Environmental Quality: Impact of Economic Growth

Sangeeta Bansal

Centre for International Trade and Development
Jawaharlal Nehru University
New Delhi 110067, India

and
University of California, Berkeley, USA
sangeeta.bansal7@gmail.com

Abstract

Using a vertically differentiated product model, the paper aims to investigate the effects of economic growth on market provision of product quality. The quality attribute considered is environmental-friendliness of products. Economic growth is modelled as a shift in income distribution. It shows that the effect of economic growth depends on the form it takes. A growth in income that is uniform across consumers improves the cleanup levels adopted by both firms. However, a growth in income that is accompanied by changes in income inequality may result in lowering of one of the two qualities. More specifically, if the growth in income is accompanied by increased disparities in income distribution, the quality of the (environmentally) inferior variant is reduced. This has serious implications for the poor consumers if the product has safety or health hazards. The paper suggests a regulatory measure to prevent such deterioration in the quality of the inferior variant.

JEL Classification: O1, O44, L11, L 13, L15.

Keywords: Economic growth; disparities in income; environmental quality; standards; vertical differentiation.

Supplementary Material as Online Appendix

R Code to solve for equilibrium number of firms, qualities, and prices in the three stage game described on page 22.

```
#Define the cumulative distribution function for the uniform
distribution to be used as the income distribution function
cdf.uniform=function(x,a,b)
{
  if (x<a)
  {
    return (0)
  }
  else if (x>b)
  {
    return (1)
  }
  return((x-a)/(b-a))
}

#Define the NashEquilibrium Computation Function

#Input:
####N: The number of firms
####income.distr: The Cumulatilve distribution function of the
comsumer income distribution
####cost.func: The Cost function, relating product quality to
cost of production

#Procedure:
##Numerically solves for nash equilibrium prices, profits and
qualities in three stages
##Given the input cost function, income distribution, number
of firms and a fixed utility function

#Output: Returns a list with 3 members:
##s.opt: The N length vector of Nash Equilibrium qualities
##p.opt: The N length vector of Nash Equilibrium prices
##pi.opt: The N length vector of nash equilibrium profits

NashEquilibrium3Stage=function(
  N=3,
  income.distr=function(x){cdf.uniform(x,10,40)},
  cost.func=function(s,q,c=0.5){q*(s^2)/2*c}
)
{
  t=function(s,p)
  {
    t=c()
    t[[1]]=NA
    for (i in 2:length(s))
    {
      t[[i]]=(p[[i]]-p[[i-1]])/(s[[i]]-s[[i-1]])
    }
```

```
    return(t)
  }

  prof=function(s,p)
  {
    t=t(s,p);
    profit=c()

    q=function(k)
    {
      if (k==1)
      {
        income.distr(t[[2]])
      }
      else if (k==N)
      {
        1-income.distr(t[[N]])
      }
      else
      {
        income.distr(t[[k+1]])-income.distr(t[[k]])
      }
    }

    for(k in 1:N)
    {
      profit[[k]]=p[[k]]*q(k)-cost.func(s[[k]],q(k))
    }
    return(profit)
  }
  #Takes in a vector, applies BR.func to each index, iterates
till convergence of entire vector
  NEQ=function(vec, BR.func, iter.max=1000, term.incr=0.01)
  {
    prev.inc=term.incr+1;
    iter=0;
    vec.old=vec;
    vec.new=vec;
    while(iter<iter.max & prev.inc>term.incr)
    {
      for (i in 1:length(vec.old))
      {
        if(i==1)
        {
          min.val=0
        }
        else
        {
          min.val=vec.new[[i-1]]
        }
        vec.new[[i]]=BR.func(vec.new,i,min.val+0.0001)
      }
      prev.inc=max(abs(vec.new-vec.old))
      vec.old=vec.new
```

```
      iter=iter+1
    }
    return(list("opt.vec"=vec.new, "conv"=prev.inc));
  }
  #Takes in a vector of prices, and a vector of qualities
  #Given all qualities, and given p[-i], returns the p[i] that
optimizes pi[i]
  BR.price=function(p,i,s,min.val)
  {
    optim(
      par=p[[i]],
      fn=function(x)
      {
        p[[i]]=x;
        -1*prof(s,p)[[i]]
      },
      method="L-BFGS-B",
      lower=min.val,
      control=list(maxit= 100)
    )$par
  }

  #Given s[-i], returns the optimal s[i] that maximizes
prof[i]
  #Assuming prices are NEQ
  BR.qual.NE.price=function(s,i,min.val)
  {
    p0= (2*(0:(N-1)))+1;
    optim(par=s[[i]],
          fn=function(x){
            s[[i]]=x
            -1*prof(s,

NEQ(p0,function(p,i,min.val){BR.price(p,i,s,min.val)})$opt.vec
            )[[i]]
          },
          method="L-BFGS-B",
          lower=min.val,
          control=list(maxit= 100)
    )$par
  }

  s0=(2*(0:(N-1)))+1
  s.opt=NEQ(s0, BR.qual.NE.price);
  p0= (2*(0:(N-1)))+1;
  p.opt=NEQ(p0,
function(p,i,min.val){BR.price(p,i,s.opt$opt.vec,min.val)});
  pi.opt=prof(s.opt$opt.vec, p.opt$opt.vec);
  return(
    list(
      "s.opt"=s.opt,
      "p.opt"=p.opt,
      "pi.opt"=pi.opt
    )
```

```
  );
}

##Loop the NashEquilibrium function over multiple income
distributions
##Output the nash equilibrium prices, qualities, profits and
maximum
##number of firms for each income distribution

##The income distributions we consider are all unifrom
distributions
##with the lower limit (a) remaining constant at 20, and the
upper
##limit (b) increasing from 30 to 100 in increments of 10

get.max.N=function(b)
{
  prev.pi.opt=1;
  res.ls=list();
  print(paste("Now predicting for ", b))
  N=2;
  while(N<=10)
  {
    res.ls[[N]]=NashEquilibrium3Stage(N,
income.distr=function(x){cdf.uniform(x,20,b)});
    prev.pi.opt=res.ls[[N]]$pi.opt
    print(paste(N,":",prev.pi.opt))

    if(min(prev.pi.opt)<1e-6)
      break;

    N=N+1;
  }

  return(list("res.ls"=res.ls, "max.N"=N-1, "a"=20,"b"=b));
}
b.ls=seq(30,100,10);

master.res.ls=lapply(b.ls, get.max.N)
save(master.res.ls, file="FullRawResults.RData")

##Reformat the results obtained in the previous step into a
table and output to file
ds=matrix(NA,length(b.ls), 4)
colnames(ds)=c("b","firm.1","firm.2","firm.3")

for(i in 1:length(b.ls))
{
  ds[i,1]=b.ls[[i]]
  for(j in 1:3)
  {

curr.res.ls=master.res.ls[[i]]$res.ls[[master.res.ls[[i]]$max.
N]]$s.opt$opt.vec
```

```
    if(length(curr.res.ls)<j)
      curr=NA
    else
      curr=curr.res.ls[[j]]
    ds[i,j+1]=curr
  }
}

write.csv(ds, "NashEquilibriumQualities.csv")
```