岡山大学経済学会雑誌22(3·4),1991, 145～151

Negligibly Small Numbers and Quasi-Equilibria

Takao Fujimoto

1 . Scarf (1960) showed examples of globally unstable equilibrium, various efforts have been made to devise out models which enjoy some sort of stability. Mukherji (1990) proposed a new adjustment equation through which an equilibrium is approached, and which is not totally unrelated with economic reality. In Fujimoto (1990), a still simpler mechanism is presented in the framework of tatonnement process. This mechanism has much to do with 'conservatism' in its broadest (not necessarily best) sense. The reader is referred to the bibliography in Mukherji (1990) for other contributions on different strands.

In the first simple model by Scarf, prices never get nearer to the equilibrium, but keep a constant 'distance' away from it. A periodic solution to a system of differential equations. When we adopt instead the system of *difference* equations, prices exhibit a diverging path, the wilder the greater is the speed of adjustment. Then, we should ask, as proposed in Mukherji (1990), where prices go in the limit. This is because prices are to remain in the non-negative orthant, and in a modestly bounded region. By this it is meant that prices are normalized once they jump over a certain level. A solution path is now contained in a compact set: whether it can show a perpetual oscillation seems a tough problem.

So, I make a hasty (hopefully not rash) resort to a personal computer to handle this problem. This note is to report the result.

2 . Let us consider the first simple model in Scarf (1960). There are three consumers in our exchange economy. Three types of commodities are existent. Before markets are opened, consumer i has one unit of commodity i respectively, where $i = 1, 2,$ and 3. Consumer i has a utility function.

$$u_i = min\ (x_{i,i}, x_{i+1,i}),$$

The subscript $i+1$ should be understood in terms of modulus when it goes beyond 3. By the usual procedure, one obtains the following excess demand functions.

$$e_1(p_1, p_2, p_3) = p_3 / (p_3 + p_1) - p_2 / (p_1 + p_2),$$
$$e_2(p_1, p_2, p_3) = p_1 / (p_1 + p_2) - p_3 / (p_2 + p_3),$$
$$e_3(p_1, p_2, p_3) = p_2 / (p_2 + p_3) - p_1 / (p_3 + p_1).$$

Here p_i is the price of commodity i.

As is seen immediately,the equilibrium price vector is $k(1, 1, 1)$ with k being a positive scalar. When we employ the ordinary version of adjustment equation

$$dp_i / dt = e_i(p_1, p_2, p_3) \quad for \quad i = 1, 2, and\ 3,$$

it is not difficult to verify Σp_i^2 = constant on a solution path,using the Walras Law. Moreover, Scarf showed that $p_1\, p_2\, p_3$ is also constant on an orbit. These two properties are enough to prove global instability.

Now we shift to the adjustment system comprising difference equations:

$$p_i(t+1) = max\ (0, p_i(t) + a \cdot e_i(p_1, p_2, p_3)\ for\ i = 1, 2, and\ 3,$$

where a denotes the speed of adjustment assumed to be common among the markets. The above system considers the non-negative constraint, while setting no upper bound on prices. When we execute a program which describes the system, it terminates by zero division because of finite memory of a computer. If one examines the situation at the termination, it will be found that the economy is certainly not in the equilibrium. For example, when the initial price vector is (2.2, 2.2, 1), the speed of adjustment $a = 1$, and the relevant variables have six significant digits,then after 50 iterations the process halts: the final state is $p = (5.1, 0, 0)$. As for the last but one state, we have $p = (5.1, 0, 0.30)$ with the excess demand $(0.056, 0, -0.94)$.

Can this penultimate state be called an equilibrium ? The excess demand for commodity 1 may be negligible, while p_3 seems negligible though e_3 is negative. Surely, whether we can neglect some small numbers depends on the choice of units of measurement. Let us suppose a suitable choice has been done, and numbers are after all small, i. e., negligible.

3 . It should be better understood if one thinks of markets where minimum prices are specified below which any transaction is prohibited. We regard such regulatory lower bounds also as 'negligibly small numbers'. Now denote by *NS* a negligibly small number, and let us set out the rules of adjustment. (We assume *NS* is the same among markets for simplicity.)

(1) If $p_i\ (t + 1)$ becomes less than or equal to *NS* while $p_i\ (t)$ is greater than *NS*, $p_i\ (t + 1)$ is readjusted to be *NS*.

(2) If both $p_i\ (t + 1)$ and $p_i\ (t)$ are not more than *NS*, it is set to zero, i.e., declared 'free'.

(3) If any one of prices becomes greater than *MAX* (a positive

number), normalization of prices is performed so that the maximum price among three is readjusted to be MAX.

One more concept of quasi-equilibrium. A state is quasi-equilibrium if the following conditions are met.

(i) Each excess demand is not more than NS.

(ii) If an excess demand is negative, the corresponding price should not be more than NS.

A computer program realizing the above adjustment process is given in Appendix. In the program, $NS = 0.01$ and $MAX = 3$, and whenever a price hits into the region of small numbers, the program examines whether the current state is quasi-equilibrium or not, and if the answer is in the affirmative, it halts.

There are three quasi-equilibria: $(3, 0, NS)$, $(NS, 3, 0)$, and $(0, NS, 3)$. (We identify price vectors on a ray from the origin.) Surprisingly or not, given any (?) initial point the process can reach a quasi-equilibrium. The greater is the speed of adjustment, the quicker it ends up happily. The system is not structually stable,though. A slight change in the speed of adjustment brings the economy to a totally different quasi-equilibrium.

4 . Here are some technical remarks. We could assume different small numbers for prices and excess demands separately. The result, however, remains essentially the same. The program is written in Turbo PASCAL version 3.01, and can run in the integrated environment of version 4 or 5 also with a slight modification. In the later versions, one can add some graphics routines so that it is possible to visualize the process colourfully. The author has prepared those programs with graphics both in Turbo PASCAL and Turbo BASIC, which are available on request.

5 . Now a final remark about two awkward points in our quasi-equilibrium. Suppose we are in the quasi-equilibrium (3, 0, *NS*). (1) The second consumer, with endowment declared to be free, donates all his commodity to the first consumer ! (2) Since the excess demand for commodity 1 is positive (though negligible), some degree of altruism is also required here.

Together with the author's note (1990), we can say 'conservatism' enhances stability and brings forth a more equitable outcome. Maybe, too much.

References

(1) T. Fujimoto, "Stabilizing Scarf's Unstable Examples," Okayama Economic Review, Vol. 22 (1990) pp. 67–71.

(2) A. Mukherji, "On Tatonnement Process," mimeo at the Institute of Socio-Economic Planning, Tsukuba University (1990).

(3) H. Scarf, "Some Examples of Global Instability of Competitive Equilibrium," International Economic Review,Vol. 1 (1960), pp. 157–172.

Appendix

```
program NEGS;    {*** for TP on '90-07-14(Sat.):rev.7-17(Tue.):TF ***}
(*********************************************************************)
(*** An Example of Globally Unstable Competitive Equilibrium      ***)
(*** due to H.Scarf(I.E.R.,Vol.1,1960): PGM="ScarfS"              ***)
(*********************************************************************)

label
     CY;
const
     NS:real=0.01;    (* a negligibly small number           *)
var
     A              : real;          (* speed of adjustment *)
     P1T1,P2T1,P3T1: real;           (* past prices         *)
     P1T0,P2T0,P3T0: real;           (* current prices      *)
     vexc1,vexc2,vexc3: real;        (* values of Excess D. *)
     CZ1,CZ2,CZ3   : integer;        (* flags               *)
     IT   : integer;                 (* iteration count     *)

function EXC1(P1,P2,P3:real):real;   (***  Excess Demand Functions  ***
begin
 EXC1:= ((P3)/(P3+P1))-P2/(P1+P2)    (* EXC1 *)
end;

function EXC2(P1,P2,P3:real):real;
begin
 EXC2:= ((P1)/(P1+P2))-P3/(P2+P3)    (* EXC2 *)
end;

function EXC3(P1,P2,P3:real):real;
begin
 EXC3:= ((P2)/(P2+P3))-P1/(P3+P1)    (* EXC3 *)
end;

procedure CHECKQE;(* check whether the state is in Quasi-Equilibrium *)
begin
writeln(#7);                       (* a bell sound when examining     *)
if (vexc1>NS) or (vexc2>NS) or (vexc3>NS) then exit;(* No. *)
 if (vexc1<0) and (P1T0>NS) then exit;
 if (vexc2<0) and (P2T0>NS) then exit;
 if (vexc3<0) and (P3T0>NS) then exit;
  begin
  writeln;
  writeln(' *** Yes,in Quasi-Equilibrium !  ');
   writeln(' ... value of a small number = ',NS:7:5);
   writeln(' *** the speed of adjustment = ',A:7:5);
  halt;
  end;
end;

procedure DISPE;(* display the values of Excess Demand functions *)
begin
 writeln('    P1 =',P1T0,';    Exc1 =',vexc1);
 writeln('    P2 =',P2T0,';    Exc1 =',vexc2);
 writeln('    P3 =',P3T0,';    Exc1 =',vexc3);
```

```
end;

begin (******* MAIN *******)

clrscr;
writeln('*** Press Shift key to have a temporary stop. To restart ***');
writeln('*** press any other key. If you hit "q",then quitting.   ***');
write('*** Key in the speed of adjustment.(Default A=1.0). A = ');
A:=0.0;
Readln(A);
if A=0.0 then A:=1.0;
(* ---------------------------------------------------------------   *)
(*                                                                   *)
(*               Initial Values                                      *)
(*                                                                   *)
   P1T1:=2.2; P2T1:=2.2; P3T1:=1;
(* ---------------------------------------------------------------   *)

   IT:=0;                         (*==  Iteration count ==*)

CY:                               (* label for looping cycle *)
   IT:=IT+1;

   P1T0:= P1T1 + A*EXC1(P1T1,P2T1,P3T1);          (* adjusting prices *)
   P2T0:= P2T1 + A*EXC2(P1T1,P2T1,P3T1);
   P3T0:= P3T1 + A*EXC3(P1T1,P2T1,P3T1);

                                                  (* normalization    *)
   if (P1T0>3) or (P2T0>3) or (P3T0>0) then begin
    if P1T0>3 then begin P1T0:=3; P2T0:=P2T0*3/P1T0; P3T0:=P3T0*3/P1T0 end;
    if P2T0>3 then begin P2T0:=3; P1T0:=P1T0*3/P2T0; P3T0:=P3T0*3/P2T0 end;
    if P3T0>3 then begin P3T0:=3; P1T0:=P1T0*3/P3T0; P2T0:=P2T0*3/P3T0 end;
   end;

   if (P1T0<=NS) then P1T0:=NS;
    if (P1T0<=NS) and (P1T1<=NS) then P1T0:=0;
   if (P1T0<=NS) and (P1T1>NS)  then CZ1:=1 else CZ1:=0;(* if newly entered
  if (P2T0<=NS) then P2T0:=NS;
   if (P2T0<=NS) and (P2T1<=NS) then P2T0:=0;
   if (P2T0<=NS) and (P2T1>NS)  then CZ2:=1 else CZ2:=0;
  if (P3T0<=NS) then P3T0:=NS;
   if (P3T0<=NS) and (P3T1<=NS) then P3T0:=0;
   if (P3T0<=NS) and (P3T1>NS)  then CZ3:=1 else CZ3:=0;

 vexc1:=EXC1(P1T0,P2T0,P3T0);
 vexc2:=EXC2(P1T0,P2T0,P3T0);
 vexc3:=EXC3(P1T0,P2T0,P3T0);

  writeln(IT);
  DISPE;

  if (CZ1=1)  or (CZ2=1) or (CZ3=1) then CHECKQE;(* if flaga are ON... *)

  P1T1:=P1T0; P2T1:=P2T0; P3T1:=P3T0;          (* prep for the next round *

  GOTO CY;                                     (* back to the iteration   *
end.
```