Credit Allocation in Heterogeneous Banking Systems

Marcello D'Amato [†] , Christian Di Pietro [‡] , Marco M. Sorge [§]

Supplementary material
Code for R-based simulations

```
# Load package distr in order to create mixtures of known absolutely continuous probability distributions in unit interval
library(distr)
# Parametrize the model
Rn <- 1.3
Ri <- 2.2
pi <-6
u0 <- 0.1
hatRi <- pi*Ri/(Ri+u0)
# Arbitrarily set small number for plotting purposes
epsilon <- 0.01
# Generate Beta distribution Be(α, β) with shape parameters α > 0 and β > 0, and compute total revenues (TR) and total costs (TC) of uninformed banks, where x := ρ_n
integrand <- function(q){q*dbeta(q,α,β)}
integrand2 <- function(q){Rn*dbeta(q,α,β)}
TR <- Vectorize(function(x) x*integrate(integrand, lower=u0/(pi-x), upper=Ri/x)$value)
TC <- Vectorize(function(x) integrate(integrand2, lower=u0/(pi-x), upper=Ri/x)$value)
# Find equilibria with both bank types as solutions to TR=TC in interval (Ri, hatRi)
rt <- uniroot(function(x) TR(x)-TC(x), c(Ri, hatRi), tol=1e-8)
rt$root
plot(TR, Ri+epsilon, hatRi-epsilon, col="blue")
plot(TC, Ri+epsilon, hatRi-epsilon, col="red", add=TRUE)
# If no root in (Ri, hatRi) exists, compute unique equilibrium with only uninformed banks as follows
TRn <- Vectorize(function(x) x*integrate(integrand, lower=u0/(pi-x), upper=1)$value)
TCn <- Vectorize(function(x) integrate(integrand2, lower=u0/(pi-x), upper=1)$value)
rt <- uniroot(function(x) TRn(x)-TCn(x), c(Rn, Ri), tol=1e-8)
rt$root
plot(TRn, Rn+epsilon, Ri, col="blue")
plot(TCn, Rn+epsilon, Ri, col="red", add=TRUE)
# Generate mixture of Beta distributions M to solve for multiple equilibria (Table 1).
mix <- UnivarMixingDistribution(Beta(shape1=4, shape2=2), Beta(shape1=0.8, shape2=8.3), mixCoeff=c(0.75, 0.25))
d <- d(mix)
integrand <- function(q){q*d(q)}
integrand2 <- function(q){Rn*d(q)}
TR <- Vectorize(function(x) x*integrate(integrand, lower=u0/(pi-x), upper=Ri/x)$value)
TC <- Vectorize(function(x) integrate(integrand2, lower=u0/(pi-x), upper=Ri/x)$value)
plot(TR, Ri+epsilon, hatRi-epsilon, col="blue")
```

[†]University of Napoli Suor Orsola Benincasa, CSEF and CELPE
[‡]University of Napoli Parthenope and CELPE
[§]University of Salerno, University of Göttingen and CSEF

```
plot(TC, Ri+epsilon, hatRi-epsilon, col="red", add=TRUE)
# In the case when there exist multiple roots on the given interval, uniroot will only find and return one of them. Searching for all the roots requires calling uniroot multiple times while specifying small intervals each containing only one root.
rt <- uniroot(function(x) TR(x)-TC(x), c(Ri, hatRi), tol=1e-8)
rt$root
# Fragile equilibria (Table 2)
Rn <- 1.3
Ri <- 2.2
pi <- 6
u0 <- 0.1
hatRi <- pi*Ri/(Ri+u0)
integrand <- function(q){q*dbeta(q,1,1)}
integrand2 <- function(q){Rn*dbeta(q,1,1)}
TR <- Vectorize(function(x) x*integrate(integrand, lower=u0/(pi-x), upper=Ri/x)$value)
TC <- Vectorize(function(x) integrate(integrand2, lower=u0/(pi-x), upper=Ri/x)$value)
rtold <- uniroot(function(x) TR(x)-TC(x), c(Ri, hatRi), tol=1e-8)
rtold$root
Ri <- 2.3
hatRi <- pi*Ri/(Ri+u0)
rtnew <- uniroot(function(x) TR(x)-TC(x), c(Ri, hatRi), tol=1e-8)
rtnew$root
all.equal(TR(rt$root), TC(rt$root))
# Compute market segments
oldQi <- 2.2/rtold$root
newQi <- 2.3/rtnew$root
changeQi <- 100* (oldQi-newQi)/(1-oldQi)
oldQn <- u0/(pi-rtold$root) newQn <- u0/(pi-rtnew$root)
changeQn <- 100*((newQi-newQn)-(oldQi-oldQn))/(oldQi-oldQn)
Rn <- 1.3
Ri <- 2.2
pi <- 6
u0 <- 0.1
hatRi <- pi*Ri/(Ri+u0)
mix <- UnivarMixingDistribution(Beta(shape1=4, shape2=2), Beta(shape1=0.8, shape2=8.3), mix-
Coeff=c(0.75, 0.25))
d <- d(mix)
integrand <- function(q){q*d(q)}
integrand2 <- function(q){Rn*d(q)}
TR <- Vectorize(function(x) x*integrate(integrand, lower=u0/(pi-x), upper=Ri/x)$value)
TC <- Vectorize(function(x) integrate(integrand2, lower=u0/(pi-x), upper=Ri/x)$value)
rtnew <- uniroot(function(x) TRn(x)-TCn(x), c(Rn, Ri), tol=1e-8)
rtnew$root
oldQi <- 2.2/rtold$root
newQi <- 2.3/rtnew$root
changeQi <- 100* (oldQi-newQi)/(1-oldQi)
oldQn <- u0/(pi-rtold$root) newQn <- u0/(pi-rtnew$root)
changeQn <- 100*((newQi-newQn)-(oldQi-oldQn))/(oldQi-oldQn)
```