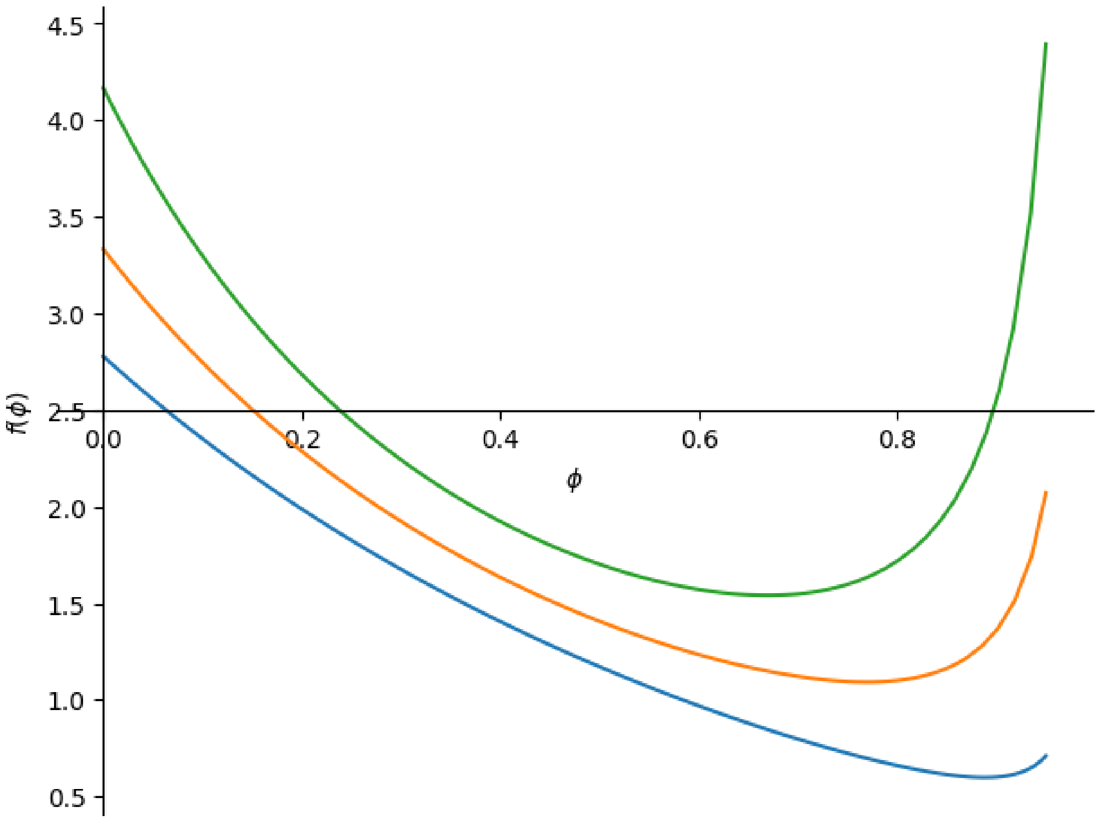
[420]: <sympy.plotting.plot.Plot at 0x7f3ba2f9a6b0>

[421]:
```
# increasing gamma
plot(a.subs(b,0.8).subs(l,1).subs(gamma,1.1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1).subs(gamma,0.9),(phi,0,0.95))
```



[421]: <sympy.plotting.plot.Plot at 0x7f3ba2cf9720>

[422]:
```
# in terms of lambda - own improvement
plot(a.subs(b,0.8).subs(l,0.9).subs(gamma,1),a.subs(b,0.8).subs(l,1).
  ↪subs(gamma,1),a.subs(b,0.8).subs(l,1.1).subs(gamma,1),(phi,0,0.95))
```



[422]: <sympy.plotting.plot.Plot at 0x7f3ba2cf8c70>

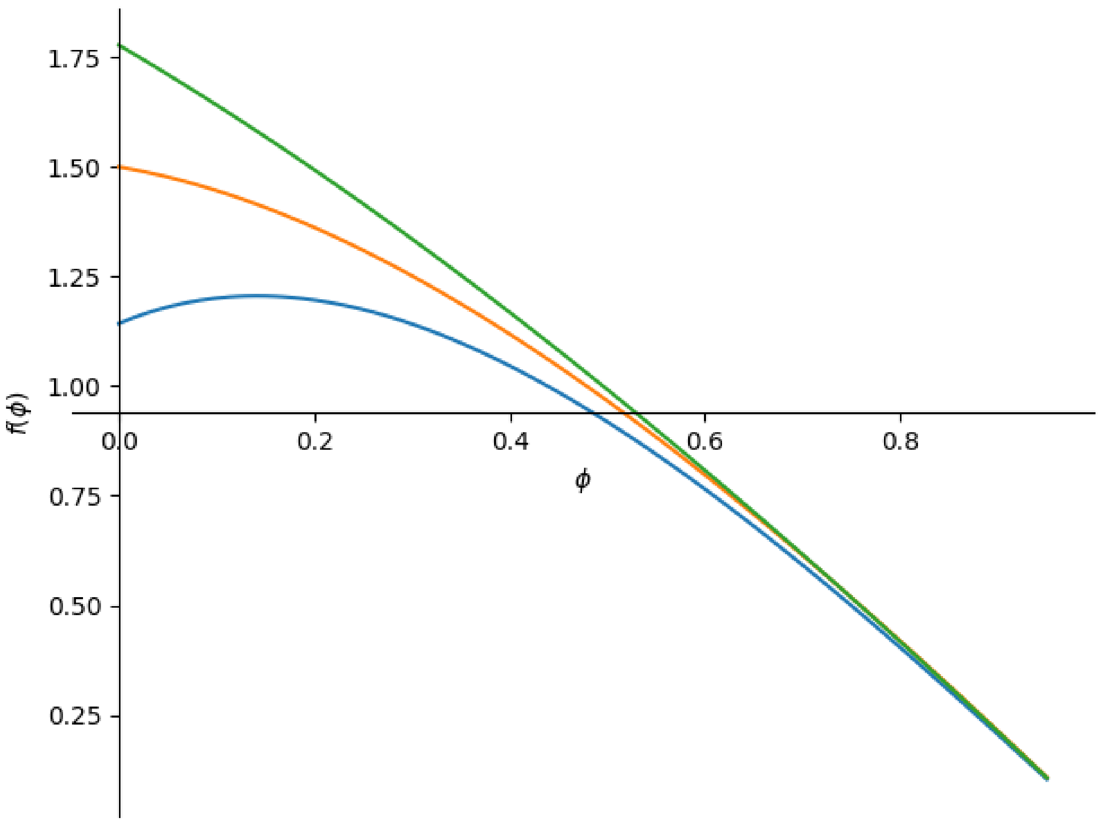
Comparative Statics

WRT Downstream / Upstream Profits Π_1/Π_A

- for different b (downstream subst. degree)
- for different γ (complementary good importnace)
- for different λ (premium to own-investment)

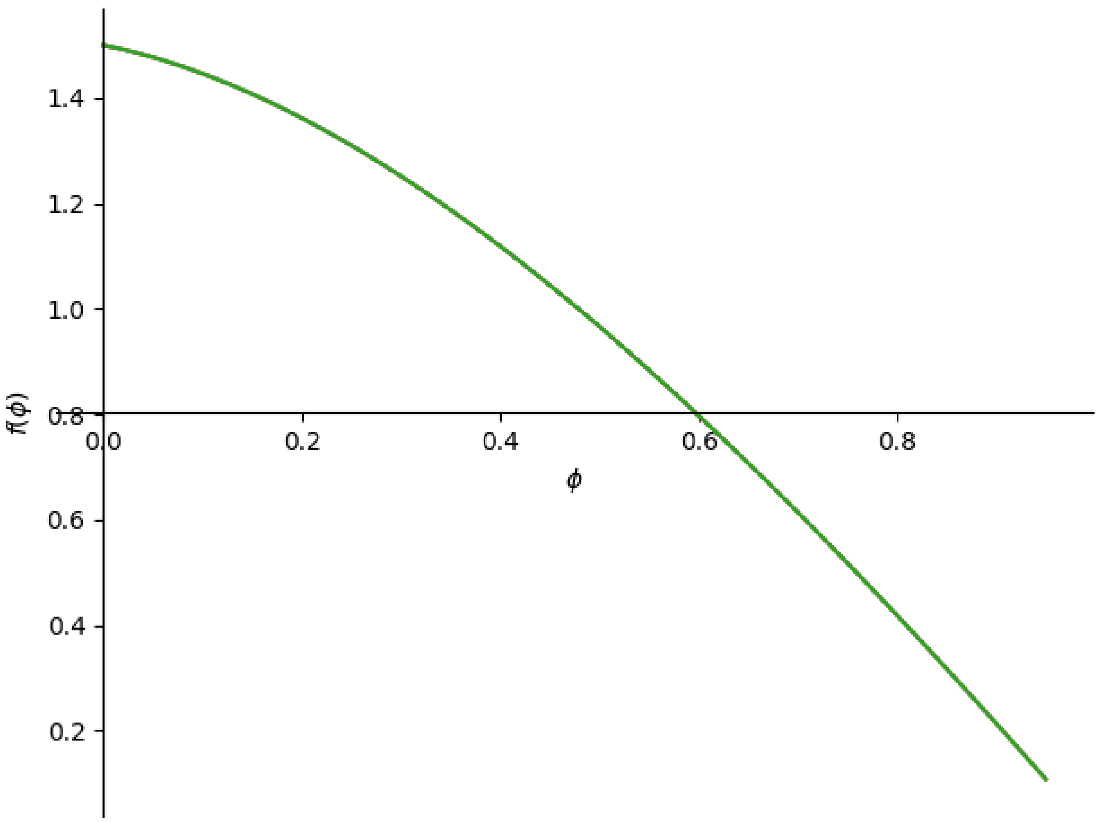
[423]:
```
a = a/u
```

[424]:
```
#increasing b
plot(a.subs(b,0.7).subs(l,1).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.9).subs(l,1).subs(gamma,1),(phi,0,0.95))
```



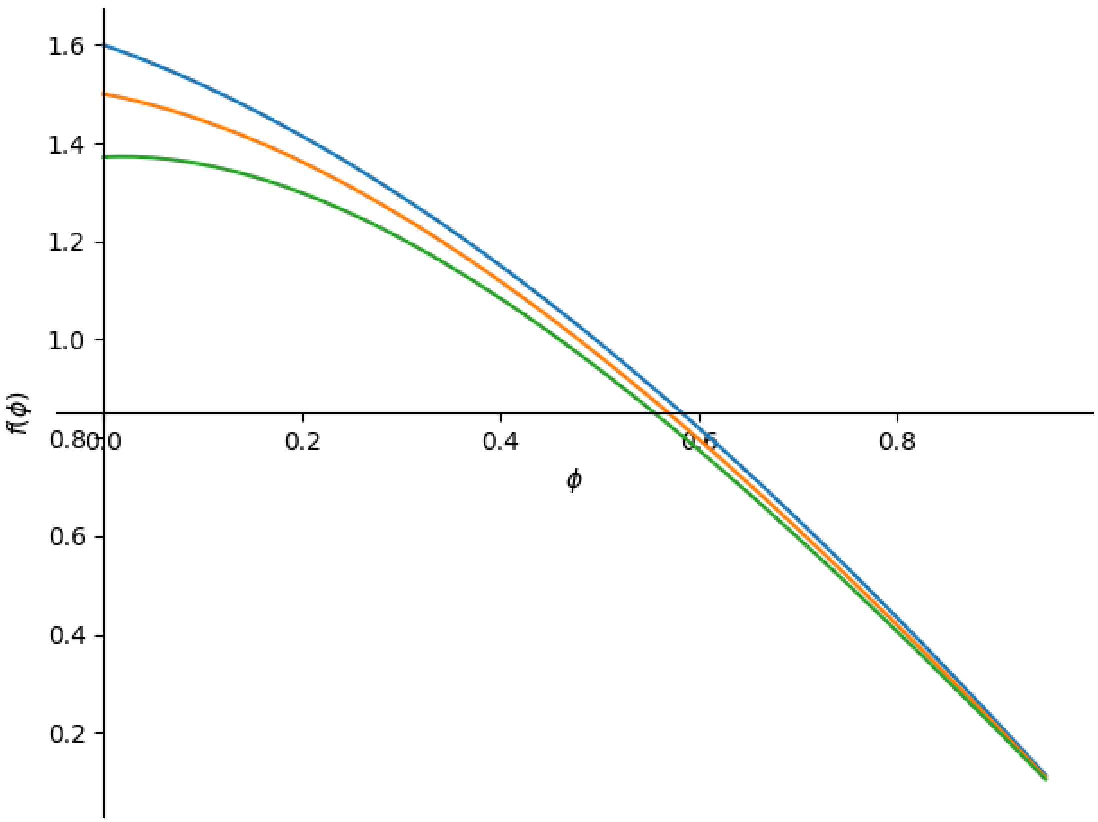
[424]: <sympy.plotting.plot.Plot at 0x7f3ba33b0dc0>

[425]:
```
# increasing gamma
plot(a.subs(b,0.8).subs(l,1).subs(gamma,1.1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1).subs(gamma,0.9),(phi,0,0.95))
```




```
[425]: <sympy.plotting.plot.Plot at 0x7f3ba2f9bee0>
```

```
[426]: # increasing lambda - own improvement
plot(a.subs(b,0.8).subs(l,0.9).subs(gamma,1),a.subs(b,0.8).subs(l,1).
 ↪subs(gamma,1),a.subs(b,0.8).subs(l,1.1).subs(gamma,1),(phi,0,0.95))
```



[426]: <sympy.plotting.plot.Plot at 0x7f3baf139180>

[426]: