Moral hazard and limited liability

Jonathan Conning[1] and Jonathan Morduch[2,3]

[1] Hunter College + The Graduate Center, CUNY .
[2] New York University .
[3] Financial Access Initiative (FAI) .

Abstract

A base model of moral hazard and limited liability with no monitoring nor intermediation.

This notebook describes a simple version of a two outcome model of moral hazard and limited liability, without a monitor. It demonstrates how minimum collateral requirements would vary depending on project conditions (probability of success, scope for moral hazard, etc).

Skip if you are familiar with this basic model.

1 No monitoring case

```
import numpy as np
import matplotlib.pyplot as plt
from ipywidgets import interact
```
   Model Parameters
```
X = 200       #Project returns under success (=0 under failure)
p = 0.7       #probability of success if diligent
q = 0.5       #probability of success if non -diligent
I = 100       #project investment
gamma = 1     #outside lenders' cost of funds
B = 20
F = 0         # Fixed cost of loan
Amax = 120
Amin = p*B/(p -q) -(p*X - gamma*I) + F
print('Expected project return {}'.format(p*X))
```

```
print('Expected cost of funds {}'.format(gamma*I+F))
print('LL rent at A=0 is {:5.1f}'.format(p*B/(p -q)))
print('Min collateral requirement is {:5.1f}'.format(Amin))
print('LL rent at A= Amin is {}'.format( -Amin+p*B/(p -q)))
Expected project return 140.0
Expected cost of funds 100
LL rent at A=0 is  70.0
Min collateral requirement is  30.0
LL rent at A= Amin is 40.0
```

2 Minimum collateral requirements

Note that in this simple two outcome world, the project can either end in "failure" (zero output) or "success" (output X). The contract states outcome-contingent payoffs of $(s_f, s_s) = (-A, s)$ to the entrepreneur/borrower, where A represents the pledged (seizable) assets. The bank therefore receives $(R_f, R_s) = (A, X - s)$.

We will derive the minimum collateral requirement A that the bank will require that a borrower provide. This is the "skin in the game" that the lender will require to diminish the scope for moral hazard via the terms of the contract.

Bank Participation constraint:

$$p(X - s) + (1 - p)A \geq \gamma I + F \tag{1}$$

Incentive compatibility constraint when entrepreneur has pledgeable assets A:

$$s \geq -A + \frac{B}{\Delta} \tag{2}$$

The **limited liability** constraint is written:

$$s \geq -A \tag{3}$$

In fact, a limited liability constraint must hold for each outcome state to make sure that the value of promised repayment in that state does not exceed the value of output plus any pledgeable collateral A. In this two-state world, however, the constraints are $(X - s) \leq A$ and $(0 - s) \leq A$, but the latter will always bind first, so we need only focus on this first constraint, as re-written and stated above.

$$\underline{A} = \frac{pB}{p - q} - (pX - \gamma I) \tag{4}$$

Diagram plots We want to plot the linear constraints and/or objectives lines in $s_f - s_s$ space. For example a zero profit condition:

$$p(X - s_s) + (1 - p)(0 - s_f) - \gamma I = 0 \tag{5}$$

can be rearranged to plot s_s as a function of s_f:

$$s_s = X - \frac{\gamma I}{p} - \frac{1-p}{p} s_f \qquad (6)$$

Since we'll be illustrating things below with a variety of linear plots, we will economize on code by writing a simple 'function factory' `line_function(a,b)` that will allow us to easily create parameterized line functions. Another `plot_contraints()` function will plot the linear participation, incentive compatibility, and limited liability constraints to identify the optimum contract and minimum collateral requirement.

```
def line_function(a, b):
    '''return a line function of the form f(x) = a + b*x'''
    def f(x):
        return a + b*x
    return f


def plot_constraints(*args, cfmin = -100, cfmax = 200):
    '''Plotting helper.  This plots contract constraints
    Accepts a variable number of lines/constraint passed as functions
    that are then plotted on the same diagram'''
    cf = np.linspace(cfmin, cfmax, cfmax -cfmin+1)  # plot range
    for count, fn in enumerate(args):
        plt.plot(cf, fn(cf))
    #plt.axhline(0, linestyle=':', color='black')
    plt.axvline(0, linestyle=':', color='black')
    plt.ylim( -10,300)
zeroprofit = line_function(X - I/p - F/p, -(1 -p)/p)
ic = line_function(B/(p -q), 1)        # IC
plot_constraints(ic, zeroprofit)
minA = p*B/(p -q) -(p*X - gamma*I)

print(f'Contract: ( -A, s) = ({ -minA:0.1f}, {zeroprofit( -minA):0.1f})')

plt.plot( -minA, zeroprofit( -minA), marker='*')
plt.ylim(0,300)

plt.ylabel(r'$s$')
plt.xlabel(r'$A$')
ax = plt.gca()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.text(100, zeroprofit(100), r'$PC(0)$')
plt.text(110, ic(100), r'$IC$')
plt.show();
Contract: ( -A, s) = ( -30.0, 70.0)
```
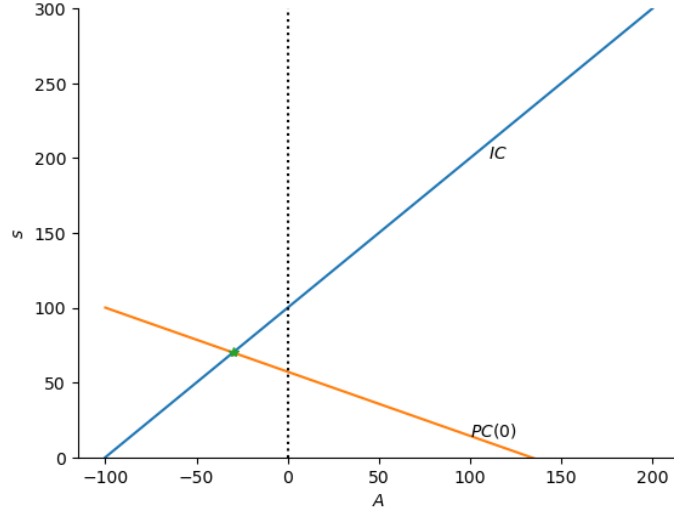
3

A limited liability constraint would appear as a vertical line (e.g. $s_f - A$). As drawn above the bank's zero-profit participation constraint $PC(0)$ intersects the incentive compatibility constraint IC at the point

$$s_f = -\underline{A}s_s = -\underline{A} + \frac{B}{\Delta} \tag{7}$$

Any borrower with assets A less than $\underline{A}$ would not be able to borrow from the bank.