

# Simple Nelwin for Maple V.4

Version 4 Sep 96  
Esben Sloth Andersen, esa@business.auc.dk

## Preliminary notes

1. This program is a simplified version of the program found in Andersen's book on Evolutionary Economics (Pinter, 1994/1996). It allows a quick overview, but if the purpose is to make major extensions of the Nelson and Winter framework, then SimpleNelwin should be abandoned and a more structured and decomposed kind of programming should be applied.
2. A program of the same kind (with a single bug) has been made available at Andersen's Web site (<http://www.business.auc.dk/evolution/models/nelwin/Nelson-and-Winter-A.html>). The previous program was programmed in Maple V.2, and therefore it cannot be run with Maple V.4. The present program is rewritten for the new version of Maple. At the same time a couple of helping procedure are made available to ease the plot of the results of the simulations.
3. Apart from the simple translation, there are different further changes. The main one is that the normal distribution is called with  $\ln(\text{mean inno})$  and the result is transformed back by  $\exp(\text{result})$ . Maple V.4's generation of random numbers has the same statistical properties as Maple V.2's random generators, but it is not possible to reproduce exactly the same sequences of innovations and imitations as with Maple V.2.
4. The worksheet is run by placing the cursor in the first line of Maple input (> restart;), press <enter>, and then press <enter> each time you are in a new input section (execution group).
5. Comments, bug reports and suggestions for new features are very welcome. Please indicate which version of the note you are referring to (the present one is Version 96/08/24).

## The program

Before running the program, we reset Maple and secure that extensive run-time error messages are given (to ease debugging).

```
> restart; printlevel := 3;  
printlevel := 3
```

Then we are ready to enter the simple Nelson and Winter-like model. During a first overview of the present note, you should not study the procedure. Just press <enter>, and then proceed to the analysis of the results of the process of Schumpeterian competition. During a second study of the note, however, the program should be studied. Here it is helpful to press the Maple button that toggles the display of Maple input between ordinary text and standard notation (with formatting and indentation of the program). However, inserted notes in programs are not shown in the formatted standard notation - so you should also study the program in its ordinary text version.

The standard parameters of Nelwin have been changed in order to obtain significant effects relatively quickly (in terms of computer time).

You can change evolutionary simulation in three ways:

1. You can change the arguments of the procedure calls: `SimpleNelwin( $n,T$ )`, e.g. `SimpleNelwin(4,16)`.  $T$  is the number of periods of the simulation. Since Maple is rather slow, you should start with few periods - e.g. 16 or 25.  $n$  is the number of firms. The present parameters give a near-equilibrium without innovations for 4 firms. But you can start in an out-of-equilibrium situation with e.g. 16 firms.
2. You can change the system variable seed, and thus change the sequences of random numbers (innovations and imitations). See below.
3. You can change the parameters in the program text, e.g. `b := 2.5`;
4. You can change the program commands and expressions.

But here is the initial version:

```
> SimpleNelwin := proc(n,T)
```

```

# This model has a discrete searchspace
# (like N&W, chap. 9; unlike N&W, chs. 12-13).

local i,t,j;
global b, c, delta, dem, d_im, d_in, eta, phi, r_im, r_in,
sigma_in, search_sp, A, A_mean, K, s, Q, TQ, P, A_max,
lambda_in, Lottery, A_prelim, A_in, lambda_im, A_im, rho,
I_des, pi, loans, I_max, constraint, Inv;
# The fact that variables and constants are "global" means
that they can
# be inspected interactively after the simulation

option `ES Andersen, Aalborg University, 24 Aug 96`;

with(stats[random]);

# PARAMETERS
b      := 1      ;
c      := 0.16   ;
delta  := 0.03   ;
dem    := 67     ;
d_im   := 0.3    ;
d_in   := 0.3    ;
eta    := 1      ;
phi    := 0.05   ;
r_im   := 0.00112 ;
r_in   := 0.0223 ;
sigma_in := 0.2  ;
search_sp := [.050, .063, .111, .160, .206, .292,
              .297, .363, .375, .403, .426, .441, .558,
              .675, .718, .742, .843, .846, .917, .981];
# Such discrete search space is not found in N&W, 1982, chs.
12-14
# but in ch. 9. It is important for analysis at the level of
techniques

# MAIN VARIABLES
A := array(1..n, 1..T+1);
A_mean := array(0..T+1);
K := array(1..n, 1..T+1);
Q := array(1..n, 1..T);
s := array(1..n, 1..T);
pi := array(1..n, 1..T);

# INITIALISATION
for i from 1 to n do
    A[i,1] := 0.16;
    K[i,1] := 89.70;
od;
A_mean[0] := 0.16;

# MAIN PROGRAM
for t from 1 to T do

    # SHORT RUN
    for i from 1 to n do
        Q[i,t] := A[i,t]*K[i,t];
    od;

```

```

TQ := sum(Q[k,t], k = 1..n);
P := dem/TQ;

# NEWTECHNO
A_max := max(seq(A[i,t], i = 1..n));
A_mean[t] := A_mean[t-1]*(1 + phi);
for i from 1 to n do

    # Innovate
    lambda_in[i] := d_in*r_in*K[i,t];
    Lottery := poisson[lambda_in[i]]();
    if Lottery() > 0 then
        A_prelim := exp(normald[ln(A_mean[t]),sigma_in]());
        j := 1;
        while search_sp[j] <= A_prelim do j := j + 1; od;
        A_in[i] := search_sp[j-1];
    else A_in[i] := 0;
    fi;

    # Imitate
    lambda_im[i] := d_im*r_im*K[i,t];
    Lottery := poisson[lambda_im[i]]();
    if Lottery > 0 then A_im[i] := A_max;
    else A_im[i] := 0;
    fi;

    # Technochoice
    A[i,t+1] := max(A[i,t], A_in[i], A_im[i]);

od;

# NEWCAPITAL
for i from 1 to n do

    s[i,t] := Q[i,t]/TQ;
    rho[i] := c/(P*A[i,t+1]);
    I_des[i] := delta + 1 - eta/(eta - s[i,t])*rho[i];

    pi[i,t] := P*A[i,t] - (c + r_in + r_im);
    if pi[i,t] <= 0 then loans[i] := 0;
    else loans[i] := b*pi[i,t];
    fi;
    I_max[i] := delta + pi[i,t] + loans[i];

    constraint := min(I_des[i], I_max[i]);
    Inv := max(0, constraint);
    K[i,t+1] := K[i,t]*(Inv + 1 - delta);

od;

od;
print(cat(`SimpleNelwin was run with `,n,` firms for `,T,`
periods.`));
print(`Data are ready for inspection and plotting.`);
print(``);
end:

```

To ease the analysis of the results of evolutionary simulations, Andersen et al. (DRUIDIC) have made a series of helping procedures. Here are a couple for plotting of time series data.

### **Procedure that plots a time series for a firm-level variable**

This procedure draws time series data for the firms of the industry. In order to allow simple comparisons across time series for different variables, each firm is given a specific colour that is the same in all plots.

```
> DrawVariable := proc(var,N,T)
  global periods,plotdata,text;
  local colour,tt,t,n,a;
  option `ES Andersen, Aalborg University, 24 Aug 96`;

  periods := seq(t, t = 1..T);
  colour := seq('COLOUR'(HUE,n/N), n=1..N);
  a:=NULL;
  for n from 1 to N do
    a := a,
    'CURVES'([seq(MergeLists([periods[tt],[var[n,tt]]),tt=1..
T)],
    colour[n],THICKNESS(2));
  od;

  text := cat(`Variable `,var,` for `, N, ` firms and `, T,
` periods.`);
  plotdata := 'PLOT'(a, 'TITLE'(text), AXESSTYLE(NORMAL),
AXESTICKS(5,5));
  plotdata;

end;
```

### **Function that helps the creation of a PLOT data structure**

This procedure performs a subtask for the DrawVariable procedure, namely to create data points.

```
> MergeLists := proc(list1, list2)
  local LookUpItem,i;
  option `ES Andersen, Aalborg University, 24 Aug 96`;

  LookUpItem := proc(list1, list2, seriesnumber)
    local listnumber;
    if type(seriesnumber, even) then
      listnumber := seriesnumber/2;
      RETURN(list1[listnumber]);
    else
      listnumber := (seriesnumber-1)/2;
      RETURN(list2[listnumber]);
    fi;
  end;

  if not nops(list1) = nops(list2) then
    ERROR(`the number of elements in lists must be the
same`);
  fi;
  [seq(LookUpItem(list1, list2, i),
i = 2..nops(list1)*2 + 1)];
end;
```

### **Legend for plots**

```
> Legend := proc(N)
  global periods,plotdata,text;
  local colour,i,n,a,b;
```

```

option `ES Andersen, Aalborg University, 28 Aug 96`;

colour := seq('COLOUR'(HUE,n/N), n=1..N);
a:=NULL; b := NULL;
for n from 1 to N do
  a := a, 'CURVES'([[0,N-n],[10,N-n]],
    colour[n],THICKNESS(2));
  b := b, 'TEXT'([11,N-n],`Firm `n, FONT(TIMES,ROMAN,12));
od;

text := cat(`Legend for `,N,` firms`);
plotdata := 'PLOT'(a, b, 'TITLE'(text), AXESSTYLE(NONE));
plotdata;

end:

```

## ☐ Simulations and data presentation

### ☐ A close race of Schumpeterian competition (seed = 4)

[ Now we are ready for running the model, e.g. for 4 firms during 16 periods (quarters of a year in Nelson's and Winter's own models).

[ To make a experiment of the many patters that arise due to purely random events, we explicitly specify the "seed" of the random number generator. Such generators are deterministic although their output looks like random data. Therefore, a given experiment can be reproduced by specifying the seed initially given to the random number generator. We start with seed = 4, and consider the resulting story.

```

> _seed := 4; st := time(): SimpleNelwin(4,16);
  `Time used`,time() - st;

```

*\_seed := 4*

*SimpleNelwin was run with 4 firms for 16 periods.*

*Data are ready for inspection and plotting.*

*Time used, 22.000*

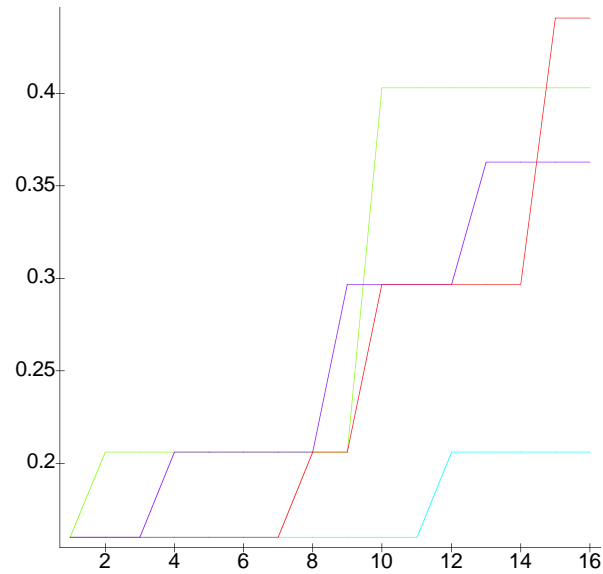
[ We start by inspecting the capital coefficients (the A's) defined as  $Q_{it} = A_{it} K_{it}$ . In the present close race, innovations are quickly followed by imitations (or compensating innovations) from the other firms of the industry. All start with 0.16, then the light green firm makes an innovation and the violet firm in quick to make a response (an imitation or an innovation). Later the red firm catch up, but the aquamarine firm is too weak. Trough a series of innovations and a few imitations the light green, violet and red firms have a close competition.

```

> DrawVariable(A,4,16);

```

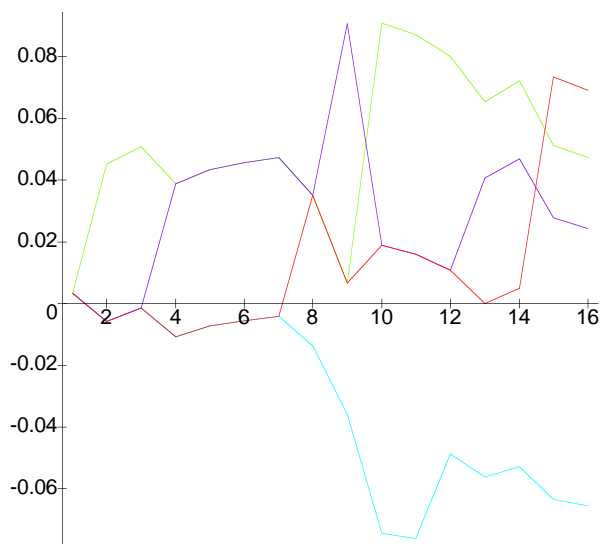
### Variable A for 4 firms and 16 periods.



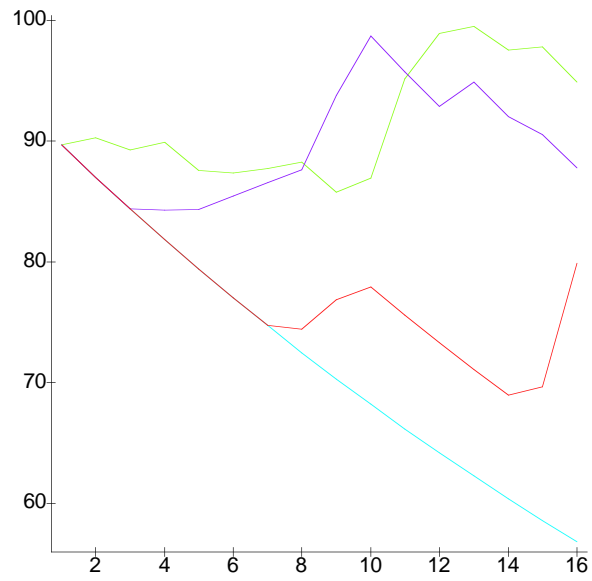
[Comment for readers of non-colour printouts: In the present version the individual firms can only be compared precisely across different plots if you have colours.]

A successful innovation or imitation increases the profitability ( $\pi_{i,t}$ ) of the firm. A profitable firm will expand its capital ( $K_{i,t}$ ). Initially the red and the aquamarine firm have negative profits but later it is one the aquavamarine that is in serious problems. This shows up as no investment and a gradual increase of capital. In the present version of the program, there is no bankruptcy mechanism - but this can easily be introduced.

> **DrawVariable(pi,4,16);DrawVariable(K,4,16);**  
Variable pi for 4 firms and 16 periods.

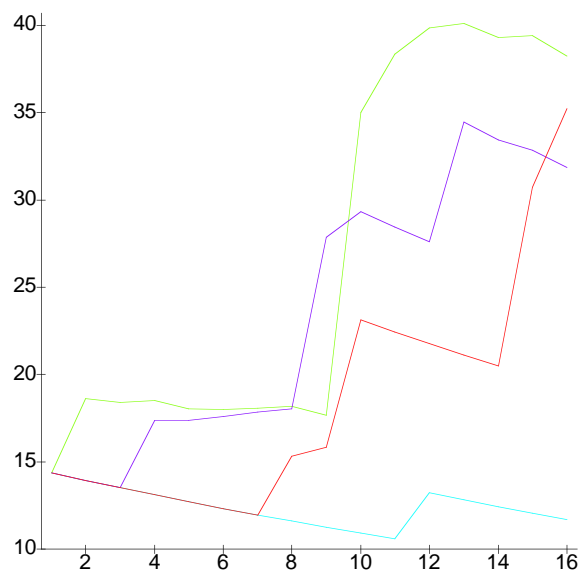


Variable K for 4 firms and 16 periods.

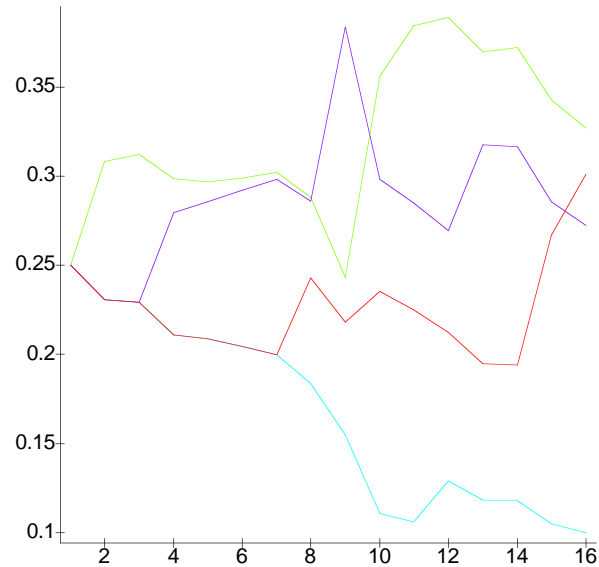


[ The innovations and imitations also influence output ( $Q_{i,t}$ ) and market share ( $s_{i,t}$ ).

> **DrawVariable(Q,4,16);DrawVariable(s,4,16);**  
Variable Q for 4 firms and 16 periods.



Variable s for 4 firms and 16 periods.



Many aspects of this story depends on random events. Thus can easily be seen by trying out another seed for the random generator.

### Further development of the note

Now follows a few parameter experiments, and then we come to changes in the model. [The student is supposed to chose one or two options].

- science based/cumulative technological development
- discrete/continuous search space
- bankruptcy
- different markup functions
- different specifications of final demand (e.g. logistic expansion of the market)
- Finally we introduce an extra evolving state variable - the R&D intensity like in Silverberg/Verspagen.

### Make your own experiments

```
[ > _seed := 123456789; SimpleNelwin(4,16);  
[ > DrawVariable(A,4,16);  
[ >
```