

From random walks to Arthur's competing technology models: A hands-on approach

Esben Sloth Andersen
Revision: 11 February 2001

1. Introduction

This note suggests a hands-on simulation approach to the study of models with increasing returns and network effects. As soon as we move away from verbal introductions such as Shapiro/Varian: *Information Rules* (1999), we find out that we are dealing with a highly formalised field. If we want a deeper understanding, we need to start with simple models like the ones presented in the present note. These models have analytical solutions, so why should we try simulate them? For two reasons: First, the simulation approach gives a deeper understanding and control over the formal models. Second, when we have implemented the simple models, it is easy to develop them further. Thereby, we may quickly move toward frontiers that can be understood even though we don't immediately go for (potentially non-existing) analytical solutions.

You don't have to use the hands-on approach since many simple examples are included in the present note. So you can just read it. However, although the note looks like a text document, it is actually a "worksheet" for the mathematical programming package Maple, which is available for many operation systems (Unix, Windows, Macintosh, etc.). Maple exists in different versions. The present worksheet is for Maple 6, but after small modifications it can be used for previous versions of Maple. If the worksheet is loaded into Maple, then all Maple input areas can be loaded into Maple's mathematical engine. Some Maple inputs are stored for later use (eg. program procedures and values of variables). Other inputs give an automatic response. If we after a Maple prompt (>) write `2+2;<enter>`, Maple answers `4`:

```
> 2+2;
```

```
4
```

If you don't want to use Maple (which is available as site licences at many universities), you can more or less easily reprogram the procedures for other programming systems – both the general ones like Pascal and C++ and the specific ones like Mathematica. Later the present worksheet will be used for programs in the programming system Lsd (Laboratory for Simulation Development) – available from <http://www.business.auc.dk/lzd/>.

2. Random walks and Polya processes

2.1. Stochastic processes

The random walk model is a convenient starting point that traces back to statistical mechanics with its Brownian movements of molecules, but we can just as well think of a drunkards walk with small and erratic steps in two directions. In the case of network economics we may think of the movement of users from an old system/technology to two new and alternative systems, A and B . Let us assume that new users come one by one and adopt each system with probability 0.5. We may also say we have two types of potential users:

- R -type users: $u(B) < u(A)$
- S -type users: $u(A) < u(B)$

Users choose the best, and we assume that the choice is irreversible – ie. that the user sticks to the chosen new system "forever". Given these assumptions, we have a stochastic process that generates a random sequence of the two systems. Here is such a random sequence of 30 choices:

A, B, A, A, B, B, A, B, A, A, A, A, A, B, B, A, B, A, A, A, B, B, B, A, A, A, B, B, A, B

Such a sequence of random choices can be transformed into a random walk by the studying the difference between the number of adoptions of A and B. The sequence of differences is:

1, 0, 1, 2, 1, 0, 1, 0, 1, 2, 3, 4, 5, 4, 3, 4, 3, 4, 5, 6, 5, 4, 3, 4, 5, 6, 5, 4, 5, 4

In period 30 we have a installation share of A equal to $\frac{15+4}{30}=.63$. What about the market share of

system A in the longer run? It is not difficult to see that due to the law of large numbers this market share will be very close to 0.5.

The crucial assumption underlying the random walk is that the existing market shares do not influence the adoption decision. This is not realistic. In a simple diffusion model of the systems, we may assume that each previous adopter helps to recruit new adopters. This may lead to a Polya process (referring to the mathematician George Polya, 1887-1985). Here the probability that a new adopter applies a particular system is proportional to its market share. We may interpret this in the following way: All potential users are homogeneous with respect to their lack of information of the utilities of the two systems. To obtain information they select by chance one existing user:

- If it is an A-user, then expectations of the new user are that $u(B) < u(A)$
- If it is an B-user, then expectations of the new user are that $u(A) < u(B)$

In this case the long-term behaviour of the market shares is totally different from the random walk case. Actually, *any* market share can represent a long-term equilibrium. The only thing we know (from Polya's proof) is that with probability 1, one of the infinite number of equilibria will be found for any particular process of adoption. The background is again the law of large numbers. In the beginning there are only a small number of adopters that try to recruit new ones. Therefore, the choices behave randomly. But these initial random events move the system towards more and more stable market shares.

Even the Polya process seems much too simplistic. For instance, it does not take into account the possibility of a "corner solution" in which one of the market shares reaches 1. This case can easily be obtained: Assume that that system A and B have different utility characteristics and that these characteristics are fully known to all adopters. In that case all adopters chose the best system.

2.2. Preparing for the simulations (plotting procedure - only relevant when running Maple)

In the next section we shall see how to implement the stochastic processes in the mathematical programming package Maple (version 6, but with small changes previous systems can be used). However, these programs just product a lot of data, so we need a tool for an easy visualisation of them. This is the following plotting procedure that uses a lot of Maple programming and plotting tricks. So, don't try to understand it. **If you are running the programe, you need to load the DataPlot procedure by placing the cursor in the field below and pressing <enter>.**

```
> DataPlot := proc(varlist, t, T, varmin, varmax, ymarks, colours, Type)
  local colourlist, colourseq, DATA, i, j, MergeLists, n;
  option `E.S. Andersen, 11 Feb 2001`;
  MergeLists := proc(list1, list2)
    local i;
    global LookUpItem;
    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])
        end if
    end proc;
    RETURN([seq(LookUpItem(list1, list2, i), i=2 .. nops(list1)*2+1)])
end proc;
colourlist := [[1.0, 0, 0], [0, 1.0, 0], [1.0, 1.0, 0], [.439, .859, .576], [0, 0, 0],
    [0, 0, 1.0], [.647, .165, .165], [.310, .184, .310], [.137, .137, .557], [1.0, .498, 0],
    [0, 1.0, 1.00], [.8, .498, .196], [.753, .753, .753], [.624, .624, .373], [1.0, 0, 1.00],
    [.557, .137, .420], [.8, .196, .196], [.737, .561, .561], [.918, .678, .918],
    [.557, .420, .137], [.859, .576, .439], [.678, .918, .918], [.847, .847, .749]];
colourseq := NULL;
DATA := table( );
DATA['Title'] := NULL;
if Type=dt then DATA['Title'] := `Difference between install bases of A and B`
else DATA['Title'] := `Market share for system A`
end if;
DATA['Plotdata'] := NULL;
if colours=black then for n to nops(varlist) do colourseq := colourseq, 0, 0, 0 end do
else for n to nops(varlist) do colourseq := colourseq, colourlist[n][ ] end do
end if;
for n to nops(varlist) do
    DATA['Plotdata'] := DATA['Plotdata'], [seq([i, varlist[n][i]], i=t .. T)]
end do;
DATA['Plotdata'] := 'CURVES'(DATA['Plotdata'], 'COLOUR'('RGB', colourseq)),
    TITLE(DATA['Title']), 'VIEW'(1 .. T+1, varmin .. varmax),
    'AXESSTYLE'(NORMAL),
    'AXESTICKS'(4, ymarks, 'FONT'(COURIER, DEFAULT, 9));
PLOT(DATA['Plotdata'])
end proc

```

2.3. Maple programs for random walks and Polya processes

Now we are ready for developing the programs. We start with the random walk program that can be invoked by a procedure call with two parameters: **walk(T, seed)**. The first parameter (**T**) is simply the integer number of periods for the simulation. The second (**seed**) is a little more complicated to understand: it is an integer that determines a unique sequence of random numbers. Thus if you call the procedure with **walk(100,1)**; the result is a random walk for 100 periods (ie. with 100 adoption decisions). If you call the procedure once more with **walk(100,1)**; the first 100 steps will be exactly the same as before. However, if you call the procedure with with **walk(100,2)**; you will see a totally different sequence of random numbers. Any seed produces its own unique sequence.

The real program is found in the loop **for t to T do ... end do**; Here you see that we call a random number generator called **coin** that gives values 0 and 1 with equal probability. If the value of 1, system *A* is chosen; otherwise system *B* is chosen. Finally, we call the DataPlot procedure to produce two time plots: one of the movement of the market share of *A* and one of the difference between the number of adoptions of *A* and *B*.

```
> walk := proc(T, seed)
  local A_share, choice, coin, d, d_max, d_min, n_A, n_B, t;
  global _seed;
  option `E.S. Andersen, 10 Feb 2001`;
  _seed := seed;
  d := array(0 .. T);
  d[0] := 0;
  d_max := 0;
  d_min := 0;
  A_share := array(1 .. T);
  A_share := 0;
  n_A := 0;
  n_B := 0;
  coin := rand(0 .. 1);
  for t to T do
    choice := coin( );
    if choice=1 then n_A := n_A+1 else n_B := n_B+1 end if;
    d[t] := n_A-n_B;
    d_max := max(d[t], d_max);
    d_min := min(d[t], d_min);
    A_share[t] := n_A/t
  end do;
  print(`\n`);
  print(DataPlot([A_share], 1, T, 0, 1, 5, black, ms));
  print(`\n\n`);
  print(DataPlot([d], 1, T, d_min, d_max, 5, black, dt))
end proc
```

The program for Polya processes is called by **polya(T,N,seed)**. The new thing is **N**, the number of simulations that you want to have plotted in a single figure. To make several runs, the program uses some Maple tricks that make it a little more difficult to interpret. The important thing is that we create a random number generator that produces uniformly distributed numbers between 0 and 1. In the loop **for t to T do ... end do**; a random number is produced. If it is smaller than the market share of *A*, then the person adopts *A*.

```
> polya := proc(T, N, seed)
  local A_share, chance, n, n_A, n_B, RandGenerator, t;
```

```

global _seed;
option `E.S. Andersen, 10 Feb 2001`;
  RandGenerator := stats[random, uniform[0, 1]]('generator');
  _seed := seed;
  for n to N do
    A_share||n[0] := 1/2;
    n_A := 1;
    n_B := 1;
    for t to T do
      chance := RandGenerator( );
      if chance( ) < A_share||n[t-1] then n_A := n_A+1 else n_B := n_B+1 end if;
      A_share||n[t] := n_A / (n_A+n_B)
    end do
  end do;
  print(`\n`);
  print(DataPlot([seq(A_share||n, n=1 .. N)], 0, T, 0, 1, 5, black, ms))
end proc

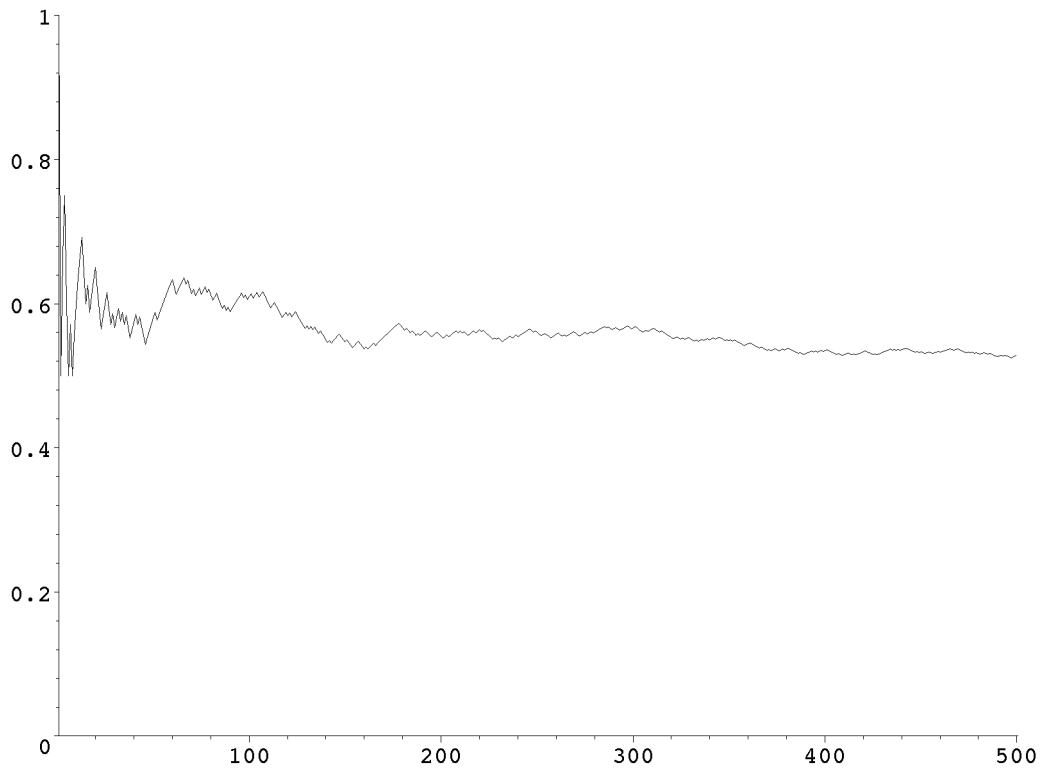
```

2.4. Simulating with the programs

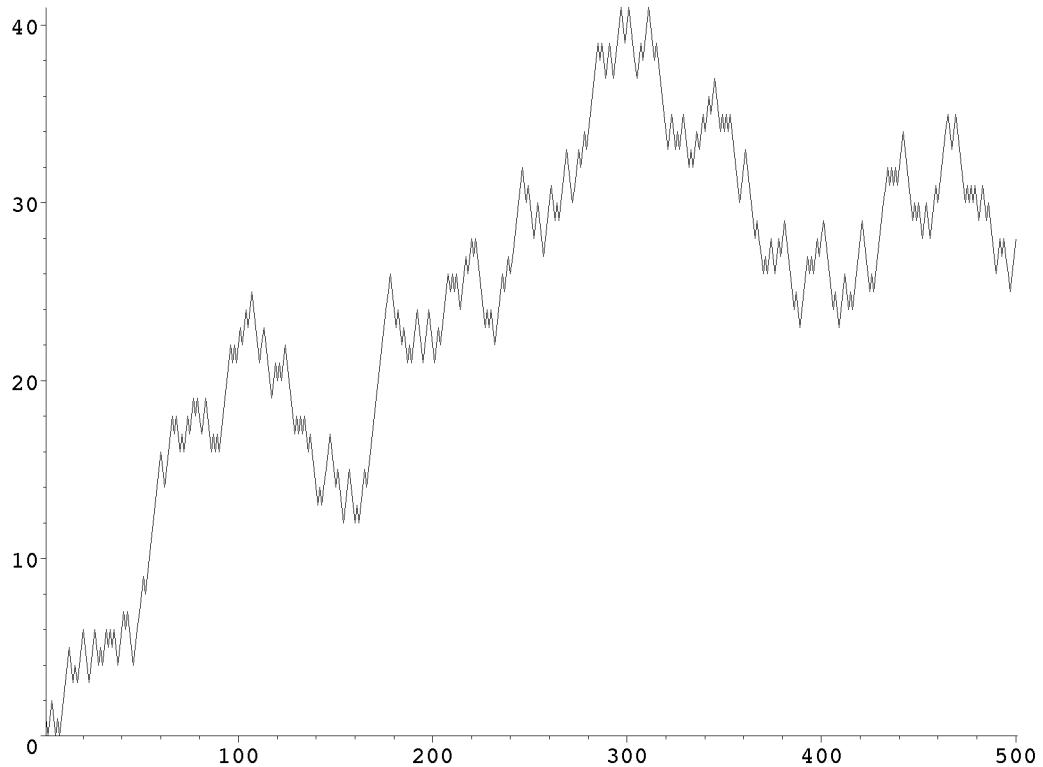
As an example of a random walk we take 500 periods based on the seed value 1. In the first plot we see that the movements of the market share of *A* gradually becomes smaller and (as predicted) approaches 0.5. In the second plot we see the random walk of the difference between the number of adopters of the two systems. However, during the 500 periods *A* has all the time more adopters than *B*. This is not in accordance with our predictions that the walk should also shown an overweight of adopters of *B*. To check that everything is in order, we extend the number of periods in the next simulation.

```
> walk(500, 1);
```

Market share for system A



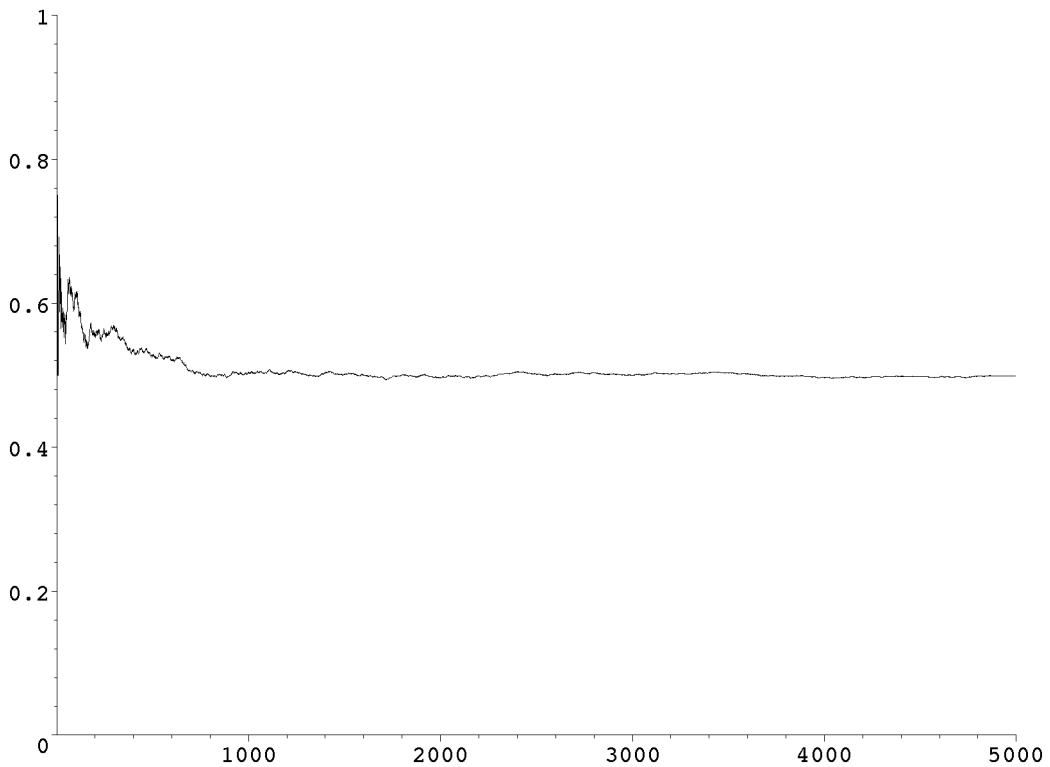
Difference between install bases of A and B



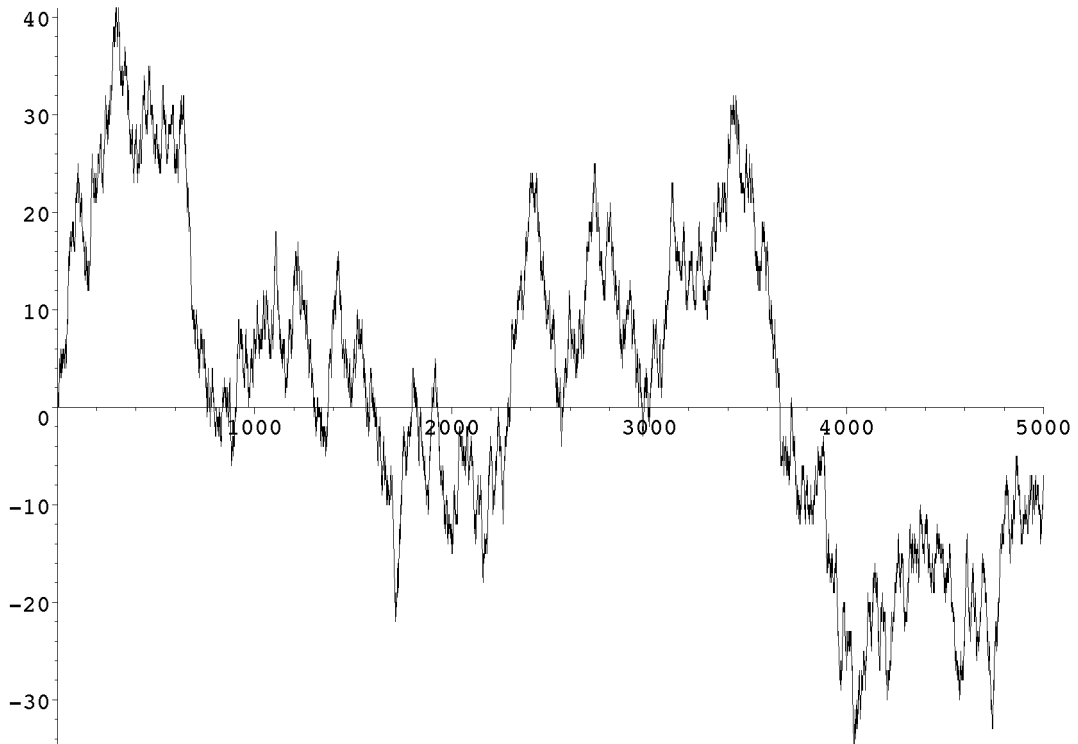
We now make a simulation for 5000 periods with the same seed (and thus the same first 500 adoptions). Now we see the expected behaviour of both the market share of A and the difference between the number of adoptions of A and B.

```
> walk(5000,1);
```

Market share for system A



Difference between install bases of A and B

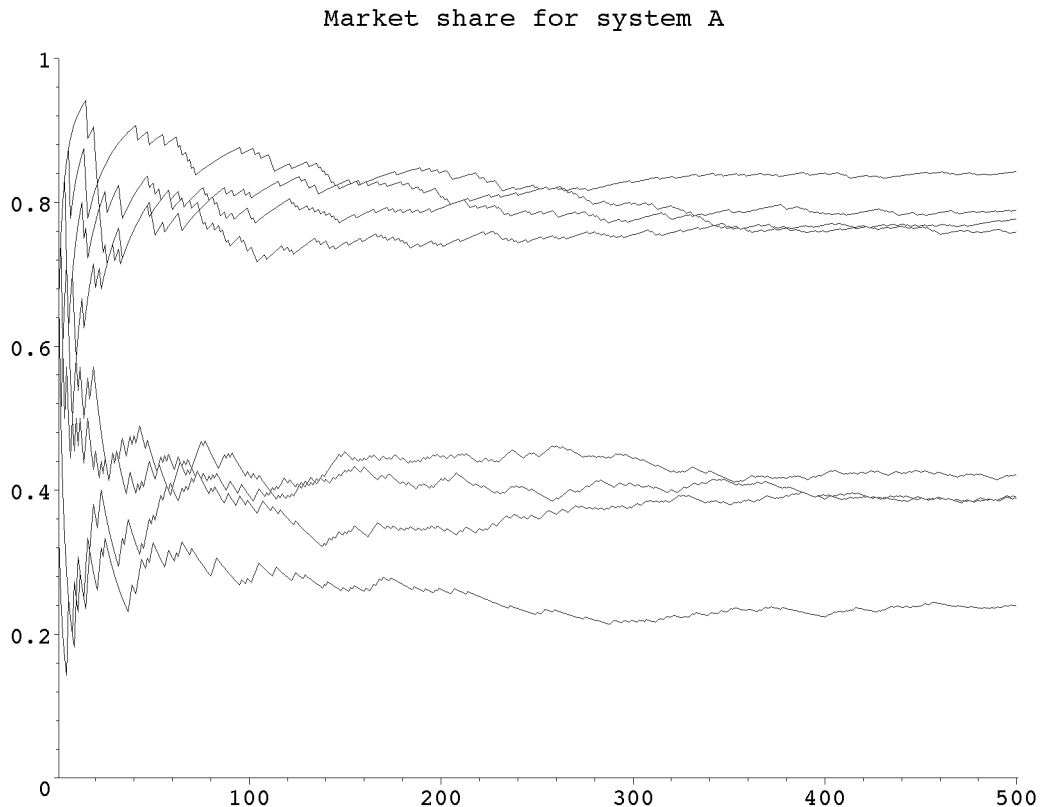


As mentioned in section 2.2, random walks are not only pretty boring but also unrealistic. As a first step towards more "realism" about the diffusion process, we turn to the Polya process. We make the procedure call **polya(600,8,1)**; ie. we simulate for 8 different runs for 600 periods. You may ask why this gives 8 different stochastic processes, since we have set the seed to 1. However,

the seed is only set once by us. Maple has its own internal seed value and this is different for each of the simulation runs. However, if you run the procedure once more with the same seen and the same number of runs, you will see exactly the same picture.

The Polya procedure generates the expected results (see section 2.2). In the beginning there are large fluctuations in market shares, but very quickly the system settles down to the predicted pattern: due to the initial path-dependency the system is brought to equilibria and these equilibria seems to be stable.

```
> polya(500, 8, 1);
```



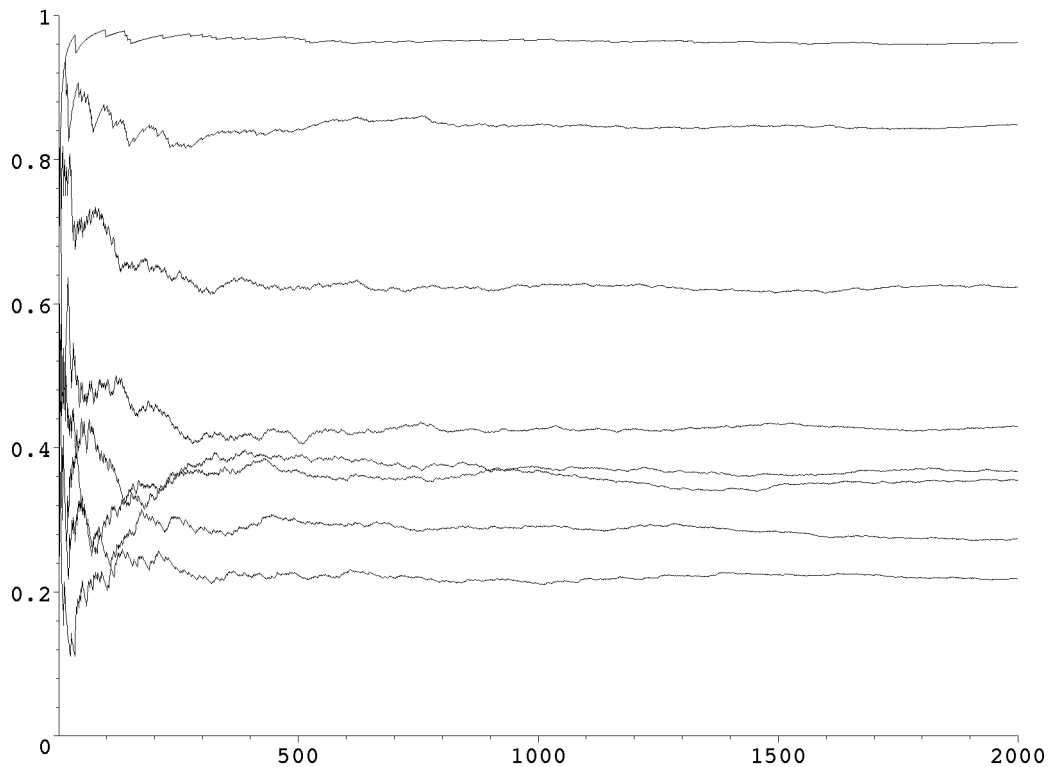
We would like to extend the number of periods of the above simulation plot for two reasons. First to make sure that each of the different paths are really an equilibrium path. Second, from the plot it is not obvious that all the real numbers between 0 and 1 can emerge as equilibrium market shares.

As our **polya** procedure is constructed, this is not possible. The reason is the following. The random number generator produces a single sequence of uniformly distributed numbers based on the given seed value. In the case of 8 different simulations for 500 periods, there is produced 4000 random numbers. In the case of 8 runs for 2000 periods, the 4000 numbers are used for the two simulations. When the program starts the second simulation it starts with a different random number than was used in the 500-period simulation.

This, of course, does not mean that we cannot study the properties of Polya processes. The questions of the stability of the equilibrium and the even distribution of the equilibria over the interval can be confronted by making many simulation experiments and by studying the underlying mathematics. In the present note we shall just consider one set of 8 simulation – each for 2000 periods. The plot below corresponds to our expectations.

```
> polya(2000, 8, 1);
```

Market share for system A



2.5. Empirical considerations

When we see a time series produced by an unknown mechanism, it is not easy to find out whether the patterns of the series are due to a random walk (or a Polya process). One tool for determining the question is spectral analysis (Fourier analysis) of the time series. A very simple introduction to this topic is found in Peak and Frame (1994, pp. 187-204). Here we should just note a few points of how the analysis proceeds. First, we define the length of the time interval (the "frequency", f) for the study. Second, we study the change of the variable that take place for each time interval. Third, we calculate the number of changes that fall into different size categories ("intensities of change"). Finally, we study the functional dependence between frequency and intensity. If we find

that $intensity = \frac{c}{frequency^2} + e$ (with a constant c and an error term e), then we have a random walk

(also called a Brownian motion) and we say that the data is characterised by Brownian noise or

that we have a $\frac{1}{f^2}$ spectrum.

The reason is found in the incremental nature of random walks. Large change within a small time interval presuppose the unlikely case of many random moves in the same direction.

Spectral analysis can also be used to find other underlying mechanisms. If we, for instance, find that $intensity = c + e$, then we have total randomness ("white noise") – but this is not easy to discern from deterministic chaos.

2.6. Conclusions

The study of random walks and Polya processes helps to the understanding of several concepts and problems. Among these are:

- dynamical systems whose movement is partly determined stochastically
- the law of large numbers as giving equilibria although individual decisions are random
- multiple equilibria and a selection between them because of small historical events in the beginning of the (eg. Polya) process

- the possibility of working backwards from time series data generated by an unknown mechanism to the underlying process – eg. by means of spectral analysis.

3. Arthur's competing technology model

3.1. Arthur's contribution

The Polya process demonstrates a simple case of path dependence where the effects of decisions by earlier adopters on the decisions of later adopters determines ultimate equilibrium outcome. Such an influence of early events is often significant in network markets and other areas of technology. But there is a need of adding a little more structure to the model before interesting problems arise. The main issue is that of increasing returns and positive feedback. These issues have especially been emphasised by Paul David (1985) and Brian Arthur (1989). They emphasise that path dependence induces a potential inefficiency arising from small differences in initial conditions which lead to outcomes that are likely to be costly to change. The case of the Qwerty keyboard is a well-known example of the path-dependence problem. The current dominance of the Qwerty keyboard today is not thought to be due to its superiority for typing but because it was invented earlier than the Dvorak keyboard. Although Liebowitz and Margolis (1994) have challenged the importance of the path dependency problem, it seems to be a basic ingredient of network economics.

Because of Arthur's emphasis of basic problems and a relationship to the proper issues of increasing returns in technological development (partly in connection with the Santa Fe Institute of complexity studies), he is a good starting point for entering the debate. His main papers are reprinted in Arthur: *Increasing Returns and Path Dependence in the Economy* (1994). The papers in this book are relatively easy to access because they combine stylised models of core issues with important background materials. The best model to start with was originally published as Arthur (1989). A somewhat extended version is found in Arthur (1994, pp. 13–32). This paper is used for the following reconstruction and exploration by means of simulation.

The starting point is an understanding of random walks and Polya processes. We have already seen how the adoption decision between two systems A and B takes place here. Arthur makes a slight addition, namely that the utility of a system is not only determined by the intrinsic preferences of each potential user but also by the number of users that have already adopted the system. Here Arthur is especially interested in the increasing returns case where the utility of a system for a new user increases with the number of users that have already adopted the system.

To keep things simple Arthur concentrates on systems that are not controlled and exploited by a particular firm (like the Windows operation whose diffusion is influenced by the strategy of Microsoft). Instead he thinks of the Qwerty keyboard which is just one of the approximately 10^{40} ways of organising the keys of a keyboard (although Dvorak took a parent of a better layout). If we further assume that the systems costs the same, then we have a pure user-side selection process. This is the topic of Arthur's paper and other contributions comes from Farrell and Saloner. Katz and Shapiro are among the authors who emphasise the decision making of firm-controlled systems.

3.2. The elements of the model

We model the evolution of at the adoptions of two non-sponsored standards or systems, A and B (like in an abstract Qwerty/Dvorak story). At each point of time, $t=1 \dots T$, one new buyer enters the game and select s once-and-for-all-times one system. The buyer is randomly selected between two infinite sets of buyer types, R and S . If we ignore a network effect, the former type prefers system A while the latter prefers system B . However, there is a network externality from the stock of adopters of a system (its install base) which influences the decision of the next adopter.

Let us summarise the notation:

- two non-sponsored and unchanging standards $i=A, B$
- two types of consumers $j=R, S$

The core of the model is the utility functions

$$u_j = a_{ij} + b_j n_{it}, \text{ where}$$

- n_{it} is the number of adopters of standard i in period t
- a_{ij} is the intrinsic preference of a consumer of type j for system of type i
- $a_{BR} < a_{AR}$ **and** $a_{AS} < a_{BS}$ (to make the choices non-trivial)
- b_j is the size of a j -type buyer's evaluation of the effect of each of the adopter's effect on his own utility
- b_j reflects the character of network externalities
 - $0 < b_j$: increasing returns to the scale of adoption
 - $b_j = 0$: constant returns to scale
 - $b_j < 0$: decreasing returns to scale

In each round one consumer of type $j = \{ R, S \}$ comes with equal probability to the market and buy one unit. After the purchase the consumer is locked-in to the chosen standard for ever.

3.3. Model analysis

Since one system is chosen in each period, the total number of systems in period t is simply t .

Thus system A's installation share at time t is $\frac{n_{At}}{t}$. It is, however, more operational to discuss the dynamics of the stochastic process in terms of the difference in install base between the two standards, ie. $d_t = n_{At} - n_{Bt}$.

R-type agents maximise their utility by buying A if:

$$a_{BR} + b_R n_{Bt} \leq a_{AR} + b_R n_{At}, \text{ and thus}$$

$$\frac{a_{BR} - a_{AR}}{b_R} \leq n_{At} - n_{Bt}, \text{ or, in abbreviated form: } \Delta_R \leq d_t$$

S-type agents buy A if:

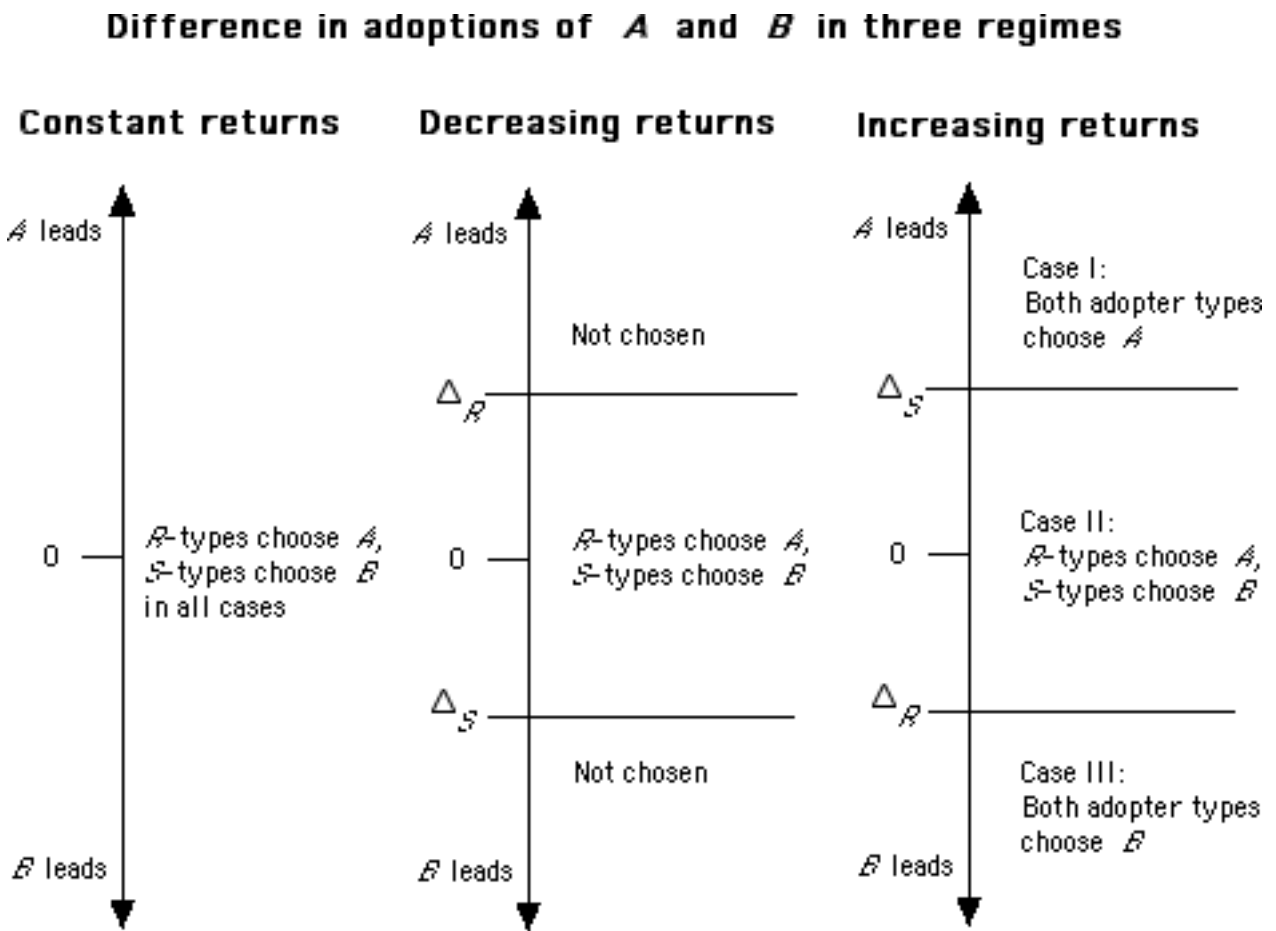
$$n_{At} - n_{Bt} \leq \frac{a_{BS} - a_{AS}}{b_S}, \text{ or, in abbreviated form: } d_t \leq \Delta_S$$

We thus see that all agents choose their preferred standards as long as d_t belongs to the interval $[\Delta_R, \Delta_S]$. The size of this interval is determined by the parameters. Let us study the decision of agent type R.

- If b_R approaches 0, then the interval approaches infinity and A will always be bought. This is the constant returns case.
- If b_R is negative, then Δ_R represents the highest possible dominance of A that will motivate its adoption. In this decreasing returns case we thus have an interval with "reflecting" boundaries. In this case have that $\Delta_S < \Delta_R$.
- If b_R is positive, then Δ_R represents the lowest possible weakness of A that will motivate its

adoption, and beyond this boundary R -type agents will always buy B . In this case we have that $\Delta_R < \Delta_S$.

The cases are depicted in the following figure:



3.4. Comparison of regimes

Criteria for comparing dynamical systems:

- Predictability: the extent to which an observer can predict the long-term outcome of the dynamical process.
- Flexibility: the ease with which the outcome of the process can be changed.
- Path Dependency: the extent to which the sequence of events determines the final equilibrium; non-path dependency (or ergodicity) means that the same set of events in a different sequence leads to the same result.
- Path efficiency: the selected path will be optimal seen from a long-term collective viewpoint.

Here is Arthur's summary of the properties of the three regimes:

	<i>redictable</i>	<i>Flexible</i>	<i>Path-dependent</i>	<i>Path-efficient</i>
<i>Constant returns</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>
<i>iminishing returns</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>
<i>Increasing returns</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>No</i>

4. Simulations with the basic Arthur model

4.1. The Maple program

The program for the Arthur model has many similarities with the programs for the random walk and the Polya process. The main difference is that we have to handle Arthur's six parameters of the utility functions of *R*-type agents and *S*-type agents. The chosen solution is that these parameters are set by the user as "global" variables outside the program. Their names are the same as in the model notation, but we cannot use subscripts, so they are called *a_AR*, *a_AS*, *a_BR*, *a_BS*, *b_R*, *b_S*. These parameters must be given values before the procedure is called by **arthur89(T,seed)**; where **T** is the number of periods (adoption decisions) and **seed** is the variable that starts a unique random sequence of { 0, 1 }.

```
> arthur89 := proc(T, seed)
  local A_share, chance, d, d_max, d_min, n_A, n_B, RandGenerator, t;
  global a_AR, a_AS, a_BR, a_BS, b_R, b_S, _seed;
  option `E.S. Andersen, 10 Feb 2001`;
  d := array(0 .. T);
  d[0] := 0;
  d_max := 0;
  d_min := 0;
  A_share := array(1 .. T);
  A_share := 0;
  n_A := 0;
  n_B := 0;
  RandGenerator := stats[random, uniform[0, 1]]('generator');
  _seed := seed;
  for t to T do
    chance := RandGenerator( );
    if 1/2 < chance then
      if a_BR + b_R * n_B ≤ a_AR + b_R * n_A then n_A := n_A + 1
      else n_B := n_B + 1
      end if
    else
      if a_AS + b_S * n_A ≤ a_BS + b_S * n_B then n_B := n_B + 1
      else n_A := n_A + 1
      end if
    end if;
    d[t] := n_A - n_B;
    d_max := max(d[t], d_max);
    d_min := min(d[t], d_min);
    A_share[t] := n_A / t
  end do;
  print(DataPlot([A_share], 1, T, 0, 1, 5, black, ms));
```

```

print(`\n\n`);
print(DataPlot([d], 1, T, d_min, d_max, 5, black, dt))
end proc

```

4.2. Simulation of constant returns (no boundaries)

The first simulation starts by setting the values of parameters for the constant returns case.

Because our program does not divide with the scale factor, we can use the values: $b_R=0$ and $b_S=0$.

In this case the exact value of the other parameters does not matter – as long as $a_{AS} < a_{AR}$ and

$a_{BR} < a_{BS}$. In this case R -type agents will always chose A , and S -type agents will always chose B .

This is simply a random walk in the difference between the number of adopters of A and B .

```

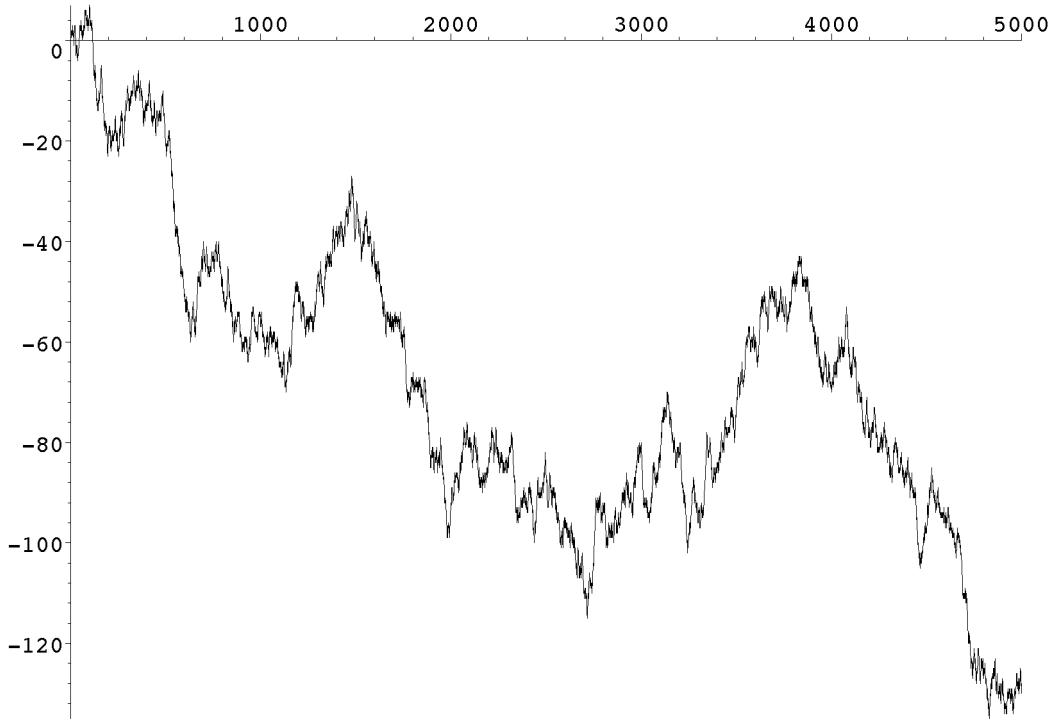
> a_AR := 10:
  a_AS := 5:
  a_BR := 5:
  a_BS := 10:
  b_R := 0:
  b_S := 0:
> arthur89(5000,5);

```

Market share for system A



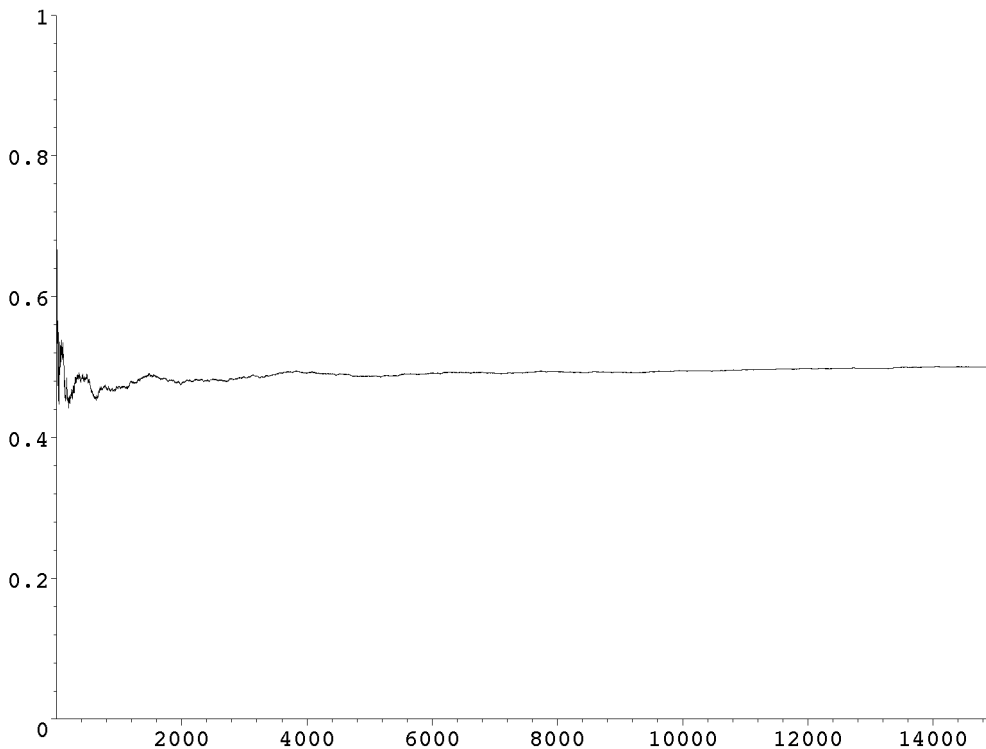
Difference between install bases of A and B

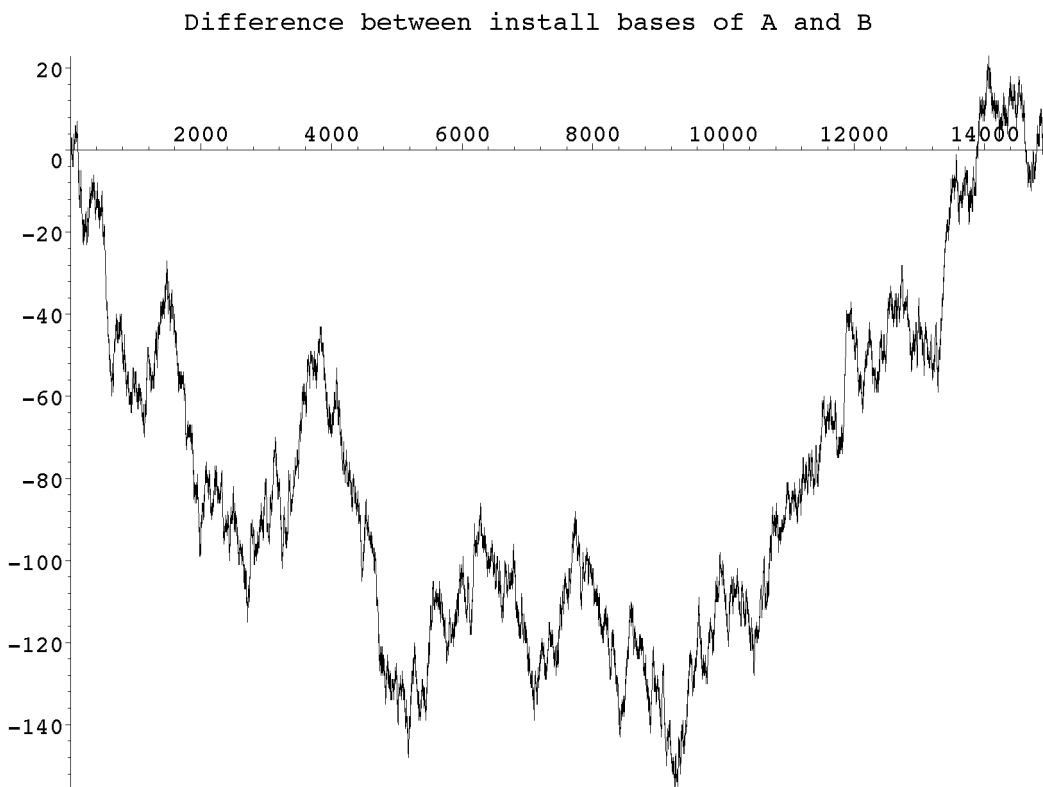


In the above plot of the market share of A we see that is, after an initial turbulent period comes closer and closer to 50%. The difference between the install bases of the two systems is more volatile. After 5000 periods, the process *seems* to be locked-in into a small overweight of system A. But sooner or later the process will move to an overweight of system B. If we continue the sequence of random events to cover 15000 periods, we see that the system moves towards an overweight of B. In the very long run the average should be zero.

> **arthur89(15000,5);**

Market share for system A





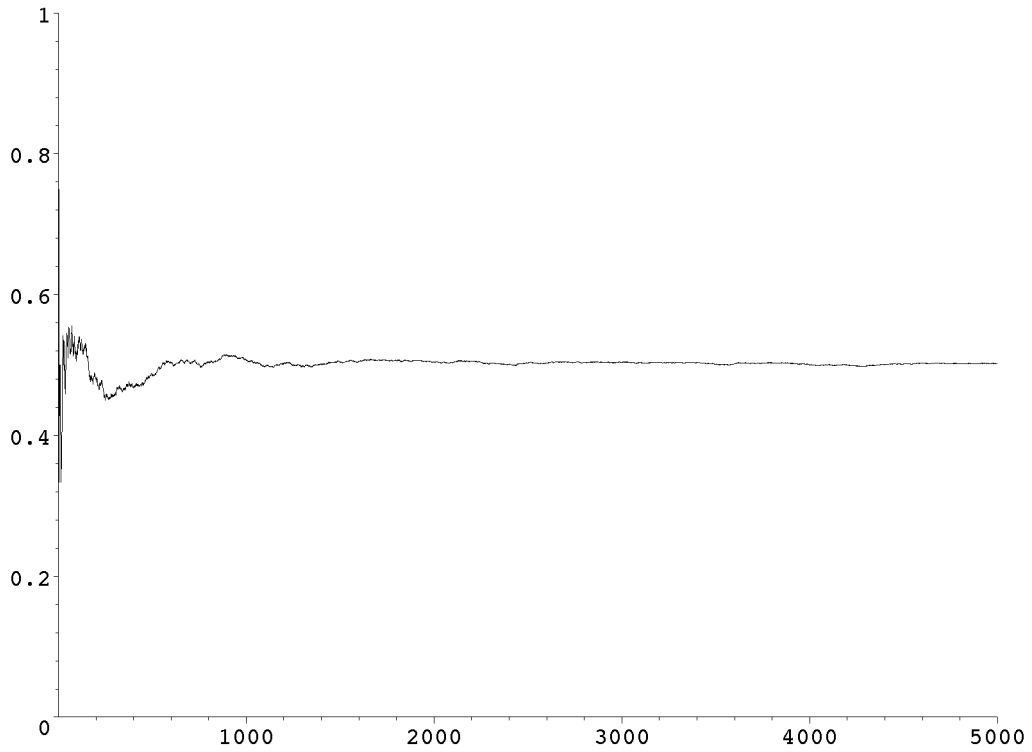
4.3. Simulation of decreasing returns (reflecting boundaries)

Now we turn to the decreasing returns case and thus to the story of negative feedback. This case is defined by $b_R < 0$ and $b_S < 0$, so now the exact value of the other parameters do matter, but we of course still have that $a_{AS} < a_{AR}$ and $a_{BR} < a_{BS}$. In the following the values of the parameters are set so that R -type agents have an intrinsic preference for system A that there is a need for an overweight of 25 adopters of system A in order to crowd them out from their preferred choice and switch to system B . S -type agents also needs the negative effects of an overweight of 25 adopters of B to switch from their preferred choice of system B to system A .

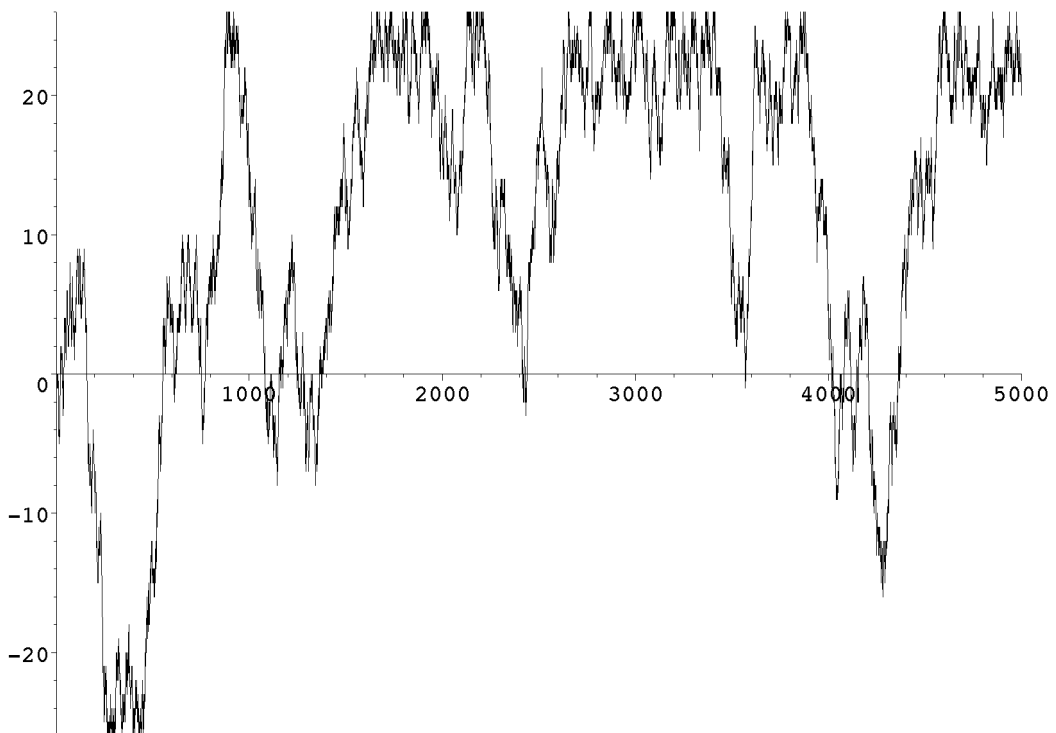
As long as the process does not reach the boundaries of the interval $[-25, 25]$ the process goes on just like in the constant returns case. As we see in the plot below, this is the case until period 275. Then the process hits the border. It continues to move randomly with occasional reflections from a border. In the very long run the process is expected to be just as frequently in the A overweight as in the B overweight (but this is not shown in the 5000 period simulation). It is, however, obvious that both install shares come very close to 50%.

```
> a_AR := 10:
  a_AS := 5:
  a_BR := 5:
  a_BS := 10:
  b_R := -0.2:
  b_S := -0.2:
> arthur89(5000,4);
```

Market share for system A



Difference between install bases of A and B



4.4. Simulation of increasing returns (absorbing boundaries)

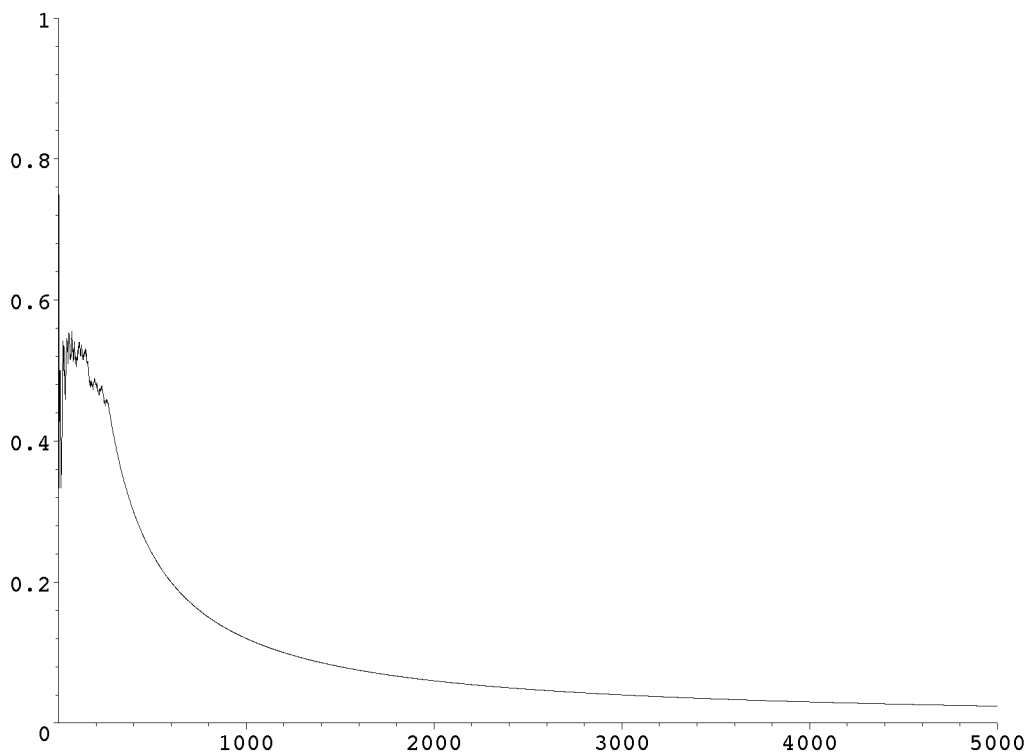
Finally, we come to Arthur's core case of increasing returns to adoption and thus to the positive feedback case. Given the previous analysis, this case is simple to handle. The case is defined by $0 < b_R$ and $0 < b_S$, and we of course still have that $a_{AS} < a_{AR}$ and $a_{BR} < a_{BS}$. We follow the decreasing

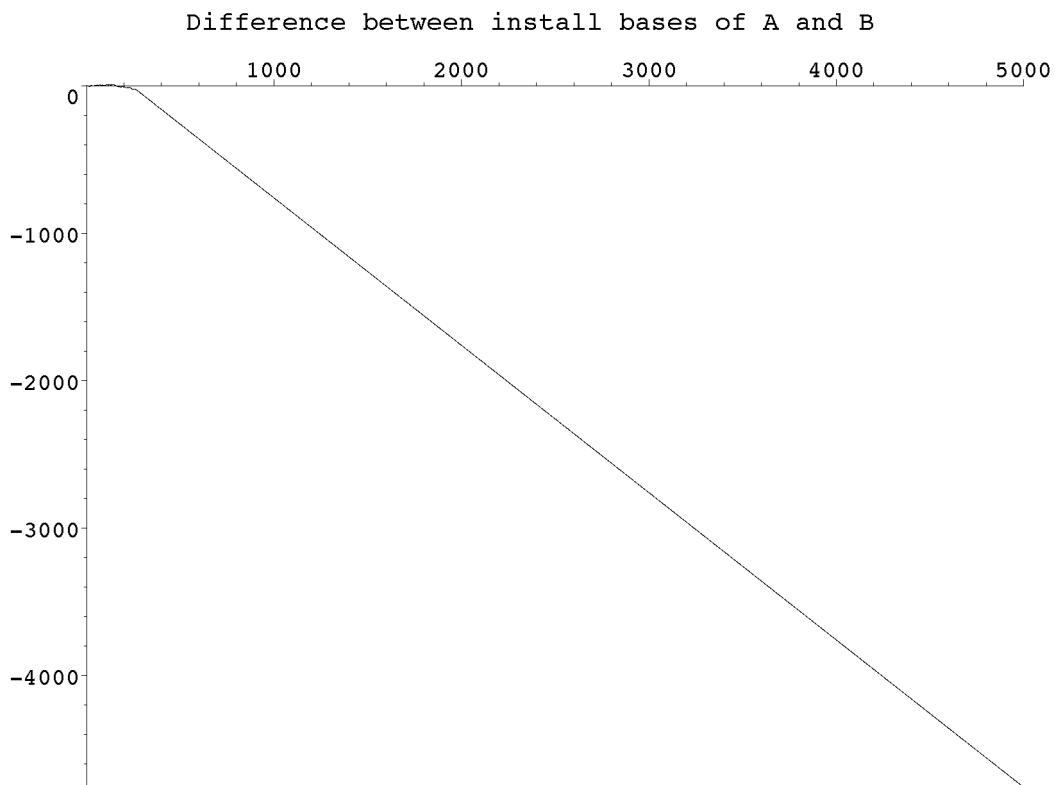
returns case in setting the rest of the parameters so that R -type agents need for an overweight of 25 adopters of system B in order to persuade them to switch to system B . S -type agents also needs an overweight of 25 adopters of A to switch from their preferred choice.

Until period 275 the process goes exactly as in the constant returns and decreasing returns cases. Then it is absorbed, and each and every new adaptation choose system B . Thus the market share of A moves towards zero, and the random walk in the difference of adoption is changed to a infinite directed walk in the B direction.

```
> a_AR := 10:  
a_AS := 5:  
a_BR := 5:  
a_BS := 10:  
b_R := 0.2:  
b_S := 0.2:  
> arthur89(5000,4);
```

Market share for system A

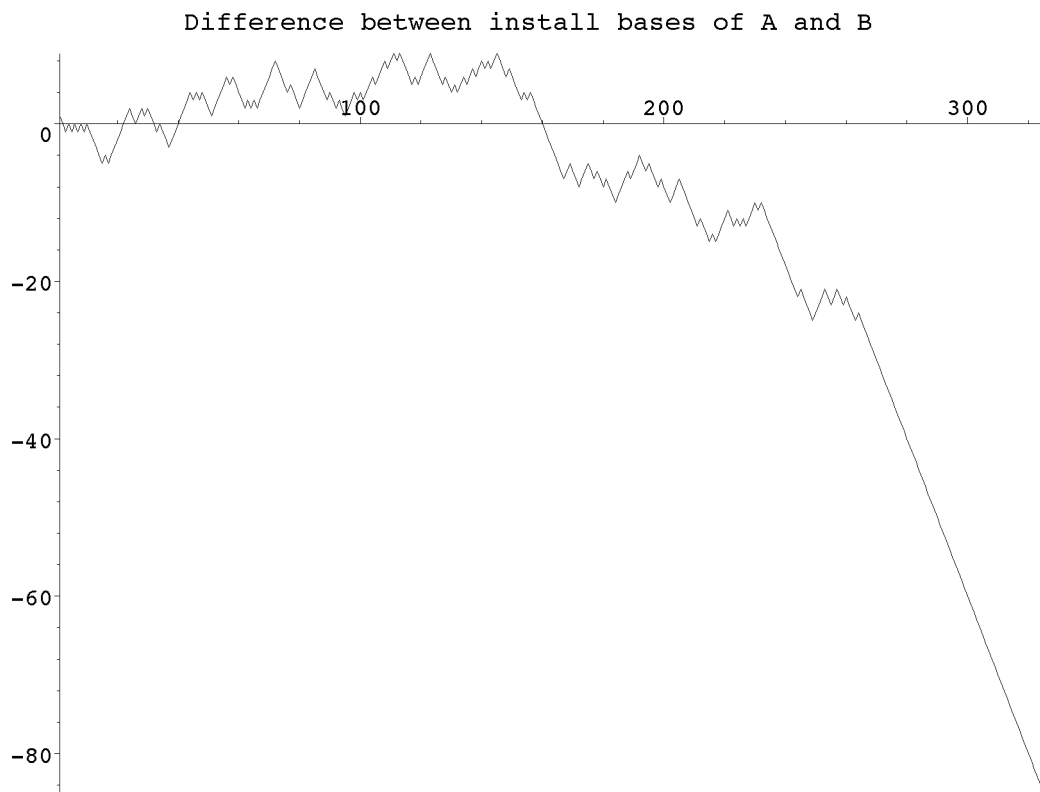




It is more relevant to study the process just before the lock in in period 275. This is shown in the following plot:

> **arthur89(325,4);**

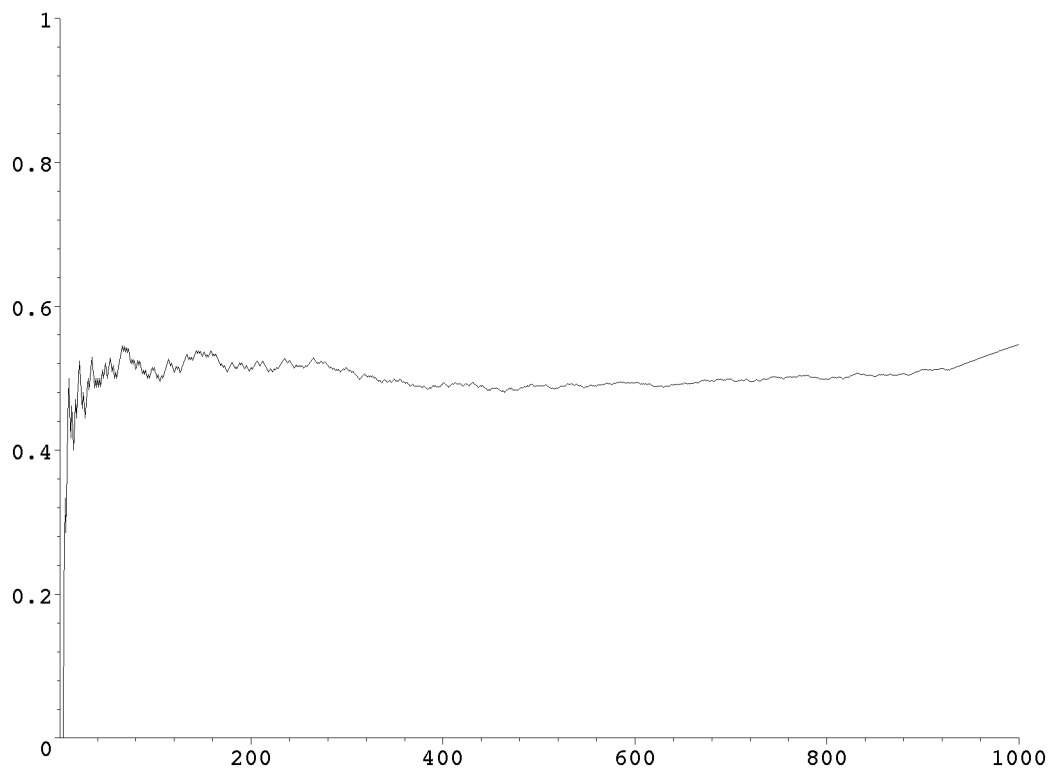




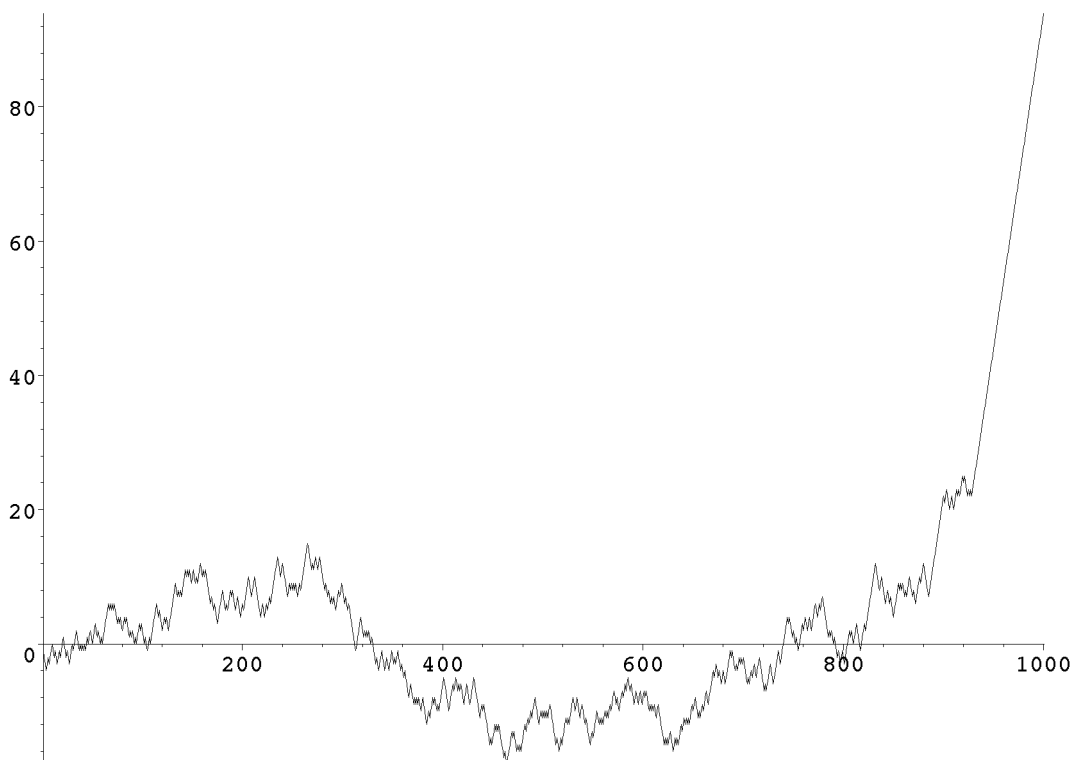
[We may also study the lock-in to system A after nearly 1000 periods...

> **arthur89(1000,1);**

Market share for system A



Difference between install bases of A and B



We do not need to make other simulations to demonstrate that the stochastic process with increasing returns is "tippy," as we know from network markets with inherent instability until they may reach a lock-in (cf. Bensen and Farrell, 1994). One may observe an unstable coexistence of incompatible products such as the classic case in the videocassette recorder market, but they also refer to video encryption of cable television programs. Dominance of one technology today does not guarantee continued success forever since such instability can happen.

5. Further experiments

Although there are many possibilities, the Arthur model is obviously constructed for the increasing returns case. Here we conclude:

- we have a random walk process with potentially absorbing barriers
- the process is not predictable (with probability 1 one of the barriers will be reached)
- ex-post inefficiency possible
- small events of history can be important under increasing returns to scale
- it is easy to introduce expectations that may quicken absorption

It is obvious that the Arthur model is just a simple starting point. Some of the problems to be exploited in further experiments are

- It is not obvious that the evaluation of external effects is connected to user types rather than to systems. In most cases a parameter b_A and b_B seem easier to interpret and more relevant.
- The role of expectations is not discussed in Arthur's paper. Rapid, random movement toward one system may create expectations that this system is "taking over". It is easy to include in the program.
- More learning behaviour is found in Arthur (1993).

References

- Arthur, W. Brian (1989), 'Competing Technologies, Increasing Returns, and Lock-In by Historical Events', *Economic Journal*, Vol. 99, pp. 116--131.

- Arthur, W. Brian (1993), 'On Designing Economic Agents that Behave Like Human Agents', *Journal of Evolutionary Economics*, Vol. 3, pp. 1--22.
- Arthur, W. Brian (1994), *Increasing Returns and Path Dependence in the Economy*, University of Michigan Press, Ann Arbor, Mich.
- Bensen, Stanley M, and Farrell, Joseph (1994), 'Choosing How to Compete: Strategies and Tactics in Standardization', *Journal of Economic Perspectives*, Vol. 8, pp. 117-310.
- David, Paul A. (1985), 'Clio and the Economics of QWERTY', *American Economic Review. Papers and Proceedings*, Vol. 75, pp. 332--337.
- Liebowitz, Stanley J., and Margolis, Stephen E. (1994), 'Network Externality: An Uncommon Tragedy', *Journal of Economic Perspectives*, Vol. 8, pp. 133--150.
- Peak, David, and Frame, Michael (1994), *Chaos Under Control: The Art and Science of Complexity*, Freeman, New York.
- Shapiro, Carl, and Varian, Hal R. (1999), *Information Rules: A Strategic Guide to the Network Economy*, Harvard Business School Press, Boston, Mass.