X-OVER PARAMETER CONTROL FOR A GA-OPTIMIZER DEDICATED TO EIGENSTRUCTURE ASSIGNMENT/LQR DESIGNS - PART I - PROBLEM FORMULATION

Celso P. Bottura

bottura@dmcsi.fee.unicamp.br
LCSI/DMCSI/FEEC/UNICAMP -C.Postal 6101 13.083-970 - Campinas - SP - Brazil

João V. da Fonseca Neto

jviana@dee.ufma.br
DEE/CCET/ UFMA - C.Postal 315 - Centro 65.001-970 - São Luís - MA - Brazil

Abstract— The genetic algorithm (GA) initial population or constraints hardness can lead the biased search to be tracked over unfeasible regions of the solution space. An alternative to overcome this undesirable situation is the application of adaptive or deterministic technics to adjust some parameters of a GA-optimizer dedicated to eigenstructure assignment via LQR designs. In this work, it is proposed a method to control parameters of the crossover (X-$OVER$) operation based on the population's average fitness and restrictions satisfaction as a reference to adjust those parameters, in the sense of guiding the search intelligently into feasible solutions. The proposed method is translated into an algorithm and implemented into a multiobjective genetic optimizer decision-making unit. Finally, the proposed adaptive strategy performance is verified into a dynamic systems model. The results of this research are presented in two papers, this paper introduces the problem formulation and a second one deals with computational simulations and result analysis.

Key Words— Genetic algorithm, eigenstructure assignment, parameters tuning , LQR designs, decision-making unit, crossover operator, paralell computation.

1 Introduction

The solution of eigenstructure assignment (EA) problem via linear quadratic regulator (LQR) design faces a great difficulty in finding the Q and R weighting matrices that assigns the desired eigenvalues and eigenvectors. Motivated by this issue, research has been focused in the last three decades to find out these matrices in a deterministic manner for pole placement, (Medanic et al., 1988), (Kawasaki and Shimemura, 1983) and (Graupe, 1972), and for EA, (Harvey and Stein, 1978), (Stein, 1979), (Ochi and Kanai, 1996), (Choi and Seo, 1999b) and (Choi and Seo, 1999a).

In our days, high speed computers emergence lead to the fast development and almost immediate usage of evolutionary computation technics, (Bäck, 1996), such as genetic algorithm, (J.H., 1975) and (Goldberg, 1989), in many fields of human knowledge. Therefore, the appearance of these methods promoted a new way for Q and R matrices search in the nineties, as an alternative to the deterministic manner, for EA via LQR design. This new way is concerned with random search biased technics to find out these matrices, specifically genetic algorithms, (Davis and Clarke, 1995) and (Bottura and Neto, 1999a).

The method developed by (Bottura and Neto, 1999a) has driven into the development of a framework for a EA based on a genetic algorithm (GA) optimizer, that is in charge of the search, and on a decision making unit (DMU), that is in charge of the search guidance. The DMU, based on the multi-armed bandit paradigm, *schema* theorem and preference articulation, had promoted good improvements on the Q and R matrices search, (Bottura and Neto, 1999c). However, much better improvements have been obtained for adjustment on the GA-optimizer crossover operation age parameter before the search cycle activation.

This work presents the development and the performance of two methods that allow the parameter adjustment of the crossover operator parameter, during the weighting matrices search. The development of these methods were stimulated by the EA good results with parameter tuning. The parameter adjustment before the search has been carried out, and at the same time looking for a way to keep the designer as far as possible of realizing inferences on the search cycle. The main benefit of the methods is the search cycle reduction, each step of the search is guided taking into account performance indexes obtained from fitness function.

The paper first part is organized as follows. Section 2 presents the parallel multiobjective genetic algorithm ($PMOGA$) optimizer framework. The fundamentals and the proposed crossover (X-$OVER$) parameter control methods are discussed in section 3. Finally, the concluding remarks are presented in section 4.

[a]The authors are thankful to CENAPAD-SP for the computational resources and to CAPES-PICDT for the financial support.

2 *PMOGA* optimizer framework

The LQR regulator problem formulated as an EA a problem, the multiobjective genetic algorithm $MOGA$ basic elements main features and the distributed parallelization are addressed in this section. A deeper treatment of this issues can be found in (Bottura and Neto, 1999a), (Bottura and Neto, 1999c), (Bottura and Neto, 1999b) and (Bottura and Neto, 2000)

2.1 The EA as a LQR formulation

The main reason to assign the dynamic system, Eq.1, eigenvalues and eigenvectors via the LQR state feedback design is because this method provides a closed loop system with guaranteed robustness for the gain and the phase margins, (Dorato et al., 1995).

$$\dot{x} = Ax + Bu \tag{1}$$

where x is the state variables vector of $n \times 1$ order , u is the control law vector of $m \times n$ order, A represents the system dynamic matrix of $n \times n$ order and B represents the control matrix of $n \times m$ order.

The LQR design to assign the eigenstructure can be seen as three step process: optimal state feedback control law determination, closed loop eigenstructure calculation and calculated eigenvalues and eigenvectors verification.

In the first step, the LQR design problem formulation allows the determination of a control law $u = Kx$, whose gains $-K(Q,R)$ are given by the algebraic Riccati equation, that minimizes the quadratic cost function, $\int_0^\infty [x^T Q x + u^T R u] dt$, subject to the dynamic system, $\dot{x} = Ax + Bu$, restriction.

In the second step, the closed loop dynamic system, $\dot{x} = (A - BK(Q,R))x$, is obtained by the control law implementation, u, and its eigenstructure calculated.

In the third step, the calculated values are compared with the desired eigenstructure, i.e., the closed loop spectrum must satisfy the desired range and its eigenvectors must satisfy the required eigenvalues sensitivities. Then, the LQR design problem is formulated as multiobjective problem (MOP), relations 2 to 4, dedicated to EA by joining the LQR solution and a the closed loop eigenvalues sensitivity bounds, inequality 3, and the eigenvalues spectrum, inequality 4.

$$\min_{Q,R} \sum_{i=1}^{n} s_i(Q,R) \tag{2}$$

s.t.

$$s_i(Q,R) \leq 1 \quad i = 1, ..., n \tag{3}$$

$$\lambda_{Li} \leq \lambda_{Ci}(Q,R) \leq \lambda_{Ri} \quad i = 1, ..., n \tag{4}$$

where $s_i(Q,R) = (\frac{\|L_i(Q,R)\|_2 \|R_i(Q,R)\|_2}{< L_i(Q,R) R_i(Q,R) >})/\epsilon_i$ is the i-th normalized eigenvalues sensitivity and the i-th design specification $\epsilon_i > 0$; $\|L_i(Q,R)\|_2$ and $\|R_i(Q,R)\|_2$ are the 2-norm of the left and right eigenvectors, respectively, and $< L_i(Q,R)\ R_i(Q,R) >$ is the eigenvectors dot product. λ_{Li} and λ_{Ri} are the left and right i-ths eigenvalues bounds, respectively, for the i-th calculated eigenvalue λ_{Ci}.

2.2 MOGA's basic elements

The GA-optimizer and the DMU are the $MOGA$ basic elements. The GA-optimizer explores the search space to find out Q and R matrices that can satisfy the required eigenstructure and the DMU defines how the search must be done to obtain these matrices.

In a high level of abstraction the GA-optimizer can be considered as a three stage structure. In the first, the Q and R matrices are modeled, each pair of matrices comprises an individual QR and a set of QR individual comprises a population; they are modeled in a decimal basis. In the second stage, the genetic chromosomic operations are performed; besides the traditional genetic operations as crossover, mutation and duplication of another operator called guest was developed to untrack the QR population of saturated regions or even increase the genetic material diversity because a feature of this GA is to stress the search space with a small size population. In the third, the fitness function calculation ranks each individual of the population after each search cycle; this GA is comprised of a fitness function (FF) set, called fitness function team (FFT), each of which represents the same environment in a different manner.

The DMU is a logical device designed to guide the GA search. The decisions taken by this logical device are based on fitness function team structures, schema theory, multiarmed bandit paradigm and Pareto's optimality criterion. The X-$OVER$ operation is the interaction point between the GA-optimizer and the strategies formulated on the DMU, ie, DMU actions are implemented and DMU input are the closed loop system cost functions and eigenstructure values.

2.3 The distributed parallelization

The EA solution determined by the proposed method is a two level parallelization process. The first level is related with GA-optimizer, in this parallelization an initial population is randomly assembled by the coordinator (master) $MOGA$ and it is distributed among several coordinates (slaves) $MOGA$'s, each $MOGA$ evolves the initial QR individuals set and the final population is sent to the coordinator $MOGA$ after some stopping criterion

is reached, this model is called independent run parallelization, (Koza, 1992). The second level of parallelization is related with the LQR problem solution, several LQR designs are performed simultaneously among the distributed $MOGA$'s.

The parallel $MOGA$ ($PMOGA$) model for EA on distributed environment, as function of GA-optimizers, DMU's, FF team, genetic operations and so on, is defined by Eq's (5- 8)

The $PMOGA$:

$$PMOGA = \{MOGA_0, ..., MOGA_{n-1}\} \quad (5)$$

where $MOGA_0$ is the master $MOGA$ and $MOGA_i$, $i = 1, \ldots n-1$, are the $MOGA$ slaves.

The master and slaves $MOGA$'s:

$$\begin{aligned} MOGA_0 &= \{GA_0 - op, DMU_0\} \quad (6)\\ MOGA_i &= \{GA_i - op, DMU_i\},\\ &\quad i = 1, \ldots, n-1 \end{aligned}$$

where $GA_0 - op$ and DMU_0 are the master optimizer and its DMU. $GA_i - op$ and DMU_i, $i = 1, \ldots, n-1$, are slaves GA-*optimizer* and DMU's, respectively.

The master and slaves GA-optimizers:

$$\begin{aligned} GA_0 - op &= \{G(QR), CO(QR), EV(FF)\} \quad (7)\\ GA_i - op &= \{CO(QR), EV(FF)\},\\ &\quad i = 1, \ldots, n-1 \end{aligned}$$

where $G(QR)$ is the QR initial population generator, $CO(QR)$ are the chromosomic operators, that are built on the individual selection criteria and the genetic operators (mutation, crossover, duplication and guest). EV(FF) is the fitness function evaluation operators.

The DMU's:

$$\begin{aligned} DMU_i = \{&Sear_S(QR, FF_rank), \quad (8)\\ &S(QR), STO_C(QR)\},\\ &i = 1, \ldots, n-1 \end{aligned}$$

where $Sear_S(QR, FF_rank)$ is the search strategies definition for the next $(t+1)$ QR-generation as a function of the ranked t population. $S(QR)$ are the selection criterion operators used to replace individuals of the permanent population. $STO_C(QR)$ are the GA-op stop criteria operators.

3 Parameter control methods

Parameters adaptation of evolutionary computation algorithms (ECA) is not an easy task, because there are quite a few parameters that can be adjusted to improve the search and if most of the parameters are adjusted a coordination among them must be taking into account. The methods developed in this section, mainly, follows the taxonomy of ECA parameter control proposed in (Eiben et al., 1999), but there are some instances in which an extension for the taxonomy can be proposed to handle some events.

Class	Function	Parameter
Search Cycle	*X-over* probability	P_X
	mutation probability	P_M
	duplication probability	P_D
	guest probability	P_V
Genetic Operations	*X-OVER* Q % factor	q_{age}
	X-OVER R % factor	r_{age}
	global mutation factor	FG_M
	local mutation base	LB_M
	local mutation exponent	LE_M
Population	size	Pop_S
	number of replaced individuals at each generation	Pop_R

Table 1. GA-optimizer parameter classification

Table 1 presents the GA-optimizer parameters that can be adjusted to increase the search performance to reach the required EA. In this application, the X-$OVER$ age parameter adjustment was chosen to be the one that will be controlled during the search cycle iterations, because the search is stressed on the crossover chromosomic operation.

The general idea behind the proposed methods for parameter variation that regulates the degree of the X-$OVER$ operation is presented on Fig.1. The search space is divided in quadrants and these quadrants are called artificial niche. The main idea is to move some QR-individuals into other quadrants of the search space or even to move individuals into regions of the same quadrant by X-$OVER$ parameter variations and to explore quadrant regions by giving small variations to these parameters. The number of search trials on a giving region is established by the number of unsuccessful trials.

The developed GA-optimizer performs three types of X-$OVER$ operations. The first is the standard operation, QR individuals off-springs are formed of parts of two parents randomly selected based on fitness proportionality, Eq. 9. The second type is based on the schema theory and on a database assembled with the best feasible individuals of the search cycle, one of the parents comes from the permanent population, chosen alike as the standard operation type, and the other comes from the feasible database, randomly chosen in accordance with his storage position, after the individuals' selection another choice is done to define the schema type that will be kept on after the crossover operation. The third type of operation

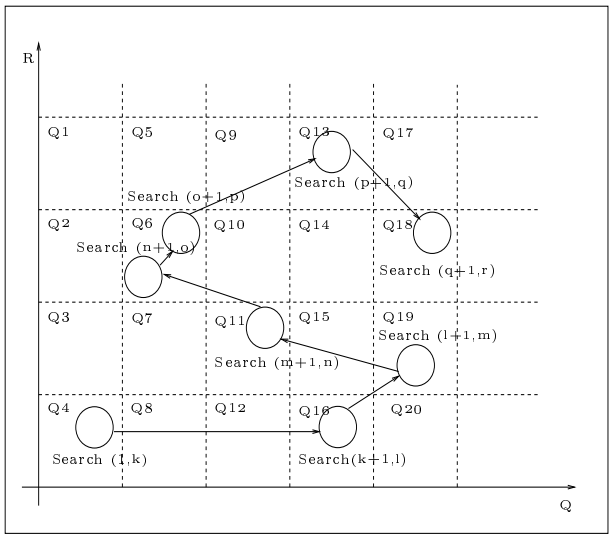

Figure 1. Solution space exploration by *X-over* parameters variations

is performed in a similar way as the second operation type, but the database is assembled with unfeasible individuals; this operation explores the multi-armed bandit (MAB) paradigm.

$$QR_{os_1} = Q_{os_1} \cup R_{os_1} \qquad (9)$$
$$QR_{os_2} = Q_{os_2} \cup R_{os_2}$$

where QR_{os_1} and QR_{os_2} are the off-springs that result from a linear combination of Q, Eq.10, parents alleles.

$$Q_{os_1} = q(t)_{age} Q_{p_1} + |q(t)_{age} - 1| Q_{p_2} \qquad (10)$$
$$Q_{os_2} = q(t)_{age} Q_{p_2} + |q(t)_{age} - 1| Q_{p_1}$$

where $q(t)_{age}$ is the Q alleles *X-OVER* parameter that can be changed during the search cycle iterations according to rules defined by the designer. Q_{p_1} and Q_{p_2} are the Q alleles of parents QR_1 and QR_2, respectively. The R_{os_1} and R_{os_2} are assembled in a similar way.

Types 2 and 3 of *X-OVER* operations, the $q(t)_{age}$, Eq.10, or $r(t)_{age}$ parameters are equal *one* for the search cycle iterations under consideration and depending on the random choice for a Q or R schema type. Supposing that a Q-schema was randomly chosen from the MAB database, Eq.11 represents the two off-springs that are formed from the $Q_{p_{MAB}}$ alleles and the $q(t)_{age}$ parameter is one. The R_{os_1} and R_{os_2} alleles are assembled as part of the R_{p_1} and $R_{p_{MAB}}$, Eq.12. The two QR off-springs are given by Eq.9.

$$Q_{os_1} = Q_{p_{MAB}} \qquad (11)$$
$$Q_{os_2} = Q_{p_{MAB}}$$

where $Q_{p_{MAB}}$ are the Q alleles that comes from the MAB database.

$$R_{os_1} = r(t)_{age} R_{p_1} + |r(t)_{age} - 1| R_{p_{MAB}} \qquad (12)$$
$$R_{os_2} = r(t)_{age} R_{p_{MAB}} + |r(t)_{age} - 1| R_{p_1}$$

where $r(t)_{age}$ is the R alleles age's parameter.

The model for the *X-OVER* age parameter variation is represented by Eq.13, this equation rules the values that can be assumed by $q(t)_{age}$ parameter. This function values changes according to two directions on a straight line and the direction changes are defined by fitness function averages, the best feasible solution and the amount of search trials on a given direction. The $q(t)_{age}$ initial value is given on the right direction.

$$q(t)_{age} = \begin{cases} q(t)_r = q(t-1)_r + \Delta q(t) \\ \quad \text{if } q_d > 0 \\ q(t)_l = q(t-1)_l - \Delta q(t) \\ \quad \text{if } q_d < 0 \end{cases} \qquad (13)$$

where $q(t)_r$ and $q(t)_l$ are the parameter values into the right and left directions. The $\Delta q(t)$ is the $q(t)_{age}$ variation step; this value may be constant or not during the search cycle and may be adjusted in intervals, synchronized with $q(t)$ change points, defined by the designer. This rule actions, in general, are presented in Fig. 2.

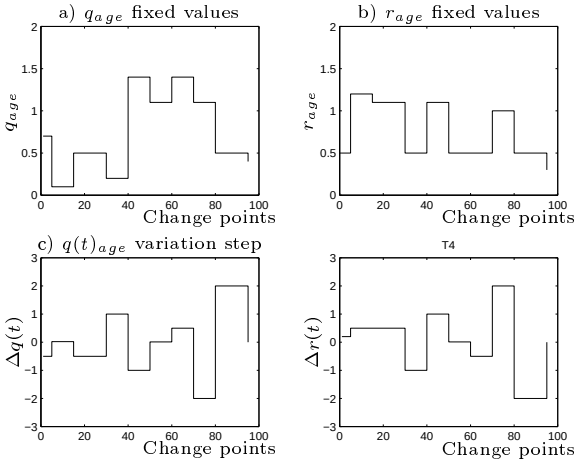

Figure 2. Q and R alleles *X-over* parameters setup and variation steps *versus* search cycle change points

The q_d, Eq.14, is given by a boolean function that defines the $q(t)_{age}$ parameter search direction, if the function value is greater than zero the variation goes into the right on the straight line and if its value is lesser than zero its variation goes to the left.

$$q_d = q_d(\Delta(Q,R)^{av}, E(Q,R)^{av}, \qquad (14)$$
$$\Delta(Q,R)^{max}, \Delta D_q)$$

where $\Delta(Q,R)^{av}$ is the population sensitivity average, $(\sum_j^N \max_i s_i^t(Q,R) \ /N)$, N is the

Method	Rule	$\Delta q(t)$	$\Delta r(t)$	D_{val_q}	D_{val_r}
QD	$R^{\dagger}$	0	0	∞	∞
DN	$R^{\ddagger}$	$\neq 0$	$\neq 0$	$<< SC$	$<< SC$
	$R^{\diamond}$	$\neq 0$	$\neq 0$	∞	∞

Table 2. Parameters variations methods and rules. $R^{\dagger}-$ Several points of the search cycle, $R^{\ddagger}-$ Adaptive, $R^{\diamond}-$ Deterministic

population size, on the search cycle step t. The $E(Q,R)^{av}$ is the population fitness cost function average value, $(\sum_i^N E_i^t(Q,R)/N)$, on step t. The $\Delta(Q,R)^{max}$ is given by $\min_j \max_i s(Q,R)_i^t$, i.e., it represents the best individual of the population on step t. The ΔD_q is a function of the maximum number of faulty trials variations allowed on a given direction. The boolean function q_d in terms of logical and relational operator is given by Eq. 15, this parameter has its direction changed only if all the relations are false.

$$q_d = \delta(t)^{av} \vee e(t)^{av} \vee \delta(t)^{max} \vee \delta D(t)_q \quad (15)$$

where $\delta^{av}(t)$ is a boolean value that results from the comparison of $(\Delta(Q,R,t)^{av} < \Delta(Q,R,t-1)^{av})$. $e(t)^{av}$ is a boolean value that results from the comparison of $(E(Q,R,t)^{av} < E(Q,R,t-1)^{av})$. $\delta(t)$ is a boolean value that results from the comparison of $(\Delta(Q,R,t)^{max} < \Delta(Q,R,t-1)^{max})$. $\delta D(t)_q$ is a boolean value that results from the comparison of $(\Delta D(t) < D_{val_q})$ and D_{val_q} is the maximum value of the parameter variations on a given direction if none of the other relations were not satisfied.

The parameter variation model, Eq. 13, allows to obtain two methods for $q(t)_{age}$ and $r(t)_{age}$ variations, Table 2. They are classified as quasi-dynamic (QD) and dynamic (DN) methods.

In the quasi-dynamic method parameter changes are ruled by fixed variations during the search cycle, Fig.'s 2(a and b). For this case, parameter $\Delta q(t)$ is zero during the whole search cycle.

The second method has two rules: adaptive and deterministic. The first rule is called adaptive because its $\Delta q(t)$ fixed parameter is not zero and its fixed parameter D_{val_q} of the boolean function, Eq. 15, is less than the number of the search cycle (SC) iterations. The second rule is considered deterministically controlled because its fixed parameter D_{val_q} is always greater than the number of the search cycle iterations and its parameter $\Delta q(t)$ or $\Delta r(t)$ is not zero; in this rule $q(t)_{age}$ and $r(t)_{age}$ parameters can not change to the left or the right directions, which are defined by the designer.

4 Concluding Remarks

A model development for parameter adjustment, that regulates the degree of combination of two individuals performed by *X-OVER* operation, and two methods, that results from certain assumptions over the proposed model, has been presented as a function of performance index and constant parameters established by the designer.

The model basic steps are based on the results obtained from the fitness function structure and the information furnished by these results are evaluated by boolean and relational operators.

The performance evaluation of the proposed methods has shown their effectiveness. The evaluation tests are performed to allocate an eigenstructure of a dynamic system and its results are presented in a second paper, (Bottura and Neto, 2001).

References

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*, first edn, Oxford Unversity Press, Inc, 198 Madison Avenue, New York, New York 10016.

Bottura, C. P. and Neto, J. F. (1999a). Parallel Eigenstructure Assignment via LQR Design and Genetic Algorithms, *American Control Conference – ACC99. San Diego, California, USA*. pp. 2295–2296.

Bottura, C. P. and Neto, J. F. (1999b). The Schema Theorem and Multiarmed Bandit Paradigm Influences on Eigenstructure Assignment via LQR Designs, 4^o *Simpósio Brasileiro de Automação Inteligente - SBAI99. São Paulo - SP - Brasil* pp. 502–506.

Bottura, C. P. and Neto, J. F. (2001). X-OVER Parameter Control for a GA-optimizer dedicated to Eigenstructure Assignment/LQR designs - Part II - Computational simulations and Performance Analysis, *V Simpósio Brasileiro de Automação Inteligente - VSBAI - Canelas-RS - Brasil* .

Bottura, C. P. and Neto, J. V. F. (1999c). Parallel Genetic Algorithm Fitness Function Team for Eigenstructure Assignment via LQR Designs, *Congress on Evolutionary Computation - CEC99. Washington, DC, USA. Vol* **2**: 1035–1042.

Bottura, C. P. and Neto, J. V. F. (2000). Rule-based Decision Making Unit for Eigenstructure Assignment via Parallel Genetic Algorithm and LQR Designs, *American Control Conference – ACC2000* .

Choi, J. W. and Seo, Y. B. (1999a). LQR Synthesis via Eigenstructure Assignment, *IFAC- 14th Triennial World Congress, Beijing, P.R. China* pp. 153–158.

Choi, J. W. and Seo, Y. B. (1999b). LQR Design with Eigenstructure Assignment Capability, *IEEE Transactions on Aerospace and Eletronic Systems* **35**(2): 700–708.

Davis, R. and Clarke, T. (1995). Parallel Implementation of a Genetic Alghorithm, *Control Eng. Practice* **3**(1): 11–19.

Dorato, P., Abdalah, C. and Cerone, V. (1995). *Linear Quadratic Control*, Prentice-Hall, Englewood Cliffs.

Eiben, . E., Hiterding, R. and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms, *IEEE Transactions on Evolutionary Computation* **3**(2): 124–141.

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimization, and machine learning*, Addison-Wesley Publishing Company Inc., USA.

Graupe, D. (1972). Derivation of Weighting Matrices towards satisfying Eigenvalue requirements, *Int. J. Control* **16**(5): 881–888.

Harvey, C. A. and Stein, G. (1978). Quadratic Weights for Asymptotic Regulator, *IEEE Transactions on Automatic Control* **23**(3): 378–387.

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

Kawasaki, N. and Shimemura, E. (1983). Determining Quadratic Weighting Matrices to Locate Poles in a Specified Region, *Automatica* **19**(5): 555–560.

Koza, J. R. (1992). *Genetic Programmming : On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Massachusetts - USA.

Medanic, J., Tharp, H. and Perkins, W. (1988). Pole Placement by Performance Criterion Modification, *IEEE Transactions on Automatic Control* **33**(5): 469–472.

Ochi, Y. and Kanai, K. (1996). Eigenstructure Assignment for Linear Quadratic Regulator, *IFAC Proceedings of 13th Triennial World Congress - San Francisco - USA* pp. 115–120.

Stein, G. (1979). Generalized Quadratic Weights for Asymptotic Regulator Properties, *IEEE Transactions on Automatic Control* **24**(4): 559–566.