El-Shagi, Makram

Working Paper

An Evolutionary Algorithm for the Estimation of Threshold Vector Error Correction Models

IWH Discussion Papers, No. 1/2010

Provided in Cooperation with:
Halle Institute for Economic Research (IWH) – Member of the Leibniz Association

Suggested Citation: El-Shagi, Makram (2010) : An Evolutionary Algorithm for the Estimation of Threshold Vector Error Correction Models, IWH Discussion Papers, No. 1/2010, Leibniz-Institut für Wirtschaftsforschung Halle (IWH), Halle (Saale), https://nbn-resolving.de/urn:nbn:de:101:1-2010031929

Institut für Wirtschaftsforschung Halle
Halle Institute for Economic Research

An Evolutionary Algorithm for the Estimation of Threshold Vector Error Correction Models

Makram El-Shagi

January 2010 No. 1

IWH-Diskussionspapiere
IWH-Discussion Papers

An Evolutionary Algorithm for the Estimation of Threshold Vector Error Correction Models

Makram El-Shagi

January 2010 No. 1

Author: *Makram El-Shagi*
Halle Institute for Economic Research
Department of Macroeconomics
Phone: +49 345 7753 835
Fax: +49 345 7753 799
Email: Makram.El-Shagi@iwh-halle.de

Editor:

Halle Institute for Economic Research (IWH)
Prof Dr Dr h. c. Ulrich Blum (President), Dr Hubert Gabrisch (Head of Research)
The IWH is member of the Leibniz Association.

Address: Kleine Märkerstraße 8, 06108 Halle (Saale)
Postal Address: P.O. Box 11 03 61, 06017 Halle (Saale)
Phone: +49 345 7753 60
Fax: +49 345 7753 20
Internet: http://www.iwh-halle.de

An Evolutionary Algorithm for the Estimation of Threshold Vector Error Correction Models*

Abstract

We develop an evolutionary algorithm to estimate Threshold Vector Error Correction models (TVECM) with more than two cointegrated variables. Since disregarding a threshold in cointegration models renders standard approaches to the estimation of the cointegration vectors inefficient, TVECM necessitate a simultaneous estimation of the cointegration vector(s) and the threshold. As far as two cointegrated variables are considered this is commonly achieved by a grid search. However, grid search quickly becomes computationally unfeasible if more than two variables are cointegrated. Therefore, the likelihood function has to be maximized using heuristic approaches. Depending on the precise problem structure the evolutionary approach developed in the present paper for this purpose saves 90 to 99 per cent of the computation time of a grid search.

Keywords: Evolutionary Strategy, Genetic Algorithm, TVECM

JEL classification: C61, C32

* The author is indebted to Herbert Buscher, Christian Schmeißer, Sebastian Giesen, Rolf Scheufele and Toralf Pusch for valuable comments and discussions.

Ein evolutionärer Algorithmus zur Schätzung von Threshold-Vektorfehlerkorrekturmodellen

Zusammenfassung

Im vorliegenden Papier wird ein evolutionärer Algorithmus zur Schätzung von Threshold-Vektorfehlerkorrekturmodellen (TVECM) mit mehr als zwei kointegrierten Variablen entwickelt. Da die fehlende Berücksichtigung eines Schwellenwerts, bei dem sich die Anpassung an das langfristige Gleichgewicht verändert, in einem Kointegrationsmodell dazu führt, dass konventionelle Schätzer nicht länger effizient sind, muss dieser Schwellenwert simultan mit dem Kointegrationsvektor geschätzt werden. Solange nur zwei kointegrierte Variablen betrachtet werden, wird die Schätzung üblicherweise mittels einer Rastersuche vorgenommen. Eine solche Rastersuche ist allerdings, wenn mehr als zwei Variablen kointegriert sind, aufgrund des immensen Rechenaufwands meist undurchführbar. Die Likelihood-Funktion muss daher über heuristische Verfahren maximiert werden. Abhängig von der genauen Problemstruktur kann der zu diesem Zweck im vorliegenden Papier vorgeschlagene Algorithmus 90 bis 99 Prozent der Rechnerkapazität, die für eine Rastersuche notwendig wäre, sparen.

Schlagworte: Evolutionäre Strategie, Genetischer Algorithmus, Vektorfehlerkorrekturmodelle

JEL-Klassifikation: C61, C32

1 Introduction

In the present paper we develop an evolutionary algorithm to estimate Threshold Vector Error Correction Models (TVECM) with more than two cointegrated variables and an unknown cointegration vector.

Since the seminal contribution of Balke & Fomby (1997) about cointegration relationships where error correction only occurs if a certain variable - mostly the error correction term or its absolute - exceed a certain threshold, the analyzis of threshold cointegration became a standard tool in applied economics.

A major problem of TVECM is that the long term relation, i.e. the cointegration vector, is not estimated efficiently by conventional estimators if the threshold is not adequately taken into account. Therefore, the cointegration vector and the threshold itself have to be estimated simultaneously (Hansen & Seo 2002). Hansen & Seo (2002) who focus on two variable cointegration relationships employ a simple grid search to find the optimum of the likelihood function. This approach is still clearly prevalent. While a grid search is feasible if no more than two variables are cointegrated and the problem is thus limited to two unknown parameters[1], it is no longer computationally feasible if more variables are considered. Gascoigne (2004) recommends a "sequentially modified grid search" that combines a limited grid search with hill climbing techniques. However, this procedure is neither saving sufficient computational time nor does it evade local optima with sufficient probability. Hansen and Seo themselves recommend the genetic algorithm of Dorsey & Mayer (1995) for problems with more than two variables.

Genetic algorithms - or rather evolutionary algorithms in general - attempt to find an optimum parameter combination for a given problem by refining a set of solution attempts consecutively by simulating an evolutionary process consisting of (fitness based) selection of promising candidates, recombination of the solution components and mutation, until convergence to one solution is achieved. This procedure ought to save a substantial proportion of the computational requirements of a full grid search, while being less prone to convergence in local optima than conventional heuristics like hill climbing algorithms. However, our results show that this simple evolutionary technique is not suited to asure robust convergence close to the global optimum of

[1] Since the cointegration vector is normalized only one component of the cointegration vector and the threshold have to be searched if two variables are cointegrated.

the likelihood function of a TVECM.[2] Likelihood functions of complex econometric models often exhibit features that make it difficult for simple evolutionary heuristics to track down the global optimum. Thus, there are only few attempts to use them in maximum likelihood (ML) estimation.[3] In the present paper we show that a properly adapted evolutionary algorithm can robustly find the global maximum of a Gaussian likelihood function while saving 90 percent and more computational time.

The paper contributes to the literature in three ways: First, we develop a new selection mechanism that guarantees population diversity and thus avoids local optima more easily. Second, we use a structured approach to mutation based on a principal component analysis of the genome of well performing individuals in the tested population. Third, we combine these new techniques with modern concepts from the evolutionary optimization literature to develop an evolutionary algorithm that is well suited for ML estimation. While we focus on a TVECM in the present paper the algorithm we propose is applicable to most econometric models where nonlinearities of some kind cause a rough surface of the likelihood function.

The remainder of the paper is structured as follows: Section 2 briefly introduces evolutionary algorithms. Section 3 describes the structure of TVECM and their encoding in chromosome form. Section 4 presents the evolutionary algorithm we develop.

2 Evolutionary Algorithms

Evolutionary algorithms go back to the works of Holland (1975)[4] on genetic algorithms and Rechenberg (1973) on evolution strategies.

[2] Dorsey and Mayer only discuss genetic algorithm for economic optimization problems in general and do not explicitly recommend their algorithm for this special purpose.

[3] Examples include Czarnitzki & Doherr (2002) and (more recently) Öztürkler & Alaten (2008).

[4] Holland actually presented some contributions on genetic algorithms earlier, but this work is usually consireded to be the origin of genetic algorithms since it includes the schema theorem that he uses to show the efficiency of his algorithm design.

These mechanisms [5] attempt to maximize an *evaluation function* by simulating the process of biological evolution. Starting with a *population* of randomly created solution candidates (*individuals*) the quality of the population is stepwisely improved by the mechanisms of selection, *recombination* (often referred to as *crossover*) and mutation:

- In the selection phase auspicious solution candidates are chosen to inherit their "genetically" coded information to an offspring generation. The probability to create offspring is based on the fitness, which is usually defined as a function that is strictly monotonous in the evaluation function.

- In the crossover phase components of randomly selected pairs of individuals from the parent population are recombined to create new individuals that replace their parents.

- In the mutation phase randomly chosen individuals from the resulting population are chosen and (randomly) altered slightly. While this might cause a certain loss of already accumulated information about the "correct" solution, it assures a thorough search over the space of possible solutions.

While the various evolutionary techniques all share these common principles, they differ heavily in the details. Selection method (most importantly the rigidity of the selection mechanism), mutation and crossover operators and probabilities have to be adapted to the problem of interest. Essentially, most of these design decisions boil down to the core trade-off, that has been mentionend in the context of mutation, between a broad randomly driven search that allows to avoid local optima and the use of accumulated information about clusters of good solution candidates.

[5] In computer science and engeneering evolutionary algorithms are traditionally quite rigidly classified in *genetic algorithms* and *evolution strategies*. For a detailed comparison see Hoffmeister & Bäck (1990). The most notable difference between these is the chromosome encoding of solutions. Genetic algorithms closely model biological chromosomes, where every gene has only a limited number of possible values, and mostly use a bitstring encoded chromosomes. In evolution strategies the genes traditionally are real numbers. However, the term "genetic algorithm" is used more general in economics and will accordingly be used as a synonym for evolutionary algorithm in the remainder of this paper. Anyhow, the concept of "real valued" genetic algorithms is not entirely unknown (see e.g. Wright (1991)), since the key difference between genetic algorithms and evolution strategies is not uncontroversiaal. Furthermore, recent developments made the distinction difficult due to the increasing mutual adoption of various techniques.

3 TVECM and their chromosome encoding

We consider the following type of TVECM :
x_t is a p-dimensinal vector from the nonstationary ($I(1)$) time series x of length T. x_t is cointegrated of rank 1, i.e. there is a β where $ec_t = \beta' x_t$ is stationary. The degree of error correction, i.e. the impact of ec_{t-1} on Δx_t depends on whether a threshold variable θ_t exceeds the threshold $\tilde{\theta}$, so that:

$$\Delta x_t = \begin{cases} \alpha_1 ec_{t+1} + \gamma_1(L)\Delta x_t + \varepsilon_t & \forall \theta \mid \theta \leq \tilde{\theta} \\ \alpha_2 ec_{t+1} + \gamma_2(L)\Delta x_t + \varepsilon_t & \forall \theta \mid \theta > \tilde{\theta} \end{cases} | \alpha_1 \neq \alpha_2 \tag{1}$$

Following Hansen & Seo (2002) this specification does not only allow the degree of error correction to change between regimes, but also accounts for possible changes of the short run correlation, i.e. the possibility that $\gamma_1 \neq \gamma_2$. Since the difference in α is the essential one, because a model with $\gamma_1 \neq \gamma_2$ and $\alpha_1 = \alpha_2$ can be estimated conventionally, this is often neglected in the specifications found in the literature.[6] However, the possibility that the short run relations change when the error correction changes can rarely be ruled out.
Since we can normalize the cointegration vector β so that $\beta_1 = 1$ and α_1,α_2, γ_1 and γ_2 are set deterministically by the choice of β the genetic algorithm has to identify the choices for $\beta_i | i = 2..p$ and $\tilde{\theta}$ that maximize a Gaussian likelihood function.[7] Since the most common threshold variable is the error correction term ec which is not known ex ante but depends on β, we do not include the threshold in absolute terms in the chromosomes but rather the relevant quantile q_θ of the variable of interest. This guarantees that any allele (i.e. a specific gene value) of the "threshold gene" can still be reasonably interpreted when β changes due to mutation or recombination. Since the estimate of β that is obtained if the threshold is ignored gives a reasonable idea about β despite its inefficiency, we restrict the search space to the proximity of this original estimate.
This leads to chromosomes of the form:

[6] See Krishnakumar & Neto (2009) and Gonzalo & Pitarakis (2005)

[7] Potentially, the lag order l could be included if the evaluation function is a properly chosen information criterion. However, this cannot be recommended. Ng & Perron (2001) and Qu & Perron (2007) show that commonly used information criteria as the AIC are strongly biased in Vector Error Correction models. They propose a modified AIC (MAIC) to correct for this bias. However, the MAIC relies on the knowledge of the likelihood ratio test of the cointegration rank which is not known before estimation.

$$c = \begin{bmatrix} q'_\theta \\ \beta'_2 \\ \beta'_3 \\ \vdots \\ \beta'_p \end{bmatrix}^T ,$$

with

$$q'_\theta \epsilon [\pi, (1-\pi)] | 0 < \pi < 0.5$$

$$\beta'_i \epsilon \mathbb{R}, \hat{\beta}_i - 3 * \hat{\sigma}_{\beta_i} \leq \beta'_i \leq \hat{\beta}_i + 3 * \hat{\sigma}_{\beta_i},$$

where the apostrophe denotes components of solution candidates and the hat denotes estimation results from the initial analysis that ignores the threshold effect. The parameter π guarantees that the algorithm considers only thresholds that divide the sample in two subsamples of suffucient size for reasonable statistical inference.

4 The evolutionary algorithm

4.1 Selection

In the selection phase of a genetic algorithm a a parent population that is subsequently used to create the offspring generation is chosen. Since we only consider mechanisms, where the parent population that is generated by sampling mechanims with replacement is of the same size as the original population, this parent population actually is a preliminary offspring generation.. This preliminary offspring generation consists of excact copies of well performing individuals of the original population.

The selection mechanism is the driving force behind any evolutionary approach. It creates the pressure that is necessary to ensure that the improvements created by random mutation are actually realized. It is evident that a strong selection leads to faster convergence. However, speed comes at a cost: strict selection limits population diversity and thus increases the risk of premature convergence in local optima.

This trade-off turns out to be especially tricky in the case of Gaussian likelihood functions of TVECM. These functions exhibit very rough surfaces with a large number of

local optima. This is usually taken into account by choosing a selection mechanism with low "pressure index". However, we often find that the global optimum exceeds local optima only slightly, a problem that is commonly solved by choosing a high pressure mechanism. Accordingly, the construction of an efficient selection mechanism requires a lot of fine tuning that allows a selective conservation of population diversity while generally working at a high level of pressure.
We compare three established selection mechanisms for this purpose: the common probabilistic "roulette wheel with replacement", a remainder stochastic sampling without replacement and a binary selection tournament. However, our main contribution is a population density adjustment of the fitness function that drastically decreases the risk of premature convergence at an almost neglectibly small cost in terms of computational time.
In this section we only present some general results on the various selection schemes, since their efficiency is strongly interacting with other design choices that are discussed in later sections. Thus, a full test of several promising combinations of selection, crossover and mutation will not be done before section 5.

4.1.1 Selection Mechanism

Roulette Wheel The traditional selection mechanism is a simple "roulette wheel" procedure. Sections of a roulette wheel are assigned to members of the parent popuation proportional to their fitness. Every member of the preliminary offspring generation is then selected by a separate spin of the wheel. With f denoting the fitness function and n denoting the population size the probability of selection $p_s$of individual x_i thus is given by:

$$p_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)}, i = 1..n.$$

While probabilities of contributing the genes to subsequent generations are proportional to the fitness with this method, results vary strongly.

Binary tournament In a binary tournament (BT) two randomly chosen individuals are draw from the parent generation. Each individual has the same probability to be selected for competition in the tournament. Only the better of the two individuals that are matched enters the preliminary offspring generation. This prodedure is repeated until the new generation is complete. Binary tournaments have been found

to be highly effective and improve the speed of convergence substantially (Goldberg & Deb 1991).
Contrary to other approaches the chance to contribute to future generations is not based on the fitness function directly, but on the corresponding percentile. Since the chance of an individual x_i to win the tournament does approximately equal the quantile of the corresponding fitness value in the current distribution of fitness values, it can be said for large population sizes that:

$$p_s(x_i) \approx \frac{2}{n} q_{f(x_i)}, i = 1..n,$$

where $q_{f(x_i)}$denotes the quantile of the fitness of x_i.

Remainder Stochastic Sampling without replacement Remainder Stochastic Sampling (RSS) is a variation of the roulette wheel approach that maintains its probabilites of creating offspring but produces much more stable results. For each individual the ratio

$$r_i = \frac{f(x_i)}{mean(f(x))}, i = 1..n,$$

is computed. Each individual of the parent population is then placed in the preliminary offspring generation according to the integer part of r. The remaining places in the preliminary offspring generation are then selected randomly from the parent population without replacement, where the selection probabilities equal $(r \, mod \, 1)$ i.e. the fractional part of r. Contrary to binary tournaments remainder stochasic sampling is known to ensure a high degree of population diversity. Contrary to the other mechanisms used, RSS always selects the best individual at least once.
Table 1 summarizes some results on the performance of these three selection schemes. For simplicity we do not maximize an actual function, but instead generate a population with preassigned evaluation values that are uniformly distributed (real numbers) between zero and one as. For this example the fitness function is defined as the difference to the worst individual. To give an idea about the dynamics induced by the selection mechanism each mechanism has been applied 10 times. Note, that this is not full fleged simulation of a genetic algorithm, since the individuals of an offspring generation are neither recombined nor mutated before the selection scheme of interest is applied again. This procedure has been repeated 2000 times for each scheme. We compare the average population mean of the evaluation value (μ_x) af-

Table 1: Selection mechanism performance

	Roulette Wheel		RSS		BT	
	μ_x	di_x	μ_x	di_x	μ_x	di_x
Iteration 1	0.6703	0.7187	0.6701	0.7212	0.6666	0.7082
	(0.245)	(0.0492)	(0.0028)	(0.0001)	(0.0230)	(0.0467)
Iteration 2	0.7663	0.7894	0.7633	0.7930	0.7988	0.8146
	(0.0274)	(0.0811)	(0.0049)	(0.0049)	(0.0260)	(0.0882)
Iteration 3	0.8308	0.8281	0.8254	0.8384	0.8865	0.8881
	(0.0263)	(0.1116)	(0.0059)	(0.0082)	(0.0225)	(0.1463)
Iteration 4	0.8776	0.8568	0.8700	0.8693	0.9376	0.9299
	(0.0242)	(0.1435)	(0.0060)	(0.0121)	(0.0175)	(0.2170)
Iteration 5	0.9111	0.8768	0.9031	0.8947	0.9656	0.9492
	(0.0214)	(0.1834)	(0.0057)	(0.0162)	(0.0133)	(0.3201)
Standard deviation given in paranthesis diversity index:$di_x = \frac{\sigma_x}{1-\mu_x}$						

ter a given number of iterations, the average population diversity (indicated by the diversity index of the population members that is explained in the table) after that number of iterations and the standard deviation of these indicators.

The less "bad" individuals are selected for the parent generation, the higher is the speed of convergence to one or - if the best indivdual is deleted in at some generation - close to some value close to one.. This essentially reflects the pressure produced by the selection mechanism. A high diversity index indicates a diverse population, that is necessary to mitigate the risk of premature convergence in local optima in more complex settings. The standard deviations of these values describe how stable the mechanism produces its "standard" result.

While the roulette wheel and RSS are almost identical on average in terms of convergence speed (as indicated by the development of the population means) and population diversity, the results produced by RSS are substantially more stable as predicted. Also, the high convergence speed of BT can be seen.

4.1.2 Evaluation Function

Squared diffence to worst vs. difference to worst The most common fitness function is the difference of the evaluation function to the worst evaluation function value of an individual in the present population, i.e.:

$$f(x_i) = e(x_i) - e_{min}(x),$$

where e is the evaluation function.
However, the likelihood function of a TVECM has some features which cause problems if this fitness function is used:
While the slope of the fitness function is often very flat around the global optimum in one dimension, it is very steep if following other dimensions. Even if mutation is carefully adapted during runtime of the algorithm, this usually leads to some very badly performing individuals in the population after mutation. Since these badly performing individuals define the new fitness benchmark, the relatively small differences in the top region of the population barely make a difference.[8]
Thus, we try to enhance the impact of differences in the better region of the population on selection probabilities by using the squared difference to the worst performing individual:[9]

$$f(x_i) = (e(x_i) - e_{min}(x))^2.$$

This strongly favors the top performers, while still allowing for some randomness. Especially if combined with selection mechanism that favor population diversity (like RSS) these fitness functions grant the necessary boost to the selection chances of the best individuals while at the same time being sufficiently diverse to prevent frequent premature convergence.

Dynamic offset We implement several adjustments to the fitness function to prevent premature convergence. One of them is a dynamic offset mechanism that reacts to a sudden decline in population diversity. As noted by Czarnitzki & Doherr (2002) an offset $\psi|\psi > 0$ can be added to the fitness function of all individuals to increase the selection probabilities of badly performing individuals and by that to increase the population diversity of following generations. Since an offset can strongly reduce convergence speed, we recommend to limit the use of a substantial offset to situations where the danger of premature convergence due to a sudden

[8] Obviously this argument is not applicable to tournament selection schemes that operate entirely based on the quantile of the fitness that is independent of any strictly monotonous transformations.
[9] This does not affect tournament selection.

decline in population diversity is imminent. Instead of relying on a fixed offset ψ we thus propose a fitness functions of the form:

$$f(x_i^k) = (e(x_i^k) - min[e_{min}(x^k), \frac{1}{z}\sum_{j=1}^{z} e_{min}(x^{k-j})])^m | m\epsilon[1,2],$$

where $e_{min}(x^k)|k\epsilon\mathbb{N}^+$ denotes the minimum of the population x in generation k. If the current minimum exceeds the rolling average of past minima, the latter is substracted instead of the first. Essentially this equals the addition of an offset, if the minimum evaluation value drastically increases from one generation to the next.

Density Adjustment The prime contribution of this paper to selection schemes that are suited for typical estimation problems in econometrics is the introduction of a density adjustment.

Due to the vast size of the search space in a genetic TVECM estimation and real valued problems in general, even a large population cannot guarantee that all clusters of "good" solutions are well covered by the population. Since the probability that the peek (or a point close to the peek) of an elevation in the evaluation function surface has been hit is lower if this specific elevation has been tested less thoroughly, this does induce a high risk of premature convergence.

To compensate for this we introduce a density adjustment to the fitness function that favors more insulated individuals in early stages of the algorithm.[10] A density adjustment factor ϕ is added to the fitness of individuals that already achieve an unadjusted fitness above the median fitness. With η_i denoting the density index around individual i, and d_{ij} denoting the standardized Euclidean distance between x_i and x_j, ϕ is set to

[10] Potts, Giddens & Yadav (1994) follow a similar idea in their "migration" based algorithm. However, they attempt to preserve several populations throughout runtime and thus have substantially higher loss of "evolutionary information" resulting in higher runtimes. This extreme procedure does not seem necessary for TVECM applications according to our experience. Partly, the different approaches are due to the different optimization philosophies: In engeneering and computer science the objective usually is to improve some technique while a true global optimum does not necessarily exist. Even if a global optimum does exist, finding an excellent solution with a parameter combination far from this optimum is clearly preferred to a less fit solution that is close to the global optimum. In econometrics, however, the final objective is not to find a parameter combination with a fitness that is close to the fitness maximum, but to find a parameter combination that is close to the parameter combination that has maximum fitness.

$$\phi_i = \begin{cases} (1-\eta_i)(f_{max}(x) - f(x_i)) & \forall i | f(x_i) \geq q_{.5}(f) \\ 0 & \forall i | f(x_i) < q_{.5}(f) \end{cases},$$

where

$$\eta_i = \frac{|\{x_j : d_{ij} < q_{.1}(d)\}|}{n-1}, j \epsilon [1..n] \backslash i.$$

That is, the density index of an individual x_i is defined to be the share of other individuals of the same population which are closer to x_i than the distance given by the 10^{th} percentile of the distribution of bilateral distances in the entire population. It has to be kept in mind that population density is not exogenous. The higher the generation number, the more the current population density at different points in the search space is driven by the fitness in these regions. Thus, the density adjustment is only applied in early stages of the algorithm. Since the number of generations until convergence differs heavily dependent on the basic selection mechanism, crossover operators and mutation schemes, the number of adjusted generations is selected based on the average runtime of the respective algorithm without adjustment. Detailed information is found in section 5 where full algorithms are compared.

4.2 Recombination

4.2.1 The recombination operator

In the second stage of a genetic algorithm pairs of chromosomes that have been selected in the selection phase are chosen for recombination:
Commonly recombination in genetic algorithms is achieved by exchanging genes between two parent chromosomes. However, it has been shown by Wright (1991) that this procedure is not in line with the schema theorem for real coded genes. Therefore, we use an alternative recombination operator:
The parent chromosome with the higher fitness remains unchanged. The parent chromosome with lower fitness is replaced by a chromosome where each gene is the mean of the parents' respective genes.[11]
Replacing the worse parent with the average of both parents creates a "gravity" towards already known clusters of good solutions. Similar gravitational effects are

[11] Averaging is commonly also found in evolutionary strategies.

part of the "differential" mutation schemes frequently used for real valued genetic algorithms (see Hrstka & Kučerová (2004) for a detailed discussion). These mutations schemes commonly include a deterministic component of mutation that causes a movement to the best parameter combination that is presently known. The major advantage of our approach, that incorporates this gravitational pull into crossover, is that it is less detrimental for population diversity since the gravitation is evenly distributed over regions with good results instead of being concentrated in one point.

4.2.2 The recombination rate

All algorithms that we use run with a recombination rate of 0.5. On a first glance, this might seem unusually high. However, while both parents are changed by the usual crossover operators, only one parent is changed by the crossover operator used herein. Thus, the actual rate of change associated with a certain recombination probability is halved compared to conventional genetic algorithms. Correspondingly, the mechanisms designed in this paper have to work with a high recombination probability.

Although it has been shown that the adaptation of the crossover rate during runtime is mostly superior to a fixed recombination rate (Bäck & Hoffmeister 1994, Bäck 1992),[12] the present algorithms use a simple fixed rate approach. A full fledged self adaptiation[13] would require an additional evaluation step following crossover. However, contrary to many other applications evaluation is by far the most computation time consuming part of genetic TVECM estimation. Since crossover has been found to be of limited importance for real valued problems (Spears 1995), an additional evaluation seem inappropriate.

[12] Thierens (2002) notes that fixed rates are nevertheless convenient for many problems (and thus dominant in practical applications) since they are not only more easily implemented but also mostly find the solution after a higher but still acceptable number of generations.

[13] We follow the definition of self-adaptation by Angeline (2002) who defines self adaptive algorithms as algorithms where the choice of the new mutation or crossover rate is based on the empirical comparison of **different** rates over one or several generations. Contrarily, adaptive algorithms use a fixed rule to change the respective rates based on the evaluation of the current success of crossover or mutation. While both evaluate the success in preceeding generations, they differ in so far as only self adaptation evaluatuates several potential rates to make the choice for coming generations.

Some experiments we run with self adaptive recombination rates (that are presented in the section on mutation) using the noisy evaluation signal aquired after the subsequent mutation phase, do not outperform fixed rate algorithms.

4.3 Mutation

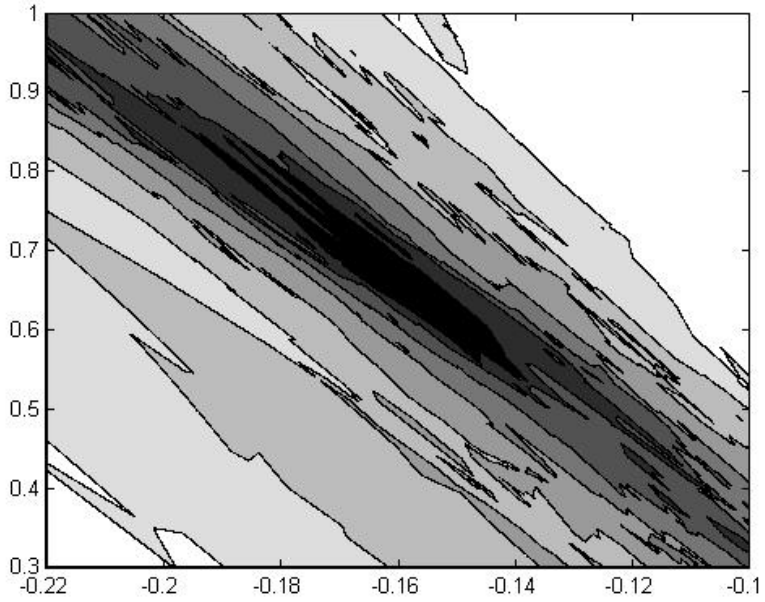
4.3.1 Dimensional Adjustment of the search space

We find that gene-by-gene mutation operators do not work well with genetic TVECM estimation. Due to mismatching structures in the likelihood function and the mutation pattern it is virtually impossible in the later stages of the algorithm that mutation leads to a better solution. This fosters premature convergence in the best points of early stages of the population.

To visialize this Figure 1 on page 18 shows the small section of a likelihood functions of one of the simulated three-variable-TVECM that have been used to test the algorithms designed in this paper. The axis show the second and third component of the cointegration vector β, the threshold held constant. Darker shades indicate higher loglikelihood. The structure in the "good" solution canditates is clearly visible. While this is partly due to the choice of scales on the axis that exaggerate this structure slightly, this general pattern can be found very frequently. The reason for this behavior is that the cointegration vector is not unique. Any multiple of a valid cointegration vector is a valid cointegration vector itself. Only by normalizing the first component the cointegration vector is made "unique". Thus, if all components of β except the normalized one (that is not part of the genetic estimation) change consistently with each other, only the first component truly deviates from an optimum solution. Therefore, the surface of the likelihood function often has the form of a long mountain range with an especially high peak where the components of β are in line with the initially chosen normalization.

A genetic algorithm will quickly evolve to populations along this mountain range. However, standard mutation operators will systematically cause mutations to "fall over the edge". Figure 2 illustrates the mutation pattern of gene-wise mutation using a (relatively high) mutation rate of 20 percent. It is evident that mutation will systematically favor mutations that are not in line with the structure found in the likelihood function. Thus, quick evolution is almost impossible. Especially if the selection pressure is high, this will lead to convergence in local optima.

Figure 1: Contourplot of a section of a TVECM likelihood function



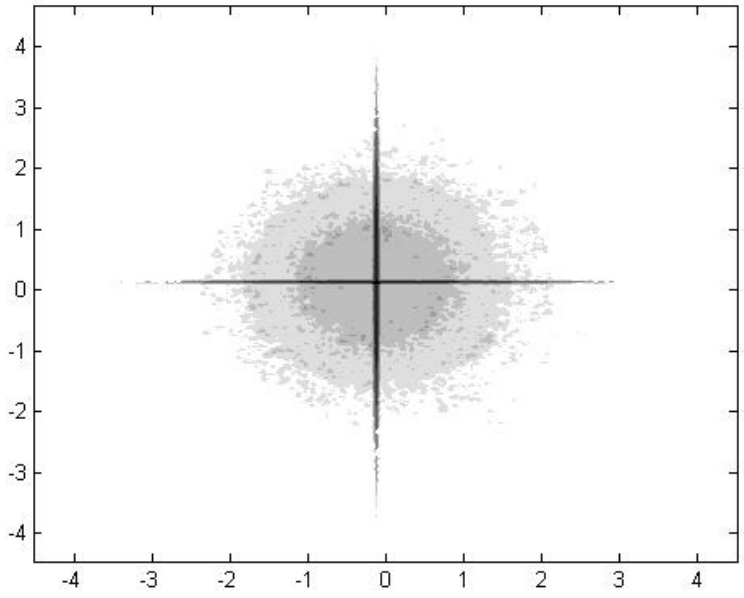
Partially, this problem can be solved by mutating all genes jointly. However, the algorithm can be improved substantially more by including structural information in the mutation process. We use a principal components analysis of the population to determine in how far changes in the different parameters are coaligned. The mutation operator is not applied to a chromosome, but instead to its counterpart from a factorized population:

Let P_j denote the population in generation j. c_j^m is the m^{th} chromosome of this population and c_{ij}^m the i^{th} gene on this chromosome. Since the distribution of individuals is entirely random in the beginning and mostly random in the first generations the algorithm uses a standard mutation approach during the first generations. Thus, if $j \leq \tilde{j}$, the mutated gene i of the chromosome c_{ij}^m in generation j is given by:

$$c_{ij}^{m*} = \begin{cases} c_{ij}^m + v_{ij}\omega_{ij}^m & \forall \rho_{ij}^m < p_M(i,j) \\ c_{ij}^m & \forall \rho_{ij}^m \geq p_M(i,j) \end{cases}, \omega_{ij} \sim N(0, \sigma_{c_{ij}}), \rho_{ij}^m \sim U(0,1),$$

where $p_M(i,j)$ is the mutation probability of gene i of the chromosome, $v_{ij}$is a variation adjustment parameter and $\tilde{j}$ is the number of "warmup" generations that are used to identify the structure of good solution candidates.

Figure 2: Pattern of mutation given a 0.2 gene-wise mutation probability



A parameter combination (0,0) is mutated by adding a number randomly drawn from a standardized normal distribution to a mutating gene.
Each gene mutates with 20 percent probability, where mutation of the genes is independent.
Darker shades indicate a higher likeliness of this mutation result.
The distribution in the figure is not theoretically derived but generated from one million mutations (i.e. mutation attempts where at least one gene did actually change).

If $j > \tilde{j}$ we consider a factorized population P_j^f. $P_j^f = (P_j - \mu_j^P) * F_j$, where F_j is the matrix of factor loadings derived by a principal components analysis of the population in generation j and μ_j^P is a matrix of the dimensions of P where every row of the μ_j^P is the current average individual μ_j^c. Let $(c_j^m)^f$denote the redimensioned chromosome c_j^m, i.e. $(c_j^m)^f = (c_j^m - \mu_j^c) * F_j$. Mutation now takes place according to the formula:

$$(c_{ij}^m)^{f*} = \begin{cases} (c_{ij}^m)^f + v_{ij}\omega_{ij}^m & \forall \rho_{ij}^m < p_M(i,j) \\ (c_{ij}^m)^f & \forall \rho_{ij}^m \geq p_M(i,j) \end{cases}, \omega_{ij} \sim N(0, \sigma_{c_{ij}^f}), \rho_{ij}^m \sim U(0,1),$$

and

$$c_{ij}^{m*} = \mu_j^c + (c_{ij}^m)^{f*} * F^T.$$

This procedure guarantees that mutation can be adjusted to follow the most crucial correlations found in the selected individuals. However, since no factors are "selected" as in the traditional principal components analysis, mutation along all dimensions is possible. The key difference is, that mutation can by scaled according to the patterns that are actually found in the data.[14]
We find that the application of this alternative mutation operator cuts convergence time roughly into half for most algorithm designs while at the same time improving the chance of convergence close to the global optimum. Details are presented in section 5.

4.3.2 Mutation rate and variation adjustment

It has been frequently shown that adapting mutation intensity during runtime improves the performance of genetic algorithms substantially.
Most of the literature concentrates on the aspect of the mutation rate when mutation intensity is analyzed. This is appropriate for standard genetic algorithms with long binary string chromosomes. In these setups distance between chromosomes is defined by the number of differing genes. Thus, mutation probabiliy is not only measuring whether a chromosome will change, but first of all how far the mutated chromosome will be from its progenitor. However, this is not applicable to the real valued domain, where chromosome distance is mostly determined by the distance of individual genes. Thus, we do not only apply adaptation mechanisms to the mutation rate, but also to the variation adjustment parameter v.
We test slightly changed versions of two self adatation mechanisms known from the literature: the self adaptive genetic algorithm (SAGA) developed by (Hinterding, Michalewicz & Peachey 1996) and the probabilistic rule-based adaptive model (PRAM) by Ho, Lee & Leung (1999). Both algorithms were originally designed as population level adjustment mechanisms. Since it has recently been found that

[14] One technical note has to be made. The algorithms used in this paper are run in MatLab. It is quite difficult to rescale the frontiers of the search space according to the new dimensions. Theoretically it would be best to repeat mutation until legitimate mutation results are achieved. However, this would enforce element by element mutation. This element by element implementation does indeed save generations but it increases the computation time for any given generation with MatLab substantially, since it does not make use of MatLabs matrix based architecture. Thus we simply adjust illegal mutations according to fit the violated border of the search space. This creates very few mutations that do not follow the intended mutation logic.

gene level adjustment generally outperforms population level adjustment (Korejo, Yang & Li 2009) both are implemented on gene level herein.

PRAM In the PRAM the population is split in three more or less equally sized subpopulations that use slightly differing values of the control parameter of interest (i.e. mutation rate or variation adjustment parameter). After each generation the success of each rate is evaluated. If the highest rate outperformed the others - and the rate has not yet reached a predefined maximum - all three rates are increased for subsequent generations. Analogue, all rates are lowered for subsequent generations if the lowest rate outperformed the alternatives. If the middle rate is best, the rates are left unchanged.
The advandate of PRAM is that - albeit several rates are simultaneously tested - the rates are close enough to assure that most of the population is subject to reasonable control parameters.

SAGA Like PRAM, SAGA uses three (or more) subpopulations. Each subpopulation uses a different value of the control parameter of interest. However, opposed to PRAM, the subpopulations differ in size and the chosen parameter values have to differ substantially. Instead of adjusting the parameter choices themselves, the sizes of the subpopulations are adjusted after each generation. The number of individuals which are subject to the most successfull parameter choice of the previous generation is increased, while the number of individuals that are treated with the worst performing parameter choice of the previous generation is decreased by the same amount. If a small adjustment of a parameter is no improvement but a larger change is, this can be recognized by SAGA while PRAM is oblivious to the necessity of large changes. However, this comes at a cost: First, SAGA is much less flexible since the parameter choices are limited by the initial decision. Second, for SAGA to work properly, a substantial part of the population has to be treated with parameter choices that are clearly inefficient at the current stage of the algorithm. The first disadvantage is partially offset in our approach, since we index variation intensity on the current distribution of the population and create and additional dynamic by this.
For the mutation rate itself it has been shown that PRAM outperforms SAGA. However, the advantage of the SAGA to allow for larger jumps might be more important if applied to the size of the mutation. Thus we concentrate on PRAM to

adapt the mutation rate, but include both SAGA and PRAM in our analysis of the mutation size adaptation.

5 Comparing algorithm designs

It turns out that design choices partially interact heavily, causing control parameter choices that perform good in one setup to be detrimental or at least less helpfull in others. Since the number of combinations is quite large we ran "pretest" with few repetitions on all possible combinations and only pursue to full fleged tests for promising candidates.

The tests are run based on two test functions derived from simulated TVECM with different identification problems. Both simulated TVECM include three variables that are cointegrated of rank 1 with a threshold effect in the error correction term. The details concerning the specifications of these are found in the appendix. Both functions have been thoroughly tested using a grid search. This is necessary since we are mostly interested in the maximizing properties of the algorithm, i.e. we are not looking for the true parameters but for the parameters that maximize the likelihood function. While the maximum likelihood function finds the true parameters on average, if the econometric model is well specified, this is obviously not true for any singe case with its specific distribution of error terms.

As mentioned before, finding parameter estimates close to the maximum of the likelihood function is favorable compared to finding strongly deviating parameter estimates that have a similarily high likelihood value. Thus we employ an indicator that controls for proximity to the best solution: To be considered as a "hit" a run of the algorithm has to fulfill two conditions: The estimation of the threshold itself must be precise enough that at most one observation in the simulated time series is classified into the wrong regime. Furthermore, each parameter estimate must deviate less than two percent from the "true" parameter.[15]

[15] While we know the true parameters from the simulation these are not necessarily those that maximize the likelihood function. Since the task of the genetic algorithm is to find the maximum of the likelihood function of the TVECM we take the true parameters from the maximum likelihood estimation. Our reference value for the "true" parameters is the optimum parameter set that is found by a tight grid search or the best result of any genetic algorithm applied, if a genetic algorithm finds a superior estimation between the knots of the grid we use.

Contrary to the comparison of achieved likelihood values this quality indicator does actually measure estimation performance.
Since there is a particularly strong tradeoff between accuracy (i.e. the probability that the algorithm produces a "hit") and runtime. To allow for a reasonable comparison of the mechanisms that are accurate but slow and the mechanism that are less accurate but substantially faster we construct an "accuracy adjusted runtime " (AAR) as additional measure, where:

$$AAR_i = \bar{g}(i) * \frac{ln(p_{min})}{ln(p(i))},$$

where $\bar{g}$ is the average runtime of the algorithm in generations, $p(i)$ is the hit ratio of algorithm i, and p_{min} is the required probability to have found a solution that is set to 99.9% for our purpose. AAR_i thus is the total number of generations that algorithm i needs on average, if the algorithm is applied sufficiently often to find the correct maximum with the minimum probability p_{min}.
All computations were done with MatLab (without toolboxes).

5.1 Screening results

These first tests include the following control six parameter choices leading to 144 possible:

- The fitness function can be based on linear differences or squared differences to the benchmark.
- Selection method can be roulette wheel, RSS or BT.
- Mutation size adaptation be be done by PRAM or SAGA.
- Densitity adjustment is or is not applied.
- Factorization of the search space before mutation is or is not done.
- The polulation size is set to 100, 200, or 300.

According to the results the squared fitness functions outperforms the linear fitness function by far, independent of the general parameter setup. Thus the full test is only run with squared fitness function.

Especially roulette wheel selection performs very badly for our kind of problem, being relatively slow and inaccurate at the same time. While RSS is comparably slow, it has a substantially higher probability of finding a solution close to the true maximum of the likelihood function than the alternative designs we took into account. Although BT has the lowest hit ratio, its immense speed outweights these problems clearly in terms of AAR. We do indeed find quite strong evidence, that it might be the computationally efficient solution to take the best solution that is produced by a battery of subsequent attempts to maximize the given likelihood function with a BT based algorithms.[16]
SAGA outperforms the more recently developed PRAM slightly but quite robustly. The "mountain range" like structure of the likelihood function we face, seems to produce substantially less cases of premature convergence if we allow for mutation over different distances. Therefore, only SAGA is included in the full test.
For the three variable case we use as a benchmark 300 is the minimum population size to produce reasonable results. Therefore, the full test is performed with populations of this size.[17]
Additionally, the baseline algorithm as found in Dorsey & Mayer (1995) is included in the screening for both likelihood functions. Since the first findings indicate strongly that this algorithm is no substantial improvement to grid search for our problem it is not included in the full test. However, it has to be emphasized that their algorithm, while having been recommended, has not been constructed for this purpose originally.

5.2 Test of promising candidates

To summarize, the algorithms we test thouroughly share the following key features:

- squared fitness functions
- BT selection mechanism

[16] However, due to the high selection pressure of BT it has to be applied with special care. If the population density is chosen to low for the complexity of the problem, the already low hit ratio drops sharply. While this is mostly mitigated by factorization and density adjustment, we recommend to repeat testing with a larger population if less then four of the 10 runs of a BT algorithm we recommend cluster arround the optimum the algorithm suggests.

[17] The necessity for large populations if real valued problems are considered is well known, see e.g. Whitley (1994).

- PRAM mutation adaptation of the mutation probability
- SAGA mutation adaptation of the mutation size
- averagering based crossover (with a fixed rate)
- the population size is 300.

The resulting core algorithm is combined with the four possible design choices concerning density adjustment of the population, and the refactorization of the search space dimensions that are tested for two simulated TVEM scenarios.
For each scenario each algorithm is used 1000 times with different, randomly generated starting populations.
One of our TVEM-simulations produces a likelihood function with the afore mentioned "mountain range structure" being diagonal to the search space. In the other scenario the mountain range is parallel to one of the main axis of the search space. However, this scenario produces a likelihood functions where "good" results cluster strongly in a region of the mountain range that is far from the global optimum, and the global optimum (albeit being part of the "mountain range" is a relatively) isolated peak. For simplicity we will refer to these scenarios as "diagonal" and "peak" for the remainder of the section.
Table 2 on page 26 summarizes the average hit rate and the average number of required generations for each algorithm for both scenarios.
The diagonal structure of the clusters of good solutions seems to be the harder problem for conventional genetic algorthims. As seen from the table, the performance of the algorithm is considerably improved by the factorization approach. In the "peak" scenario there is only a slight improvement. While the hit ratio increases slightly and the runtime decreases the results on both factors are insignificant.
Unsurprisingly, the picture is the other way round when density adjustment is considered. While density adjustment improves the performance of the baseline algorithm in the diagonal setup, its additional benefits when the (necessary) factorization is also applied are quite small in this scenario. Contrarily, densitiy adjustment can substantially improve the results in the "peak" scenario, raising hit ratios from roughly 60% to abotu 80% and by that saving more than a third of the accuracy adjusted runtime.
Both innovations presented in this paper contribute strongly to the mitigation of the problems caused by the common features of TVEM based likelihood functions.

Table 2: Hitrate and runtime of genetic algorithms

Scenario	Factorization	Density adjustment	Hit ratio	AAR
diagonal	-	-	12.6	2343
diagonal	-	+	21.0	1433
diagonal	+	-	53.0	443
diagonal	+	+	56.8	417
peak	-	-	61.2	405
peak	-	+	81.8	269
peak	+	-	63.60	339
peak	+	+	83.0	207

Since the structure of the likelihood function is commonly unknown, and there does not seem to be a relevant computation cost attached to the use of either mechanism, they can be recommended withouth substantial caveat.

The application of both mechanisms simulateously raises the hit ratio concerning the more difficult optimization scenario from less than 13% to 56%. Finding these hit ratios for very complex TVECM structures (that were actually designed to feature a high degree of complexity) we recommend to run the mechanism 10 times to find the true global optimum with a 99.9 % probability.

6 Conclusion

We show that genetic algorithms perform extremely well in optimizing the likelihood functions of TVEM that are characterized by a "mountain range" like structure and frequent local optima if the right genetic techniques are combined and two major changes are made.

Our essential innovations are the inclusion of a density adjustment that reduces premature convergence and thus allows strong selection that is generally considered to result in fast algorithms. Furthermore, we use a factor model to adapt the dimensions of the search space to the specific structure of the function of interest.

The recommended algorithm consists of these two features, squared fitness functions, binary tournament based selection, an averaging crossover operator, PRAM based adaptation of the mutation rate and SAGA based adaption of the mutation size.
In a three variable scenario with 200 observations per variable these mechanisms score hit ratios of roughly 60% to 80%. Taking the best of ten results generated by the algorithm gives a result in the close proxity of the true maximum with a probability 99.9%. That means that we have to run roughly 120.000 regressions to compute the maximum of the likelihood function. Even if the grid of choice only features 100 possible choices for each coefficient that is estimated and all (reasonable) thresholds are taken into account this results in 1.6 million regressions. However, according to our tests, the genetic algorithm commonly finds a solution that is better than the grid search solution. Finding solution that is as accurate as the genetic algorithm requires (roughly) 200 possible choices per estimate, leading to 6.4 million required regressions. The genetic algorithm thus saves about 98% of the required computation time.

References

References

Angeline, P. J. (2002). Adaptive and Self-Adaptive Evolutionary Computations, *Computational Intelligence* pp. 152–161.

Balke, N. S. & Fomby, T. B. (1997). Threshold cointegration, *International Economic Review* **38**(3): 627–645.

Bäck, T. (1992). Self-Adaptation in Genetic Algorithms, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 263–271.

Bäck, T. & Hoffmeister, F. (1994). Basic Aspects of Evolution Strategies, *Statistics and Computing* **4**(2): 51–63.

Czarnitzki, D. & Doherr, T. (2002). Genetic Algorithms: A Tool for Optimization in Econometrics-Basic Concept and an Example for Empirical Applications, *ZEW Discussion Paper No. 02-41* .

Dorsey, R. E. & Mayer, W. J. (1995). Genetic Algorithms for Estimation Problems with Multiple Optima, Nondifferentiability, and Other Irregular Features, *Journal of Business & Economic Statistics* **13**(1): 53–66.

Gascoigne, J. (2004). Estimating Threshold Vector Error-Correction Models with Multiple Cointegrating Relationships, *Sheffield Economic Research Paper Series no* **13**.

Goldberg, D. E. & Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, *Foundations of Genetic Algorithms* **1**: 69–93.

Gonzalo, J. & Pitarakis, J.-Y. (2005). Threshold Effects In Multivariate Error Correction Models.

Hansen, B. E. & Seo, B. (2002). Testing for Two-Regime Threshold Cointegration in Vector Error-Correction Models, *Journal of Econometrics* **110**(2): 293–318.

Hinterding, R., Michalewicz, Z. & Peachey, T. (1996). Self-Adaptive Genetic Algorithm for Numeric Functions, *Parallel Problem Solving from Nature–PPSN*

IV: International Conference on Evolutionary Computation, the 4th International Conference on Parallel Problem Solving from Nature, Berlin, Germany, September 22-26, 1996: proceedings, Springer Verlag, p. 420.

Ho, C., Lee, K. & Leung, K. (1999). A Genetic Algorithm Based on Mutation and Crossover with Adaptive Probabilities, *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99*.

Hoffmeister, F. & Bäck, T. (1990). Genetic Algorithms and Evolution Strategies-Similarities and Differences, *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, Springer-Verlag London, UK, pp. 455–469.

Holland, J. H. (1975). Adaptation in Artificial and Natural Systems, *Ann Arbor: The University of Michigan Press* .

Hrstka, O. & Kučerová, A. (2004). Improvements of Real Coded Genetic Algorithms Based on Differential Operators Preventing Premature Convergence, *Advances in Engineering Software* **35**(3-4): 237–246.

Korejo, I., Yang, S. & Li, C. (2009). A Comparative Study of Adaptive Mutation Operators for Genetic Algorithms, *MIC 2009: The VIII Metaheuristics International Conference*.

Krishnakumar, J. & Neto, D. (2009). Estimation and Testing for the Cointegration Rank in a Threshold Cointegrated System, *Cahiers du département d'économétrie, Faculté des siences économiques et socliales, Université de Genève* .

Ng, S. & Perron, P. (2001). Lag Length Selection and the Construction of Unit Root Tests with Good Size and Power, *Econometrica* pp. 1519–1554.

Potts, J., Giddens, T. & Yadav, S. (1994). The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection, *IEEE Transactions on Systems, Man and Cybernetics* **24**(1): 73–86.

Qu, Z. & Perron, P. (2007). A Modified Information Criterion for Cointegration Tests Based on a VAR Approximation, *Econometric Theory* **23**(04): 638–685.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*, Frommann-Holzboog Stuttgart.

Spears, W. M. (1995). Adapting Crossover in Evolutionary Algorithms, *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pp. 367–384.

Thierens, D. (2002). Adaptive Mutation Rate Control Schemes in Genetic Algorithms, *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, Vol. 1.

Whitley, D. (1994). A Genetic Algorithm Tutorial, *Statistics and computing* **4**(2): 65–85.

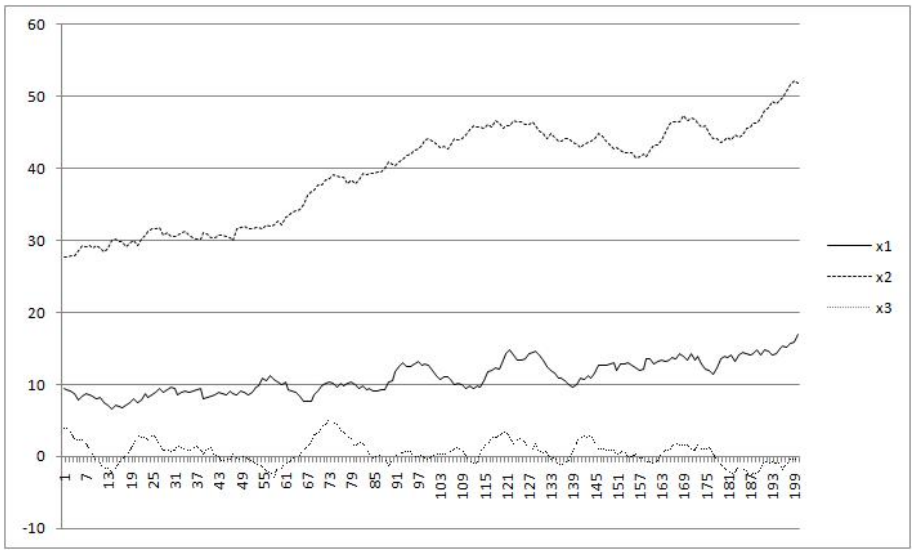
Wright, A. H. (1991). Genetic algorithms for real parameter optimization, *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 205–218.

Öztürkler, H. & Alaten, S. (2008). A Genetic Algorithm Approach To Parameter Estimation In Nonlinear Econometric Models, *The Journal of Social Sciences of Dumlupinar University* **20**: 67–76.

Appendix

"Peak"-Scenario

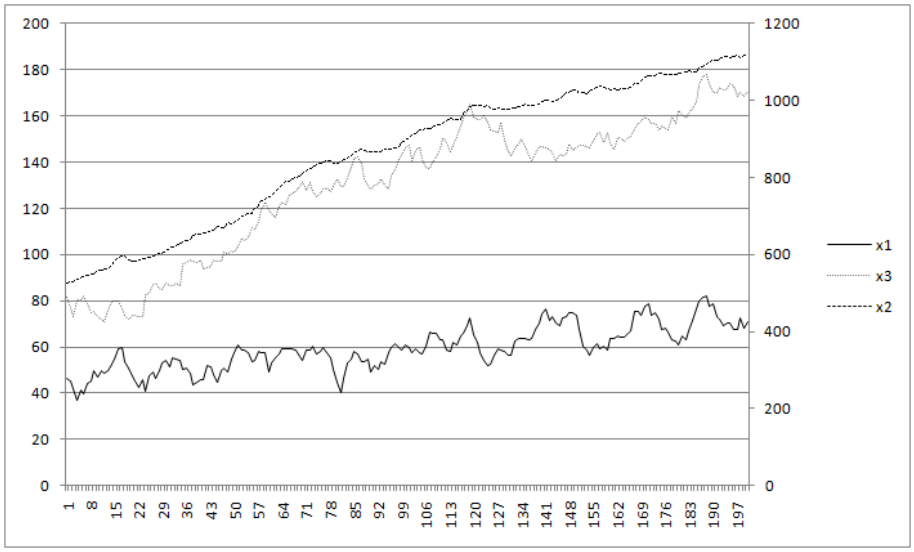
Figure 3: Time series of the "peak scenario"



True parameters: $\beta = [1 - 0.297 + 0.178]'$, $\theta = 0.795$

"Diagonal"-Scenario

Figure 4: Time series of the "peak scenario"



True parameters: $\beta = [1 - 0.194 + 0.905]'$, $\theta = 0.415$

Note: The right hand scale refers to time series x2.