Code for the Baseline Simulation: The Model of Cartels in the Presence of a Corporate Leniency Program

The case of non-linear p(λL+R):

$$p(\lambda L+R) = \frac{\tau}{\xi+\nu\ (\lambda L+R)^{\rho}}$$

November 11, 2012

Myong-Hun Chang
Department of Economics
Cleveland State University
Cleveland, OH 44115
216-687-4523
m.chang@csuohio.edu

Preliminary

Call Packages and Set the Text Style

```
<< MultivariateStatistics`;

$TextStyle = {FontFamily -> "Helvetica", FontSize -> 10}

{FontFamily → Helvetica, FontSize → 10}
```

Specify the distributions

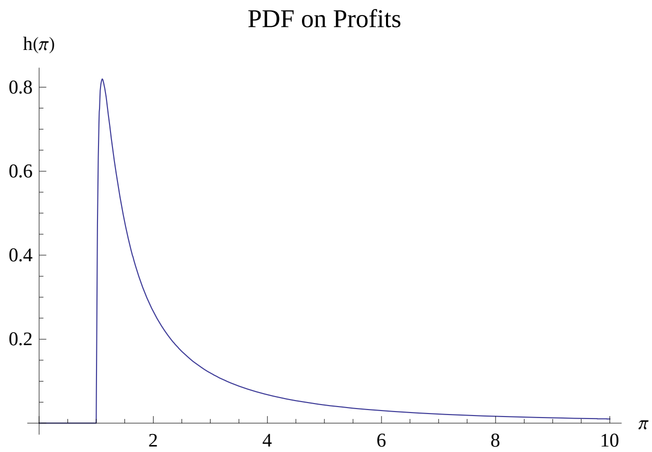
Specify LogNormal Distribution on Profit Shocks

```
pftmin = 1; (* π *)
pftmax = 100; (* π̄ *)
pftdist = LogNormalDistribution[0, 1.5]; (* CDF on profit shocks *)

μ = NIntegrate[x * PDF[pftdist, x - pftmin], {x, pftmin, ∞}] (* Mean of the distribution *)

4.08022
```

```
Plot[PDF[pftdist, i - pftmin], {i, 0, 10}, PlotRange → All,
 AxesLabel → {"π", "h(π)"}, PlotLabel → "PDF on Profits"]
```

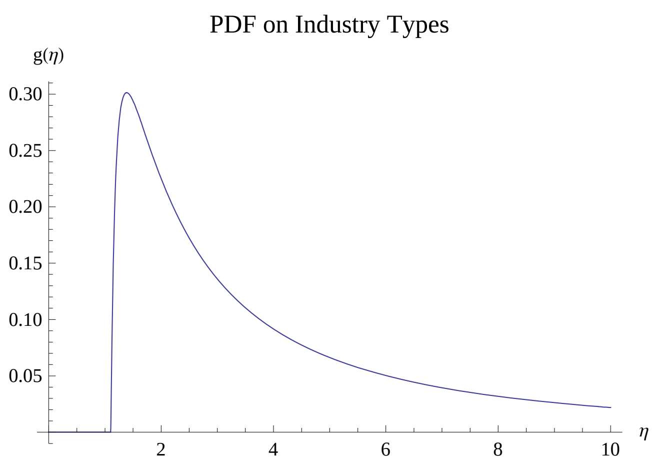



```
(* Note: LogNormalDistribution is defined for [0,∞].  Since π∈[pftmin, pftmax],
we write PDF[pftdist,π-pftmin]. *)
```

Specify Log Normal Distribution on Industry Types

```
ηmin = 1.1; (* η *)
ηmax = 10; (* η̄ *)
ηdist = LogNormalDistribution[1, 1.5];(* CDF on industry types *)
```

```
Plot[PDF[ηdist, i - ηmin], {i, 0, 10}, PlotRange → All,
 AxesLabel → {"η", "g(η)"}, PlotLabel → "PDF on Industry Types"]
```




```
NIntegrate[i * PDF[ηdist, i - ηmin], {i, ηmin, ∞}](* Mean of the distribution *)
```

```
9.4729
```

```
(* Note: LogNormalDistribution is defined for [0,∞].  Since η∈[ηmin, ηmax],
we write PDF[ηdist,η-ηmin]. *)
```

```
(* Prepare η for computing η̂. *)
(* Specify the level of precision *)
ηprecis = 0.01;
Yprecis = 0.000001;
Sprecis = 0.01;
(* Create the grid, Γ(ηmin, ηmax), by dividing [ηmin, ηmax] into n equal pieces *)
n = 1001;
h = (ηmax - ηmin)/(n - 1);
eitaVal = Table[N[j], {j, ηmin, ηmax, h}];
eitaPos = Table[j, {j, 1, Length[eitaVal], 1}];
```

Cartel Model with (θ, ω) Leniency Program

Define Functions and Procedures

Functions and Procedure for computing $\hat{\eta}$

Functions for Computing $\hat{\eta}$

```
W[Y_] := ((1 - κ) * (1 - δ) * α * μ + κ * Y)/(1 - δ * (1 - κ));

ϕ[Y_ , η_, xr_, xs_] := 1/((1 - δ * (1 - κ)) * (η - 1)) (δ * (1 - q * xr * xs) * (1 - κ) * (Y - α * μ) -
    γ * (1 - δ * (1 - κ)) * (q * xr * xs - Min[q * xr * xs, θ]) * (Y - α * μ));

A[Y_, xr_, xs_] := δ * W[Y] - (1 - δ) * q * xr * xs * γ * (Y - α * μ);

B[Y_, xr_, xs_] := δ * (1 - q * xr * xs) * Y + δ * q * xr * xs * W[Y] - (1 - δ) * q * xr * xs * γ * (Y - α * μ);

AA[Y_] := δ * W[Y] - (1 - δ) * ω * γ * (Y - α * μ);

ψ[Y_ , η_, xr_, xs_] :=
  If[q * xr * xs ≤ θ,
```

```
    If[ϕ[Y, η, xr, xs] ≤ pftmin,
      (1 - δ) * α * μ + A[Y, xr, xs],
      If[ϕ[Y, η, xr, xs] ≥ pftmax,
        (1 - δ) * μ + B[Y, xr, xs],
        (1 - δ) * α * μ +
          (1 - δ) * (1 - α) * NIntegrate[x * PDF[pftdist, x - pftmin], {x, pftmin, ϕ[Y, η, xr, xs]}] +
          (δ * (1 - q * xr * xs) * ((1 - δ) * (1 - κ))/(1 - δ * (1 - κ)) * (Y - α * μ)) * CDF[pftdist, ϕ[Y, η, xr, xs] - pftmin] +
          A[Y, xr, xs]]],
    If[ϕ[Y, η, xr, xs] ≤ pftmin,
      (1 - δ) * α * μ + AA[Y],
      If[ϕ[Y, η, xr, xs] ≥ pftmax,
        (1 - δ) * μ + B[Y, xr, xs],
        (1 - δ) * α * μ +
          (1 - δ) * (1 - α) * NIntegrate[x * PDF[pftdist, x - pftmin], {x, pftmin, ϕ[Y, η, xr, xs]}] +
          (δ * (1 - q * xr * xs) * ((1 - δ) * (1 - κ))/(1 - δ * (1 - κ)) * (Y - α * μ) + (1 - δ) * (ω - q * xr * xs) * γ * (Y - α * μ)) *
            CDF[pftdist, ϕ[Y, η, xr, xs] - pftmin] + AA[Y]]]
  ];

(* Identify Y*(η) *)
optY[a_, b_, c_, xr_, xs_] :=
  (x = a;
    While[x - ψ[x, b, xr, xs] ≥ c,
      x = ψ[x, b, xr, xs];
    ];
    x
  );

(* bisection method for locating η̂ *)
bisect[a_, b_, xr_, xs_] :=
  (mid = a + Floor[(b - a)/2];
    Ymid = optY[μ, eitaVal[[mid]], Yprecis, xr, xs];
    If[Ymid - α * μ ≤ Sprecis,
      ceil = mid;
      Yceil = Ymid,
      flr = mid;
```

```
    Yflr = Ymid;
   ];
  );
```

Procedure for Computing $\hat{\eta}$

```
procEitaHat[yr_, ys_] :=
 (
(* Identify fixed points with the bisection method *)
flr = 1;
ceil = Length[eitaVal];
Yflr = optY[μ, eitaVal[[flr]], Yprecis, yr, ys];
Yceil = optY[μ, eitaVal[[ceil]], Yprecis, yr, ys];
If[(Yflr - α * μ > Sprecis) && (Yceil - α * μ ≤ Sprecis),
   While[ceil - flr > 1,
    bisect[flr, ceil, yr, ys];
   ];
   ηhat = eitaVal[[flr]];
   ηhatPos = flr;
   ,
   If[(Yflr - α * μ ≤ Sprecis) && (Yceil - α * μ ≤ Sprecis),
     ηhat = eitaVal[[flr]];
     ηhatPos = flr;
     ,
     Print["η̂ is out of bounds: η̂=η̅."];
    ];
  ];
(* Compute Y* and ϕ* *)
Ystar = Table[optY[μ, eitaVal[[i]], Yprecis, yr, ys], {i, 1, ηhatPos}];
phistar = Table[ϕ[Ystar[[i]], eitaVal[[i]], yr, ys], {i, 1, ηhatPos}];

  (* return η̂, η̂'s position, Y*, ϕ* *)
  {ηhat, ηhatPos, Ystar, phistar}
 )
```

Functions and Procedure for computing p[L, R]

Functions for computing L: mass of cartel cases generated by the leniency program

```
F1[x_, xr_, xs_] := (If[eitaVal[[x]] == ηmin, 0, (1 - CDF[pftdist, phistar[[x]] - pftmin]) *
      (κ * (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) /
      (1 - (1 - κ) * (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) *
     PDF[ηdist, eitaVal[[x]] - ηmin]]);
```

Functions for computing R: mass of cartel cases not through the leniency program

```
F2[x_, xr_, xs_] := (If[eitaVal[[x]] == ηmin, 0,
    (q * xr * κ * (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) /
    (1 - (1 - κ) * (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) *
     PDF[ηdist, eitaVal[[x]] - ηmin]]);
F3[x_, xr_, xs_] := (If[eitaVal[[x]] == ηmin, 0,
    ((q * xr * CDF[pftdist, phistar[[x]] - pftmin]) * κ *
        (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) /
      (1 - (1 - κ) * (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) *
     PDF[ηdist, eitaVal[[x]] - ηmin]]);
```

Procedure for computing p[L, R]

```
procPLR[yr_, ys_] :=
  (

(* compute η̂ *)
procEitaHat[yr, ys]; (* ηdata={ηhat,ηhatPos,Ystar,phistar} *)

   (***** L:mass of cartel cases
     generated by the leniency program *****)
   (***** 0 if q*r*s≤θ *****)
   (*****
```

$$\int_{\underline{\eta}}^{\hat{\eta}} (1-H(\phi^*(qrs,\eta))) \frac{\kappa(1-qrs)H(\phi^*(qrs,\eta))}{1-(1-\kappa)(1-qrs)H(\phi^*(qrs,\eta))} g(\eta)d\eta \quad \text{if } q*r*s>\theta \quad *****)$$

```
    If[ηhatPos > 0,
     If[q * yr * ys ≤ θ,
       L = 0,
       pnts = Table[h * F1[i, yr, ys], {i, 1, ηhatPos, 1}];
       pnts = ReplacePart[pnts, h/2 * F1[1, yr, ys], 1];
       pnts = ReplacePart[pnts, h/2 * F1[ηhatPos, yr, ys], ηhatPos];
       L = Apply[Plus, pnts]];
     ,
```

```
    Print["NO CARTELS EVER: η̂ = η"];
    L = 0;
   ];


   (***** R:mass of cartel cases
     not through the leniency program *****)
   (***** qr ∫_η^η̂ κ(1-qrs)H(ϕ*(qrs,η)) / (1-(1-κ)(1-σ)H(ϕ*(η))) g(η) dη  if q*r*s≤θ *****)
   (*****
    qr ∫_η^η̂ H(ϕ*(qrs,η)) κ(1-qrs)H(ϕ*(qrs,η)) / (1-(1-κ)(1-qrs)H(ϕ*(qrs,η))) g(η) dη  if q*r*s>θ *****)
   If[ηhatPos > 0,
    If[q * yr * ys ≤ θ,
     pnts = Table[h * F2[i, yr, ys], {i, 1, ηhatPos, 1}];
     pnts = ReplacePart[pnts, h/2 * F2[1, yr, ys], 1];
     pnts = ReplacePart[pnts, h/2 * F2[ηhatPos, yr, ys], ηhatPos],
     pnts = Table[h * F3[i, yr, ys], {i, 1, ηhatPos, 1}];
     pnts = ReplacePart[pnts, h/2 * F3[1, yr, ys], 1];
     pnts = ReplacePart[pnts, h/2 * F3[ηhatPos, yr, ys], ηhatPos]];
    R = Apply[Plus, pnts];
    ,
    Print["NO CARTELS EVER: η̂ = η"];
    R = 0;
   ];


   (***** Compute s=p(L,R) = τ / (ξ+ν(λL+R)^ρ),
   where ξ≥τ *****)
 {τ / (ξ + ν * (λ * L + R)^ρ), L, R, L + ys * R}
  );
```

Functions and Procedure for computing $1 - \tilde{\beta}(0)$: Mass of Cartels

Functions for computing $1-\tilde{\beta}(0)$: Mass of Cartels

```
b0[x_, xr_, xs_] := (1 - (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) /
                    (1 - (1 - κ) * (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]);
f0[x_, xr_, xs_] :=
  (If[eitaVal[[x]] == ηmin, 0, (1 - b0[x, xr, xs]) * PDF[ηdist, eitaVal[[x]] - ηmin]]);
```

Procedure for computing 1-$\tilde{\beta}$(0) : Mass of Cartels

```
procCMS[yr_, ys_] :=
  (If[ηhatPos > 1,
     pnts = Table[h * f0[i, yr, ys], {i, 1, ηhatPos, 1}];
     pnts = ReplacePart[pnts, h/2 * f0[1, yr, ys], 1];
     pnts = ReplacePart[pnts, h/2 * f0[ηhatPos, yr, ys], ηhatPos];
     cartelMass = Apply[Plus, pnts];
     ,
     Print["NO CARTELS EVER: η̂ = η̲"];
     cartelMass = 0;
   ];
   cartelMass
  );
```

Functions and Procedure for computing cartel durations

Functions for computing $\tilde{\beta}$(l) : Mass of Cartels of Duration l

```
bl[l_, x_, xr_, xs_] := (1 - (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) /
    (1 - (1 - κ) * (1 - q * xr * xs) * CDF[pftdist, phistar[[x]] - pftmin]) *
   κ * (CDF[pftdist, phistar[[x]] - pftmin] * (1 - q * xr * xs))^l;
fl[l_, x_, xr_, xs_] := (If[eitaVal[[x]] == ηmin, 0,
     bl[l, x, xr, xs] * PDF[ηdist, eitaVal[[x]] - ηmin]]);
gl[l_, xr_, xs_] := (If[ηhatPos > 1,
    pnts = Table[h * fl[l, i, xr, xs], {i, 1, ηhatPos, 1}];
    pnts = ReplacePart[pnts, h/2 * fl[l, 1, xr, xs], 1];
    pnts = ReplacePart[pnts, h/2 * fl[l, ηhatPos, xr, xs], ηhatPos];
    area = Apply[Plus, pnts],
    area = 0;
   ];
  area
 )
```

Procedure for computing $\tilde{\beta}$(l) : Mass of Cartels of Duration l

```
procDuration[yr_, ys_] :=
 (
betal = Table[gl[i, yr, ys], {i, 1, 1000}];
  fpl = betal / cartelMass;
  Table[i, {i, 1, 1000}].fpl (* average duration of cartel *)
 )
```

Functions and Procedure for computing durations for convicted cartels

Functions for computing convicted cartels of duration l

```
G1[l_, x_, xr_, xs_] :=
  (
   If[eitaVal[[x]] == ηmin,
    0,
    (1 - CDF[pftdist, phistar[[x]] - pftmin]) *
     bl[l, x, xr, xs] * PDF[ηdist, eitaVal[[x]] - ηmin]
   ]
  );

G2[l_, x_, xr_, xs_] :=
  (
   If[eitaVal[[x]] == ηmin,
    0,
    q * xr * bl[l, x, xr, xs] * PDF[ηdist, eitaVal[[x]] - ηmin]
   ]
  );

G3[l_, x_, xr_, xs_] :=
  (
   If[eitaVal[[x]] == ηmin,
    0,
    q * xr * CDF[pftdist, phistar[[x]] - pftmin] *
     bl[l, x, xr, xs] * PDF[ηdist, eitaVal[[x]] - ηmin]
   ]
  );

Lhatl[l_, xr_, xs_] :=
  (
   If[ηhatPos > 0,
    If[q * xr * xs ≤ θ,
      area = 0,
      pnts = Table[h * G1[l, i, xr, xs], {i, 1, ηhatPos, 1}];
      pnts = ReplacePart[pnts, h/2 * G1[l, 1, xr, xs], 1];
      pnts = ReplacePart[pnts, h/2 * G1[l, ηhatPos, xr, xs], ηhatPos];
      area = Apply[Plus, pnts]
     ];
```

```
    ,
    Print["No Cartels Ever: η̂=η"];
    area = 0;
   ];
   area
  );

Rhatl[l_, xr_, xs_] :=
  (
   If[ηhatPos > 0,
    If[q * xr * xs ≤ θ,
     pnts = Table[h * G2[l, i, xr, xs], {i, 1, ηhatPos, 1}];
     pnts = ReplacePart[pnts, h/2 * G2[l, 1, xr, xs], 1];
     pnts = ReplacePart[pnts, h/2 * G2[l, ηhatPos, xr, xs], ηhatPos];
     ,
     pnts = Table[h * G3[l, i, xr, xs], {i, 1, ηhatPos, 1}];
     pnts = ReplacePart[pnts, h/2 * G3[l, 1, xr, xs], 1];
     pnts = ReplacePart[pnts, h/2 * G3[l, ηhatPos, xr, xs], ηhatPos];
    ];
    area = Apply[Plus, pnts];
    ,
    Print["No Cartels Ever: η̂=η"];
    area = 0;
   ];
   area
  );
```

Procedure for computing the average duration of cartels convicted via leniency program

```
procCvtDurL[yr_, ys_] :=
  (
   massL = Table[Lhatl[i, yr, ys], {i, 1, 1000}];
   Lagg = Apply[Plus, massL];
   Print[Lagg];
   If[Lagg > 0,
    fL = massL / Lagg,
    fL = massL
    ];
   Table[i, {i, 1, 1000}].fL (* average duration of cartel *)
   );
```

Procedure for computing the average duration of cartels convicted via non-leniency program

```
procCvtDurR[yr_, ys_] :=
  (
   massR = ys * Table[Rhatl[i, yr, ys], {i, 1, 1000}];
   Ragg = Apply[Plus, massR];
   Print[Ragg];
   If[Ragg > 0,
    fR = massR / Ragg,
    fR = massR
    ];
   Table[i, {i, 1, 1000}].fR (* average duration of cartel *)
   );
```

Procedure for locating the fixed point for s

```
procFix[zr_, a_, b_] :=
 (
   sFlr = a;  (* initial floor value for "s" *)
   sCeil = b;  (* initial ceiling value for "s" *)
   dataFlr = procPLR[zr, sFlr];
   dataCeil = procPLR[zr, sCeil];
   pFlr = dataFlr[[1]];  (* p(L,R) for the floor value of s *)
   pCeil = dataCeil[[1]];  (* p(L,R) for the ceiling value of s *)
   If[Sign[sFlr - pFlr] == 0,
    sMid = 0;
    pMid = 0;
    LMid = dataFlr[[2]];
    RMid = dataFlr[[3]];
    AAMid = dataFlr[[4]];
    ,
    If[Sign[sCeil - pCeil] == 0,
      sMid = 1;
      pMid = 1;
      LMid = dataCeil[[2]];
      RMid = dataCeil[[3]];
      AAMid = dataCeil[[4]];
      ,
      If[Sign[sFlr - pFlr] != Sign[sCeil - pCeil],  (* if the p values for floor and
        ceiling values of s are on the opposite sides of the diagonal, then bisect *)
       sMid = N[(sFlr + sCeil)/2];
       dataMid = procPLR[zr, sMid];
       pMid = dataMid[[1]];
       LMid = dataMid[[2]];  (* L value at the mid-point value of s *)
       RMid = dataMid[[3]];  (* R value at the mid-point value of s *)
       AAMid = dataMid[[4]]; (* AA objective *)
       ,
       Print["No Fixed Point Suspected!"];
      ];
```

```
      While[Abs[sMid - pMid] > 0.001,
       (* bisection continues until the solution is within the precision range *)
       If[Sign[sFlr - pFlr] != Sign[sMid - pMid],
        sCeil = sMid;
        pCeil = pMid;
        ,
        sFlr = sMid;
        pFlr = pMid;
       ];
       sMid = N[(sFlr + sCeil)/2];
       dataMid = procPLR[zr, sMid];
       pMid = dataMid[[1]];
       LMid = dataMid[[2]];
       RMid = dataMid[[3]];
       AAMid = dataMid[[4]];
      ]; (* close the while-loop *)
     ]; (* close the if-floor-is-the-fixed-point-loop *)
   ]; (* close the if-ceiling-is-the-fixed-point-loop *)
   cms = procCMS[zr, sMid];
   {zr, sMid, pMid, q * zr * sMid, LMid, RMid, AAMid, cms,
    procDuration[zr, sMid], procCvtDurL[zr, sMid], procCvtDurR[zr, sMid]}
   (* report the values of {r, ŝ, p(ŝ), σ̂, L, R, AA} at the fixed point *)
  );
```

Baseline Simulation: γ = .5 and λ = 1.0

Main Procedure A (θ=0: Full Leniency)

Specify Parameter Values

```
α = 0; (* degree of competitiveness *)
f = 4; (* number of firms -- this is equivalent to "n" in the model *)
θ = 0;
(* leniency parameter: discount on penalty for the single recipient of leniency *)
ω = N[(f - 1 + θ)/f];
(* leniency parameter: discount on penalty for all recipients when all firms report *)
κ = 0.05;  (* opportunity rate to cartelize *)
δ = 0.85;  (* discount factor *)
γ = 0.5; (* damage multiple *)
β = 0; (* fixed penalty -- not part of the current model *)
```

```
(* parameters for p(L,R) = τ/(ξ+ν(λL+R)^ρ) *)
τ = 1.0;
ξ = 1;
ν = 1000;
λ = 1.0;
ρ = 1.4;
```

Locate the fixed point for s via bisection method

```
q = 0.2; (* probability that the cartel is reported to CA *)
sol = {};
Do[sol = Append[sol, procFix[r, 0, 1]]; Print[sol];, {r, 0, 1, 0.1}];
(* r: fraction of reported cases that the CA chooses to pursue *)

TableForm[sol, TableHeadings → {None, {"r", "s*", "p[s*]", "σ*",
    "L*", "R*", "AA*", "1-β̃(0)", "duration", "l-duration", "r-duration"}}]
```

r	s*	p[s*]	σ*	L*	R*	AA*	$1-\tilde{\beta}(0)$	duration	l-duration	r-duration
0.	1	1	0.	0	0.	0.	0.138472	127.378	0	0.
0.1	0.699219	0.699725	0.0139844	0.00191782	0.00201505	0.00332678	0.10267	39.4485	24.342	39.736
0.2	0.621094	0.620929	0.0248438	0.00172653	0.00333237	0.00379625	0.0850359	25.9621	17.9302	26.1286
0.3	0.574219	0.573476	0.0344531	0.00155632	0.00426881	0.00400755	0.0727032	20.1298	14.8274	20.2458
0.4	0.537109	0.537387	0.0429688	0.00145764	0.0050088	0.00414791	0.0640677	16.7971	12.779	16.8907
0.5	0.513672	0.513897	0.0513672	0.00136569	0.00555095	0.00421705	0.0568752	14.4634	11.2849	14.5416
0.6	0.5	0.499952	0.06	0.00128011	0.00591775	0.00423898	0.0505947	12.6672	10.0899	12.7341
0.7	0.503906	0.503258	0.0705469	0.0011494	0.00598077	0.00416315	0.0438692	11.0433	9.07209	11.0964
0.8	0.515625	0.515266	0.0825	0.00103181	0.00585781	0.00405224	0.0376431	9.64512	8.12274	9.68802
0.9	0.523438	0.524125	0.0942188	0.000970303	0.00574684	0.00397841	0.0328972	8.55142	7.27262	8.59029
1.	0.566406	0.56642	0.113281	0.000837129	0.00510902	0.00373091	0.0263822	7.25692	6.32376	7.2875

```
(* 1-β̃(0) is the rate of cartel for a given "r" *)
sol = Flatten[sol];
```

Export Data (Save onto hard drive)

```
stmp = OpenWrite["C:\CartelData\Sept12\LN\BASE\datC0"];
Write[stmp, sol];
Close[stmp];
```

Main Procedure B (θ=1: No Leniency)

Specify Parameter Values

```
α = 0; (* degree of competitiveness *)
f = 4; (* number of firms -- this is equivalent to "n" in the model *)
θ = 1;
(* leniency parameter: discount on penalty for the single recipient of leniency *)
ω = N[(f - 1 + θ)/f];
(* leniency parameter: discount on penalty for all recipients when all firms report *)
κ = 0.05;  (* opportunity rate to cartelize *)
δ = 0.85;  (* discount factor *)
γ = 0.5; (* damage multiple *)
β = 0; (* fixed penalty -- not part of the current model *)

(* parameters for p(L,R) = τ/(ξ+ν(λL+R)^ρ) *)
τ = 1.0;
ξ = 1;
ν = 1000;
λ = 1.0;
ρ = 1.4;
```

Locate the fixed point for s via bisection method

```
q = 0.2; (* probability that the cartel is reported to CA *)
sol = {};
Do[sol = Append[sol, procFix[r, 0, 1]]; Print[sol];, {r, 0, 1, 0.1}];
(* r: fraction of reported cases that the CA chooses to pursue *)

TableForm[sol, TableHeadings → {None, {"r", "s*", "p[s*]", "σ*",
    "L*", "R*", "AA*", "1-β̃(0)", "duration", "l-duration", "r-duration"}}]
```

r	s*	p[s*]	σ*	L*	R*	AA*	1-$\tilde{\beta}$(0)	duration	l-duration	r-duration
0.	1	1	0.	0	0.	0.	0.138472	127.378	0	0.
0.1	0.857422	0.857681	0.0171484	0	0.00199506	0.00171061	0.099753	33.8334	0	33.8334
0.2	0.755859	0.756207	0.0302344	0	0.00320616	0.00242341	0.0801541	22.1426	0	22.1426
0.3	0.691406	0.690502	0.0414844	0	0.00405704	0.00280507	0.0676174	17.1783	0	17.1783
0.4	0.648438	0.647697	0.051875	0	0.0046585	0.00302075	0.0582313	14.2922	0	14.2922
0.5	0.617188	0.616834	0.0617188	0	0.00512203	0.00316125	0.0512203	12.3088	0	12.3088
0.6	0.601563	0.602177	0.0721875	0	0.00535239	0.0032198	0.0446032	10.7813	0	10.7813
0.7	0.59375	0.593701	0.083125	0	0.00548888	0.00325902	0.0392063	9.50952	0	9.50952
0.8	0.601563	0.601021	0.09625	0	0.00537085	0.0032309	0.0335678	8.3622	0	8.3622
0.9	0.617188	0.617143	0.111094	0	0.00511725	0.0031583	0.0284292	7.35743	0	7.35743
1.	0.640625	0.641461	0.128125	0	0.00474997	0.00304295	0.0237498	6.46256	0	6.46256

```
(* 1-β̃(0) is the rate of cartel for a given "r" *)

sol = Flatten[sol];
```

Export Data

```
stmp = OpenWrite["C:\CartelData\Sept12\LN\BASE\datC1"];
Write[stmp, sol];
Close[stmp];
```