Appendix E: How Mutation Works in the Simulation

On initialization, all evolutionary variables controlling an agent's behaviour are randomly generated. These variables are inherited by new agents unless the code is 'mutated'. 'Mutation' is possible each time a new agent is 'born'.

For example, the agent variable *mutation* determines the rates of mutation of all the evolutionary variables including the *mutation* variable. Each agent generated at the beginning of the simulation has *mutation* initially set through this code:

```
set mutation random 200
```

Experimentation suggested the exact code simulating mutation made little difference in the behaviour of the simulation. The existing code is basically the last experimental code used. In the simulation v203, the mutation test for the variable *mutation* is:

```
if random 10000 - 1 < mutation [ set mutation mutation + random 10 - random 10
                                 if mutation < 0 [set mutation 0]]
```

All the agent's behaviour variables, except *mutation*, can be turned off completely through the process of mutation. Equally, all variables have no practical ceiling. The rates of mutation (e.g. + random 10 - random 10) were tested through repeated experimentation.

There are no claims that the simulation's mutation code accurately reflects the reality of human beings. Experimentation indicates that changing these rates makes little difference to the overall shapes exhibited by the simulations. Mutation is merely a utilitarian feature of this evolutionary model. Selection, rather than mutation, is the dominant evolutionary force during periods of change such as those featured by the relatively the short duration (in terms of evolution) of a non-renewable economy.

The full mutation code for the agent's variables is as follows:

```
      if random 10000 - 1 < mutation [
            set mutation mutation + random 10 - random 10
             if mutation < 0 [set mutation 0]]
       if random 10000 - 1 < mutation [
            set nurture-needs nurture-needs + (random-float 1 / 10) - (random-float 1 /
10)
             if nurture-needs < 0 [set nurture-needs 0]
             if nurture-needs > 1 [set nurture-needs 1]]
       if random 10000 - 1 < mutation [
            set vision vision + random-float 1 - random-float 1
             if vision < 0 [ set vision 0 ]]
       if random 1000 - 1 < mutation [
            set fertile fertile + random-float 2 - random-float 2
             if fertile < 0 [set fertile 0]]
       if random 10000 - 1 < mutation [
            set charity charity + random-float .1 - random-float .1
             if charity > 1 [set charity 1]
             if charity < 0 [set charity 0]]
       if random 10000 - 1 < mutation [
            set movement movement - random 2 + random 2]
            if movement < 0 [ set movement 0 ]
       if random 10000 - 1 < mutation [
            set movement-wants movement-wants - random 2 + random 2]
            if movement-wants < 0 [ set movement-wants 0 ]
       if random 10000 - 1 < mutation [
            set exploration exploration - random 10 + random 10 ]
            if exploration < 0 [ set exploration 0 ]
       if random 10000 - 1 < mutation [
            set savings-rate-needs savings-rate-needs + random-float .5 - random-float
.5
             if savings-rate-needs <= 0 [set savings-rate-needs 0]]
       if random 10000 - 1 < mutation and NBB_Evolve = true [
            set child-need-buffer child-need-buffer + random-float 2 - random-float 2
            if child-need-buffer < 0 [ set child-need-buffer 0 ]]
       if random 10000 - 1 < mutation [
            set money-value money-value + random-float .2 - random-float .2
            if money-value < 0 [ set money-value 0 ]]


if industrial-innovate > 0 [
       if random 10000 - 1 < mutation [
            set nurture-money nurture-money + (random-float 1 / 10) - (random-float 1 /
10)
             if nurture-money < 0 [set nurture-money 0]
             if nurture-money > 1 [set nurture-money 1]]
       if random 10000 - 1 < mutation [
            set price-up price-up + random-float .001 - random-float .001
             if price-up <= 0 [set price-up .00001]]
       if random 10000 - 1 < mutation [
            set price-down price-down + random-float .001 - random-float .001
             if price-down <= 0 [set price-down .00001]
             if price-down > .999999 [set price-down .99999 ]]
```

```
        ]

If industrial-innovate > 1 [

        if random 10000 - 1 < mutation [
             set nurture-wants nurture-wants + (random-float 1 / 10) - (random-float 1 /
10)
              if nurture-wants < 0 [set nurture-wants 0]
              if nurture-wants > 1 [set nurture-wants 1]]
        if random 10000 - 1 < mutation [
             set savings-wants savings-wants + random .5 - random .5
              if savings-wants < 0 [set savings-wants 0]]
        if random 10000 - 1 < mutation [
             set migration migration + random-float .1 - random .1
              if migration < 0 [set migration 0]
              if migration > 1 [set migration 1]]

        if random 10000 - 1 < mutation [
             set needs-stock needs-stock + random living-cost - random living-cost
              if needs-stock < 0 [set needs-stock 0]]              if random 10000 - 1 <
mutation [
             let propensity-capital 1 - propensity-labour - propensity-health
             set propensity-labour propensity-labour + random-float .3 - random-float .3
               if propensity-labour < 0 [ set propensity-labour 0 ]
             set propensity-health propensity-health + random-float .3 - random-float .3
               if propensity-health < 0 [ set propensity-health 0 ]
             set propensity-capital propensity-capital + random-float .3 - random-float .3
               if propensity-capital < 0 [ set propensity-capital 0 ]
             let all propensity-labour + propensity-health + propensity-capital
             set propensity-labour propensity-labour / all
             set propensity-health propensity-health / all

             set propensity-use-needs propensity-use-needs + random-float .10 -
random-float .10
               if propensity-use-needs < 0 [ set propensity-use-needs 0 ]
               if propensity-use-needs > 1 [set propensity-use-needs 1]]
                        ]

If industrial-innovate > 2 [
         if random 10000 - 1 < mutation
             [set order-of-trade random 12 ]]
       ]

         set needs needs - (needs * nurture-needs)
         set money money - (money * nurture-money)
         set wants wants - (wants * nurture-wants)
         set children children + 1

end
```