FH-Kiel, Germany
University of Applied Sciences

© **Andreas Thiemer, 2002**
e-mail: andreas.thiemer@fh-kiel.de

Genetic Learning of Firms in a Competitive Market

Summary:

This worksheet deals with the adaptive learning behaviour of boundedly rational agents in a competitive market and presents the model of Dawid/Kopel (1998) and Dawid (1999), which includes two versions. In the first scenario the firms make only quantity decisions. In the other scenario firms decide to exit or stay in the market, and then they decide how much to produce. A simple Genetic Algorithm (GA) with different coding schemes, similar to that used by Dawid and Kopel, simulates the adaptive learning of the firms. Depending from the setup of the GA the behaviour of these artificial agents converges to the rational expectations equilibrium of the market model.

A basic model of a competitive market

There are n firms which produce and offer the same good. Each firm $i = 0, 1 .. n-1$ produces a quantity y_i. All firms have the same **cost function**:

$$c(y) = \begin{cases} \alpha + \beta \cdot y^2 & \text{if } y > 0 \\ 0 & \text{if } y = 0 \end{cases} \qquad \text{where } \alpha, \beta > 0$$

The parameter α denotes the short term fixed costs of the firm, which only incur if $y>0$ (for example a fee for market entry). The **market supply** is:

$$Y = \sum_{i=0}^{n-1} y_i$$

The price p that clears the market in every period is determined by the **inverse demand function**:

$$p(Y, a, b, n) := a - \frac{b}{n} \cdot Y \qquad \text{where} \quad a, b > 0$$

In order to allow the clearance of the market $y \le \frac{a}{b}$ has to be satisfied.

The **profit** Π of a firm is given by:

$$\Pi(p, y, \alpha, \beta) := \begin{cases} p \cdot y - \alpha - \beta \cdot y^2 & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases}$$

In order to guarantee that $\Pi > 0$ at least for some values of p and y, we check:

$$\text{profit\_condition}(\alpha, \beta, a) := \begin{cases} \text{"Positive profits are possible!"} & \text{if } \alpha < \frac{a^2}{4 \cdot \beta} \\ \text{"Fixed costs are to high!"} & \text{otherwise} \end{cases}$$

The firms do not know the price p, when they have to decide which quantity to produce. However, each firm forms an expectation p_e about the price. Solving the **first order condition** of a profit maximum ($\frac{d}{dy}\Pi = 0$) we obtain:

$$\frac{d}{dy}\left(p_e \cdot y - \alpha - \beta \cdot y^2\right) \text{ auflösen}, y \rightarrow \frac{1}{2} \cdot \frac{p_e}{\beta}$$

But this solution for the optimal ouput is only valid, if the price expectation is higher than the **shut down price** (where $\Pi = 0$), which is the positive root of the following solution:

$$p_e \cdot y - \alpha - \beta \cdot y^2 \left| \begin{array}{l} \text{ersetzen}, y = \frac{1}{2} \cdot \frac{p_e}{\beta} \\ \text{auflösen}, p_e \end{array} \right. \rightarrow \begin{bmatrix} 2 \cdot (\alpha \cdot \beta)^{\left(\frac{1}{2}\right)} \\ -2 \cdot (\alpha \cdot \beta)^{\left(\frac{1}{2}\right)} \end{bmatrix}$$

In case of this limit price the optimal output $y > 0$ is:

$$\frac{1}{2} \cdot \frac{p_e}{\beta} \text{ ersetzen}, p_e = 2 \cdot (\alpha \cdot \beta)^{\left(\frac{1}{2}\right)} \rightarrow \frac{(\alpha \cdot \beta)^{\left(\frac{1}{2}\right)}}{\beta}$$

Thus we have to complete the **optimal output rule**:

$$y_{opt}\left(p_e\right) = \begin{cases} \frac{1}{2} \cdot \frac{p_e}{\beta} & \text{if } p_e > 2 \cdot (\alpha \cdot \beta)^{\left(\frac{1}{2}\right)} \\ \left[0 \text{ or } \frac{(\alpha \cdot \beta)^{\left(\frac{1}{2}\right)}}{\beta} \right] & \text{if } p_e = 2 \cdot (\alpha \cdot \beta)^{\left(\frac{1}{2}\right)} \\ 0 & \text{otherwise} \end{cases}$$

In a **homogenous rational expectations equilibrium** all firms have perfect foresight ($p=p_e$) and all take the same optimal action $y_{opt}(p_e)$. Hence, if $p_e>2\cdot(\alpha\cdot\beta)^{\left(\frac{1}{2}\right)}$ solving $Y=n\cdot y_{opt}(p_e)$ yields the equilibrium price if :

$$p_e=a-\frac{b}{n}\cdot n\cdot\left(\frac{1}{2}\cdot\frac{p_e}{\beta}\right) \text{ auflösen}, p_e \rightarrow 2\cdot\frac{a}{(2\cdot\beta+b)}\cdot\beta$$

Solving

$$2\cdot\frac{a}{(2\cdot\beta+b)}\cdot\beta=2\cdot(\alpha\cdot\beta)^{\left(\frac{1}{2}\right)} \text{ auflösen}, \alpha \rightarrow a^2\cdot\frac{\beta}{\left(4\cdot\beta^2+4\cdot\beta\cdot b+b^2\right)}$$

it is seen, that the condition $p_e>2\cdot(\alpha\cdot\beta)^{\left(\frac{1}{2}\right)}$ holds if and only if

$$\alpha\le\frac{a^2\cdot\beta}{(2\cdot\beta+b)^2}$$

On the other hand, if $p_e<2\cdot(\alpha\cdot\beta)^{\left(\frac{1}{2}\right)}$ such that $Y=0$, we get $p=a$. However this parameter constellation contradicts the "profit condition" from above. In this case no homogenous rational expectation equilibrium p_E exists. Thus:

$$p_E(a,b,\alpha,\beta):=\begin{cases}2\cdot\frac{a}{(2\cdot\beta+b)}\cdot\beta & \text{if } \alpha\le\frac{a^2\cdot\beta}{(2\cdot\beta+b)^2}\\ \text{"No homogenous rational expectation equilibrium exists!"} & \text{otherwise}\end{cases}$$

Scenario I: pure quantity decisions

In the basic scenario the same parameter values are used as in Dawid (1999, p. 124):

$a := 5$ $\qquad$ $\alpha := \frac{1}{4}$

$b := 5$ $\qquad$ $\beta := 1$

- Check the profit condition:

  profit_condition(α, β, a) = "Positive profits are possible!"

- Compute the **rational expectations equilibrium price**:

  $p_E(a, b, \alpha, \beta) \rightarrow \frac{10}{7}$

- Compute **optimal output** of one firm at the rational expectations equilibrium:

  $y_E := \frac{1}{2} \cdot \frac{p_E(a, b, \alpha, \beta)}{\beta} \rightarrow \frac{5}{7}$

Because our firms are only "boundedly" rational agents, they do not know this equilibrium values. Instead they learn from each other by imitation and "trial and error". Like Dawid we use a simple Genetic Algorithm (GA) to simulate the adaptive learning behaviour of the firms.

Excurs: A Stepwise Introduction to a Simple Genetic Algorithm

Step 1: Define the fitness function

First, we define the fitness function, which assigns at every time some positive "fitness value" to any firm. To rule out negative values we use the normalized version of the profit function:

$$fit(p, y, \alpha, \beta, a, b) := \Pi(p, y, \alpha, \beta) + \alpha + \beta \cdot \left(\frac{a}{b}\right)^2$$

Step 2: Define encoding and decoding

Typically a GA works on a population of binary strings ("chromosomes"), for example:

chromosome := (1 0 0 1)

Each firms output strategy is encoded by such a binary string. Thus, every chomosome encodes a real number in $[0, \frac{a}{b}]$. The following function decodes a single binary string:

$$DEZ(gen) := \sum_{j=0}^{spalten(gen)-1} gen_{0,j} \cdot 2^{spalten(gen)-j-1}$$

Example: $DEZ(chromosome) = 9$

The following subroutine decodes the strings of a whole population (i.e. all firms) where POP is a matrix which collect all strings of the population as single rows:

```
DEZ_POP(POP) := | for j ∈ 0 .. zeilen(POP) - 1
                |     gen ← submatrix(POP, j, j, 0, spalten(POP) - 1)
                |     z_j ← DEZ(gen)
                | z
```

Depending from the length (= BITS) of a chromosome the maximal encoded value of the real number vary.

Example: $ZMAX(BITS) := \sum_{i=0}^{BITS-1} 2^{BITS-1-i}$

$\Rightarrow$ $ZMAX(4) = 15$ or $ZMAX(10) = 1.023 \cdot 10^3$

Therefore, we normalize the decoded values such, that they fall into the output intervall [0, a/b]:

Example: $DEZ(chromosome) \cdot \frac{\frac{a}{b}}{ZMAX(spalten(chromosome))} = 0.6$

Step 3: Create an initial population

An initial population of firms is generated randomly, where "POPSIZE" means the number of firms:

```
POP0(BITS, POPSIZE) := | for i ∈ 0 .. BITS - 1
                       |     for j ∈ 0 .. POPSIZE - 1
                       |         x_{j,i} ← floor(rnd(1) + .5)
                       | x
```

Example:

$$P_{init} := POP0(4,8) \quad \Rightarrow \quad P_{init} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

... or decoded and normalized:

$$y_{init} := DEZ\_POP\left(P_{init}\right) \cdot \frac{\frac{a}{b}}{ZMAX\left(spalten\left(P_{init}\right)\right)} \quad \Rightarrow \quad y_{init} = \begin{bmatrix} 0.067 \\ 0.067 \\ 1 \\ 0.2 \\ 0.667 \\ 0.467 \\ 0.933 \\ 0.067 \end{bmatrix}$$

Step 4: Compute the fitness of the population

The market supply of this initial random selection of output decisions is:

$$Y_{init} := \sum_{i=0}^{zeilen\left(P_{init}\right)-1} y_{init_i} \quad \Rightarrow \quad Y_{init} = 3.467$$

Hence we obtain the market price:

$$p_{init} := p\left(Y_{init}, a, b, zeilen\left(P_{init}\right)\right) \quad \Rightarrow \quad p_{init} = 2.833$$

Using the fitness function for all members of the population we get:

$$F := \overrightarrow{fit\left(p_{init}, y_{init}, \alpha, \beta, a, b\right)} \quad \Rightarrow \quad F = \begin{bmatrix} 1.184 \\ 1.184 \\ 2.833 \\ 1.527 \\ 2.444 \\ 2.104 \\ 2.773 \\ 1.184 \end{bmatrix}$$

Step 5: Sort the population

This subroutine expands the population matrix with a new column which represents the fitness of every population member and sorts the chromsomes from the highest fitness in row 0 to the lowest fitness in the last row:

$$\begin{aligned} &SORT\_POP(Z, F) := \\ &\quad X \leftarrow erweitern(Z, F) \\ &\quad X \leftarrow spsort(X, spalten(X) - 1) \\ &\quad X \leftarrow umkehren(X) \\ &\quad X \end{aligned}$$

Example:

$$PF := SORT\_POP\left(P_{init}, F\right) \quad \Rightarrow \quad PF = \begin{bmatrix} 1 & 1 & 1 & 1 & 2.833 \\ 1 & 1 & 1 & 0 & 2.773 \\ 1 & 0 & 1 & 0 & 2.444 \\ 0 & 1 & 1 & 1 & 2.104 \\ 0 & 0 & 1 & 1 & 1.527 \\ 0 & 0 & 0 & 1 & 1.184 \\ 0 & 0 & 0 & 1 & 1.184 \\ 0 & 0 & 0 & 1 & 1.184 \end{bmatrix}$$

Step 6: Select the "fittest"

This subroutine models the **selection operator**, which is intended to implement the idea of the "survival of the fittest". This operator determines which of the chromosomes in the current population will be allowed to pass their "genetic material" (i.e. the output strategy) to the next generation. The parameter "KEEP" means the even number of the fittest firms which will remain unaltered for the next generation (they form the **elite**; for KEEP = 0 there is non elite formed). To fill up the "mating pool", also a **tournament selection** takes place. In every tournament two strings are selected randomly from the current population but only the one with the higher fitness value enters the mating pool. The output of this subroutine builds a matrix which includes the pairwise position numbers of the winners from the tournaments.

```
PAIR_T(KEEP, POPSIZE, BITS, POP) :=
    j ← 0
    REPLACE ← (POPSIZE − KEEP) / 2
    for i ∈ 0 .. REPLACE − 1
        j ← 0
        while j ≤ 1
            A ← floor(rnd(POPSIZE − 1) + .5)
            B ← floor(rnd(POPSIZE − 1) + .5)
            pair_{i,j} ← A  if  POP_{A,BITS} ≥ POP_{B,BITS}
            pair_{i,j} ← B  otherwise
            j ← j + 1
    pair
```

Example:

keep := 2

mating_pool := PAIR_T(keep, zeilen(PF), spalten(PF) − 1, PF) ⇒ mating_pool = $\begin{bmatrix} 0 & 4 \\ 3 & 3 \\ 4 & 1 \end{bmatrix}$

Step 7: Use the crossover operator

Crossover is intended to join the genetic material of chromosomes with a high fitness in order to produce better individuals. An economic interpretation is that crossover is (partial) imitation of successfull behaviour. The mating pool is splitted into pairs of strings called parents. Given the crossover probability CROSSPROB (often CROSSPROB=1 is used), **one crossover point** is drawn randomly between the bits of a string. The following subroutine gives these points for every pair of parents:

```
X(KEEP, POPSIZE, CROSSPROB, BITS) :=
    REPLACE ← (POPSIZE − KEEP) / 2
    yes ← runif(REPLACE, 0, 1)
    for i ∈ 0 .. REPLACE − 1
        yes_{i,0} ← 1  if  yes_{i,0} ≤ CROSSPROB
        yes_{i,0} ← 0  otherwise
    X ← runif(REPLACE, 1, BITS − 1)
    X ← (floor(X + 0.5) · yes)   (overscript →)
    X
```

Example:

$$X\_points := X(keep, zeilen(PF), 1, spalten(PF) - 1) \quad \Rightarrow \quad X\_points = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

Now for every pair of parents the bits to the right of the crossover point are swapped between these two parents:

$CHILD(POP, PAIR, XPOINTS, BITS) :=$

$REPLACE \leftarrow zeilen(PAIR)$

for $i \in 0 .. REPLACE - 1$

  for $j \in 0 .. BITS - 1$

    for $k \in 0 .. XPOINTS_{i,0}$

      $child_{2 \cdot i, k} \leftarrow POP_{PAIR_{i,0}, k}$

      $child_{2 \cdot i + 1, k} \leftarrow POP_{PAIR_{i,1}, k}$

    for $k \in XPOINTS_{i,0} .. BITS - 1$

      $child_{2 \cdot i, k} \leftarrow POP_{PAIR_{i,1}, k}$

      $child_{2 \cdot i + 1, k} \leftarrow POP_{PAIR_{i,0}, k}$

child

Example:

$$offsprings := CHILD(PF, mating\_pool, X\_points, spalten(PF) - 1)$$

$$\Rightarrow \quad offsprings = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Step 8: Use the mutation operator

The mutation operator should allow the GA to find strategies, which contain bit values that are not existent in the current population. In economic terms mutation stands for innovation. With mutation probability MUTPROB > 0 each bit in any string of the offsprings is inverted:

$$\begin{aligned}&MUT1(POP, MUTPROB) := \\ &\quad \text{for } i \in 0..\text{spalten}(POP) - 1 \\ &\quad\quad \text{for } j \in 0..\text{zeilen}(POP) - 1 \\ &\quad\quad\quad POP_{j,i} \leftarrow \left| POP_{j,i} - 1 \right| \text{ if } rnd(1) \le MUTPROB \\ &\quad POP\end{aligned}$$

Example:

$$mutated\_offsprings := MUT1(offsprings, 0.3) \quad \Rightarrow \quad mutated\_offsprings = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Step 9: The next generation

The elite and the mutated offsprings generate a new population of strategies. The procedure from step 4 to step 9 is repeated until a prescribed number of iterations MAXITER has been performed.

----------End of Excurs----------

The following **main routine** GA1 iterates the steps from above. For any period (= generation) the average production output of the firms will be computed.

Genetic Algorithm 1 (GA1)

Enter GA-Parameters:

$$POPSIZE := 30$$

$$KEEP := 0$$

$$BITS := 10$$

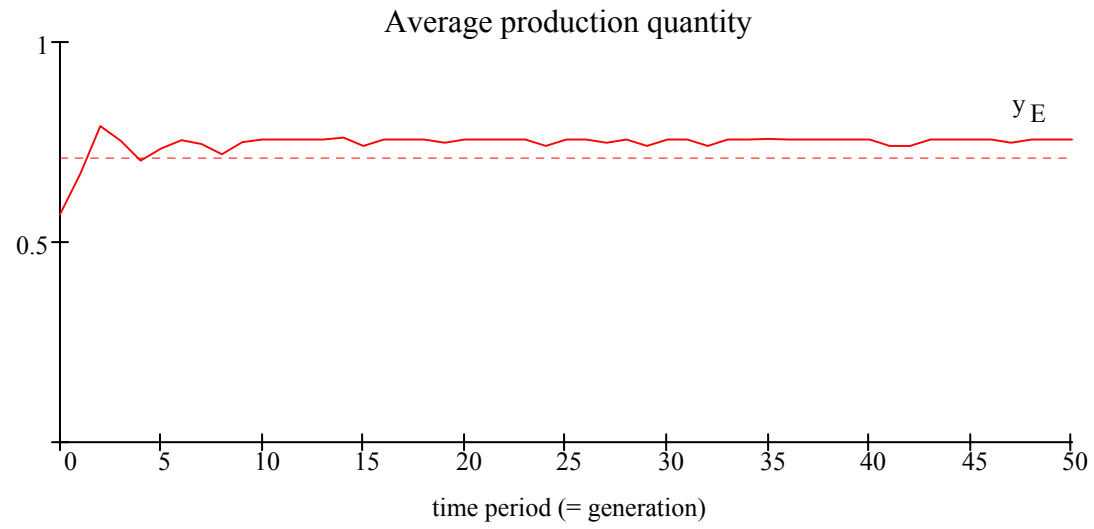
$$CROSSPROB := 1$$

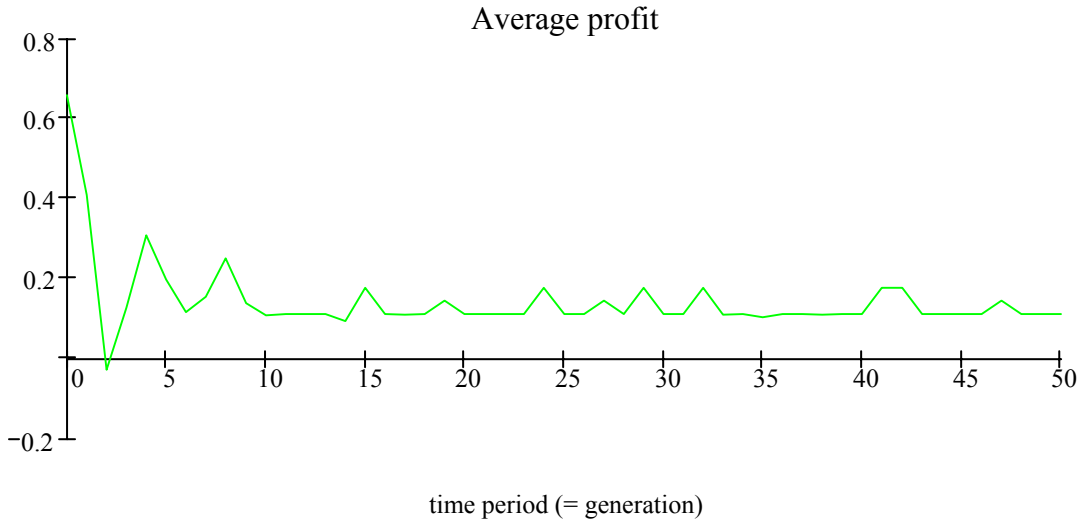
$$MUTPROB := .001$$

$$MAXITER := 50 \qquad time := 0, 1 .. MAXITER$$

$y_{mean} := GA1(POPSIZE, KEEP, BITS, CROSSPROB, MUTPROB, MAXITER, \alpha)$

To start a new simulation with the same parameters but with another initial random population click on the red command line and press the F9-key.





Try more simulations to see that the firms are able to adapt their production decision in such away that their output quantities converge towards the rational expectations equilibrium.

Scenario II: entry and exit decisions

Now the fixed costs are higher than before: $\alpha := 1$

- Check the profit condition:

  profit_condition(α, β, a) = "Positive profits are possible!"

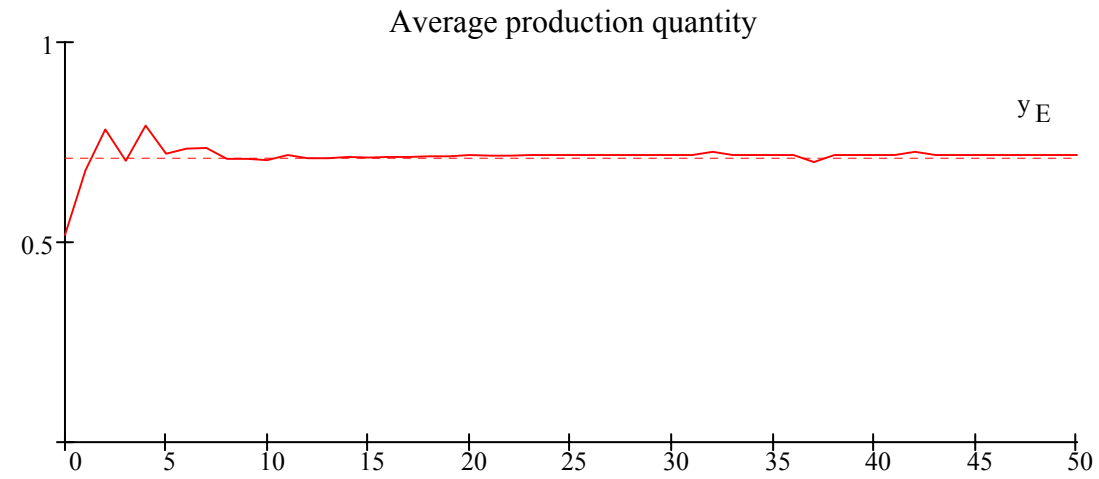
- Compute the **rational expectations equilibrium price**:

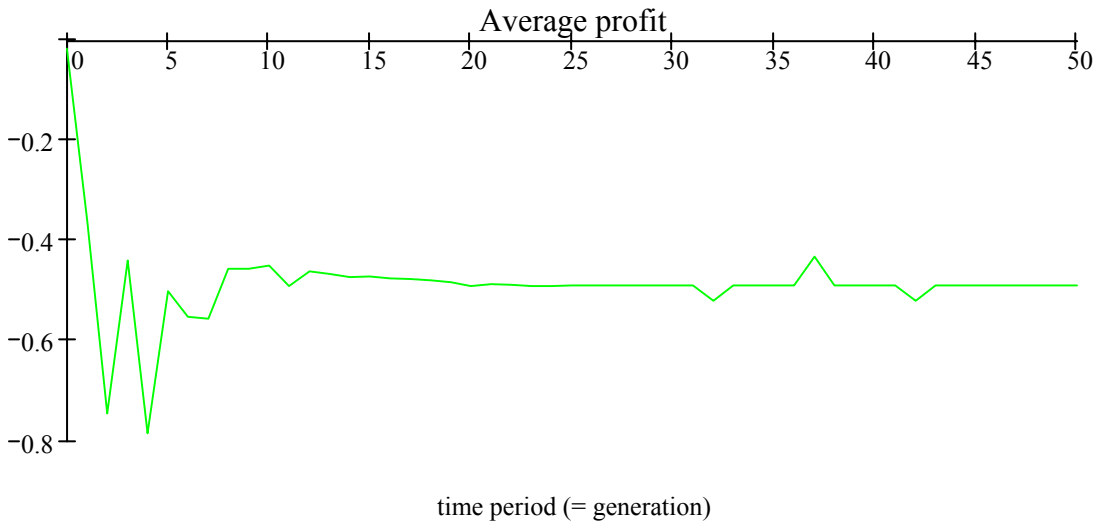
  $p_E(a, b, \alpha, \beta) \rightarrow$ "No homogenous rational expectation equilibrium exists!"

Start a simulation with the GA-Parameters from scenario I:

$y_{mean} := GA1(POPSIZE, KEEP, BITS, CROSSPROB, MUTPROB, MAXITER, \alpha)$

To start a new simulation with the same parameters but with another initial random population click on the red command line and press the F9-key.





Although there exists no homogenous rational expecatations equilibrium, the average produced quantity converges toward y_E and the price towards $p_E = a - b \cdot y_E$. There is a "lock in" of the GA in a "false" economic equilibrium. If all firms produce y_E their profit will be negative:

$$\Pi\left(a - b \cdot y_E, y_E, \alpha, \beta\right) = -0.49$$

Obviously any single firm will be better off by exiting the market with a profit of 0. But this does not happen, because the GA converges towards some kind of local equilibrium, where no firm can gain by a small unilateral deviation from its current strategy. Dawid (1999, proposition 4.6.2) gives a formal explanation why the GA behaves in this way. But an uniform population with all firms out of the market is also no economic equilibrium. In this case the price is p = a. Because this price is higher than the shut down price, any single firm will decide for market entrance! Thus in this case an economic equilibrium with an uniform population doesn't exist. In the long run only a profit $\Pi = 0$ can be realized. Therefore, the rational price expectation equals the shut down price:

$$p_E := 2 \cdot (\alpha \cdot \beta)^{\left(\frac{1}{2}\right)} \rightarrow 2$$

In such a **heterogenous rational expectations equilibrium** the firms have the same (self fulfilling) expectation, but the optimal production quantity is not unique. $y = 0$ and $y = \sqrt{\frac{\alpha}{\beta}} \rightarrow y = 1$ yield the optimal profit of 0. Denote the fraction of firms producing $\sqrt{\frac{\alpha}{\beta}}$ as x. Then

$$x := \left[2 \cdot \sqrt{\alpha \cdot \beta} = a - \frac{b}{n} \cdot (x \cdot n) \cdot \sqrt{\frac{\alpha}{\beta}} \right] \text{ auflösen}, x \rightarrow \frac{3}{5}$$

and the average production of all firms is $y_X := \sqrt{\frac{\alpha}{\beta}} \cdot x \rightarrow \frac{3}{5}$.

With the previous coding scheme the basic decision to exit the market (y = 0) can only be made by changing all bits in the string to 0. It seems more plausible to seperate the decision of the firm in two parts:

1. Entry (or exit) decision
2. Quantity decision

To represent this stepwise decision making in the GA, Dawid (1999) adds a special bit to the string. Only if this bit has value 1, the quantity encoded by the other bits in the string is really produced. Otherwise no production takes place. Thus, we have to modify our decoding procedure:

```
DEZ_POP(POP) := | M ← spalten(POP)
                | for j ∈ 0 .. zeilen(POP) − 1
                |     gen ← submatrix(POP, j, j, 0, M − 2)
                |     z_j ← DEZ(gen) · POP_{j, M − 1}
                | z
```

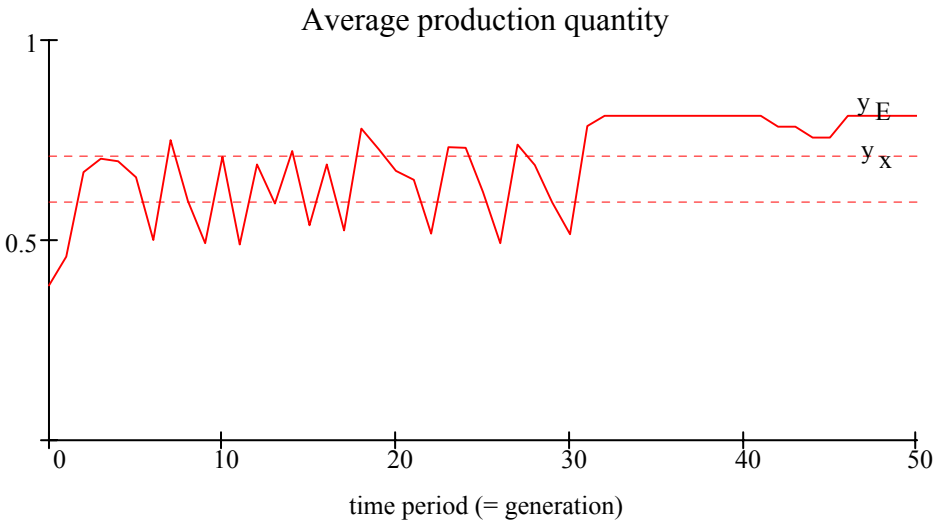
Also some small adjustments of the main routine are necessary.

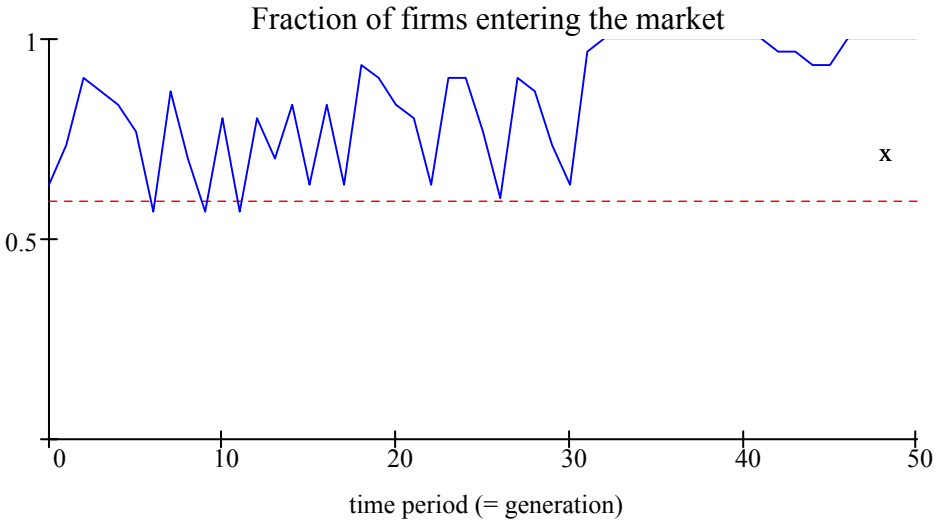
Genetic Algorithm 2 (GA2)

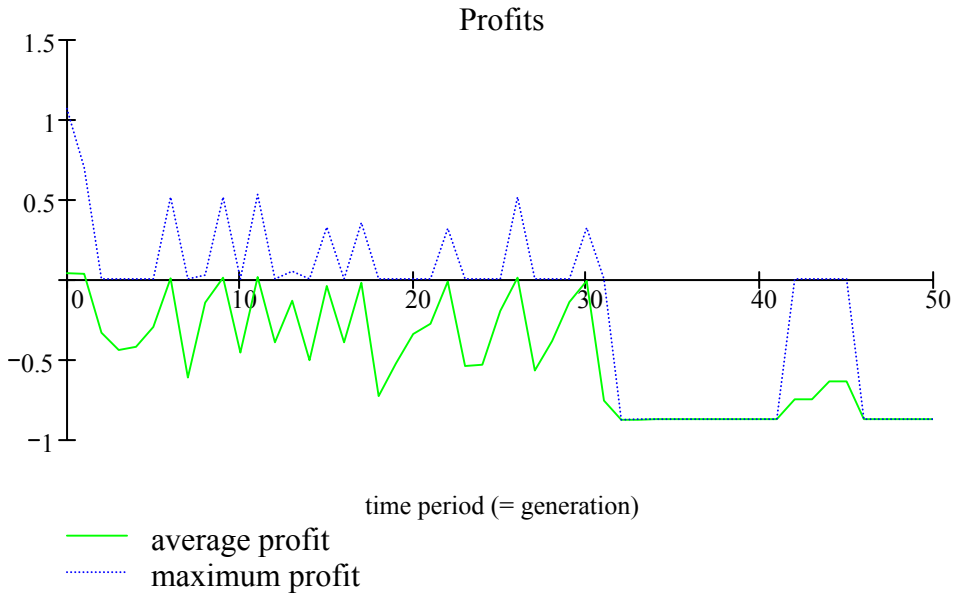
Running this program the same parameters are used as before.

$$out := GA2(POPSIZE, KEEP, BITS + 1, CROSSPROB, MUTPROB, MAXITER, \alpha)$$

To start a new simulation with the same parameters but with another initial random population click on the red command line and press the F9-key.







Now the average production ends up oscillating heavily around y_X. Although the average profit may be negative, there arise single firms with extraordinary high positive profits, if the fraction of firms exiting from the market is high. Hence in the next period more firms are allured to market entry (via the selection operator) and profits decrease, leading to market exits in the following period.

Literature:

- Dawid, H./Kopel, M.: On Economic Applications of the Genetic Algorithm: a Model of the Cobweb Type. In: Evolutionary Economics, vol. 8 (1998), 297 - 315.
- Dawid, H: Adaptive Learning by Genetic Algorithms. Berlin et al. 1999.
- Haupt, R.J./Haupt, S.E.: Practical Genetic Algorithms. New York et al. 1998.