Multi-Objective Pole Placement with Evolutionary Algorithms

Gustavo Sánchez, Minaya Villasana, and Miguel Strefezza

Universidad Simón Bolívar. Venezuela
gsanchez@usb.ve

Abstract. Multi-Objective Evolutionary Algorithms (MOEA) have been succesfully applied to solve control problems. However, many improvements are still to be accomplished. In this paper a new approach is proposed: the Multi-Objective Pole Placement with Evolutionary Algorithms (MOPPEA). The design method is based upon using complex-valued chromosomes that contain information about closed-loop poles, which are then placed through an output feedback controller. Specific cross-over and mutation operators were implemented in simple but efficient ways. The performance is tested on a mixed multi-objective $\mathcal{H}_2/\mathcal{H}_\infty$ control problem.

Key words: Multi-objective control; Pole placement; Evolutionary Algorithms.

1 Introduction

Most control design problems can be solved using numerical optimization. Thus, Multi-Objective Evolutionary Algorithms (MOEA) have been successfully applied for this purpose, provided the problem is non-convex in the optimization parameters and cannot be efficiently solved by conventional local optimization algorithms [1]. To illustrate this point, let's review five previous related publications. Note that an excellent survey can be found in [2].

In 1995, Fonseca and Fleming [3] developed an approach to multiple objective and constraint handling with genetic algorithms, with application to control system design. A Multiple Objective Genetic Algorithm (MOGA) was proposed, which is still frequently used in many applications.

In 1995, Whidborne *et al* [4] compared the performance of three search methods. Two were based on hill-climbing techniques: Nelder-Mead Dynamic Min-Max (NMDM) and Moving Boundaries Process (MBP). The third was precisely MOGA. The three were found to be useful for interactive multi-objective controller design. Besides, the author introduced MODCONS: a MATLAB toolbox for Multi-Objective Design of Control Systems.

In 2000, Herreros [5] proposed an algorithm for Multi-objective Robust Control Design (MRCD). It was tested against a Linear Matrix Inequalities (LMI) approach for mixed multi-objective $\mathcal{H}_2/\mathcal{H}_\infty$ control problems. An adaptive search

space was proposed, motivated by two reasons: the selection of the initial population and the delimitation of the search space. In fact, these are still open problems in the field.

In 2005, Liu and Ishihara [6] discussed the use of multi-objective genetic algorithms and the method of inequalities. The performance of the proposed design method was tested on a special set of benchmark control problems.

In 2006, Molina-Cristobal *et al* [7] compared MOGA against a LMI approach to find the trade-off of a multi-objective $\mathcal{H}_2/\mathcal{H}_\infty$ control problem. The author asserted that MOGA could find an improved Pareto-optimal front compared to the LMI approach.

Despite all this important work, many improvements remain still to be accomplished. This includes elements like parameters tuning, space search adaptation, performance assessment and controller coding. Particularly, regarding the latter, a new approach for solving the design problem is proposed in this work: the Multi-Objective Pole Placement with Evolutionary Algorithms (MOPPEA) technique.

The main idea is using complex-valued chromosomes, containing information about closed-loop poles. This representation allows poles be placed through a classical observer-based feedback controller, based on the information contained within each chromosome. Note that, unlike this representation, usually controllers are coded in terms of real parameters [8]. Specific cross-over and mutation operators were implemented in simple but efficient ways.

The exposition is organized as follows. In section **2**, we formulate the controller design problem. The proposed solution method is described in section **3**. In section **4**, it is applied to solve a mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem. Finally, conclusions are drawn in section **5**.

2 Problem Formulation

2.1 Preliminaries

Let $L_2^{p\times m}$ be the set of $p \times m$ matrix functions $f : \mathcal{R}_0^+ \rightarrow \mathcal{R}^{p\times m}$ such that:

$$\int_0^{+\infty} trace[f^T(t)f(t)]dt < \infty \tag{1}$$

Let $\mathcal{R}(s)^{p\times m}$ be the set of $p\times m$ rational complex matrix functions $G : \overline{\mathfrak{C}}^+ \rightarrow \mathfrak{C}^{p\times m}$ such that:

$$G_{ij}(s) = \frac{b_{nu}s^{nu} + b_{nu-1}s^{nu-1} + .. + b_1 s + b_0}{s^{nd} + a_{nd-1}s^{nu-1} + .. + a_1 s + a_0} \quad i = 1, 2, ..., p \quad j = 1, 2, ..., m \tag{2}$$

and $nd \geq nu$, $\forall G_{ij}(s)$; $a_k, b_h \in \mathcal{R}, k = 1, 2, ..., nd$ and $h = 1, 2, ..., nu$.

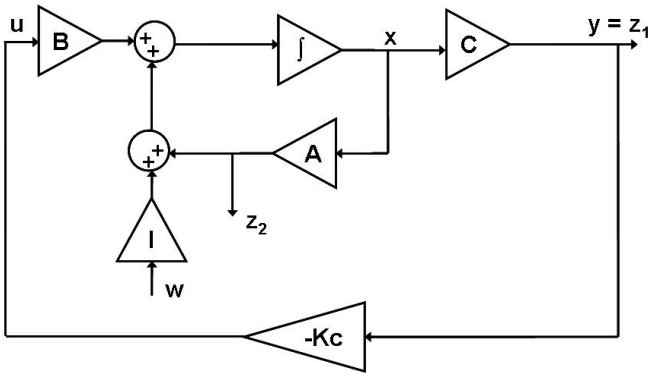

Fig. 1. Continuous-time closed-loop design model

2.2 Control Topology

We focus on the problem of designing a linear controller $Kc \in \mathcal{R}(s)^{n_u \times n_y}$ for the continuous-time model shown in figure 1. Matrices $A \in \mathcal{R}^{n \times n}$, $B \in \mathcal{R}^{n \times n_u}$ and $C \in \mathcal{R}^{n_y \times n}$ denote the given plant state matrices.

As usual, $w \in L_2^{n_w \times 1}$denote the exogenous input, $z_1 \in L_2^{n_{z_1} \times 1}$ and $z_2 \in L_2^{n_{z_2} \times 1}$ represent the outputs to be regulated, while $u \in L_2^{n_u \times 1}$ and $y \in L_2^{n_y \times 1}$ represent the control input and the measured output respectively. It is assumed that $G(s) = C(sI - A)^{-1}B$ is strictly proper, stabilizable from u and detectable from y. The open-loop state-space equations are:

$$\begin{cases} \dot{x} = Ax + Iw + Bu \\ z_1 = y = Cx \\ z_2 = Ax \end{cases} \tag{3}$$

The state-space equations of the controller are:

$$\begin{cases} \dot{x}_c = A_c x_c - Ly \\ u = Kx_c \end{cases} \tag{4}$$

with $K \in \mathcal{R}^{n_u \times n}, L \in \mathcal{R}^{n \times n_y}$ and

$$Kc(s) = -K(sI - A_c)^{-1}L \tag{5}$$

Let

$$T_1(K_c) = T_{z_1 w}(K_c) \tag{6}$$

$$T_2(K_c) = T_{z_2 w}(K_c) \tag{7}$$

be the closed-loop transfer function from w to z_1 and z_2 respectively.

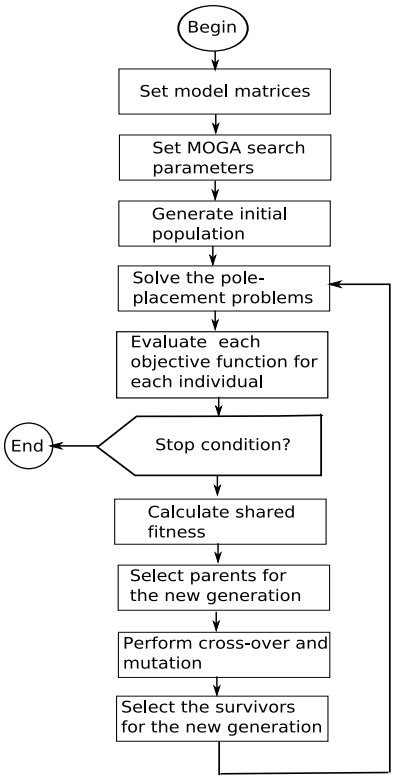

Fig. 2. MOPPEA flowchart

2.3 The Control Problem

The mixed $\mathcal{H}_2/\mathcal{H}_\infty$ Objective Control Problem (MOCP) can now be stated as:

$$\min_{Kc \in \mathcal{K}^n \subset \mathcal{R}(s)^{n_u \times n_y}} \begin{pmatrix} \|T_1(K_c)\|_2 \\ \|T_2(K_c)\|_\infty \end{pmatrix} \tag{8}$$

where $\mathcal{K}^n \subset \mathcal{R}(s)^{n_u \times n_y}$ is the set of all stabilizing controllers of degree n. Note that in this problem the term "*min*" means finding a solution which give the values of the objective functions acceptable to the designer [9]. Note also that control law specifications (i.e. controller structure) are not considered in this formulation.

3 MOPPEA: a Linear Controller Design Method.

A flowchart of the proposed design method is shown in figure 2.

3.1 The Pole Placement Method

An output feedback controller can be designed by combining a full information controller (i.e. a controller which has immediate access to the information about all states) with a state observer (a subsystem which attempts to reconstruct the current states using the information about past measurements and control inputs). The resulting output feedback sub-system is called "observer-based controller" and has the following state-equations:

$$\begin{cases} \dot{\widehat{x}} = (A + BK + LC)\,\widehat{x} - Ly \\ u = K\widehat{x} \end{cases} \tag{9}$$

where $\widehat{x}$ is the estimated state. Thus, in our framework the system closed-loop state equations, using the estimation error $\widehat{e} = x - \widehat{x}$ as state variable are:

$$\begin{cases} \begin{pmatrix} \dot{x} \\ \dot{\widehat{e}} \end{pmatrix} = \begin{pmatrix} A + BK & -BK \\ 0 & A + LC \end{pmatrix} \begin{pmatrix} x \\ \widehat{e} \end{pmatrix} + \begin{pmatrix} I \\ I \end{pmatrix} w \\ z_1 = y = \begin{pmatrix} C\ 0 \end{pmatrix} \begin{pmatrix} x \\ \widehat{e} \end{pmatrix} \\ z_2 = \begin{pmatrix} A\ 0 \end{pmatrix} \begin{pmatrix} x \\ \widehat{e} \end{pmatrix} \end{cases} \tag{10}$$

Let $pk \in \mathfrak{C}^{n_k}$ and $pl \in \mathfrak{C}^{n_l}$ be the eigenvalues of $A + BK$ and $A + LC$ respectively. To assure closed-loop system stability, the gain matrix K and L must be calculated in such way that pk and pl belong to $\mathfrak{C}^-$ (open left-half complex plan). Several algorithms to compute K and L from pk, pl, B and C have been proposed [10]. In this work, the MATLAB *place* function has been used.

3.2 Problem Reformulation

The key concept of the proposed design method is using an evolutionary process in order to evolve pk and pl, moving across $\mathfrak{C}^-$ in order to find the best feasible closed-loop poles locations. Thus, the MOCP problem (see equation 8) can be stated again as:

$$\min_{pk,pl \in \mathfrak{C}^-} \begin{pmatrix} \|T_1(pk,pl)\|_2 \\ \|T_2(pk,pl)\|_\infty \end{pmatrix} \tag{11}$$

Note that, in this case, the stability restriction $Kc \in \mathcal{K}^n$ has disappeared. This is the main advantage of the proposed method when compared to previous works (see [5] and [7]).

3.3 Representation of Individuals

Different representations have been proposed for controllers (see [5], [7] and [8]). In this work, chromosomes are complex-valued vectors containing the concatenation of pk and pl (see figure 3). A recursive algorithm has been implemented for randomly generating the initial population (see appendix A). Parameters *supr* and *supi* determine the size of the initial search space.

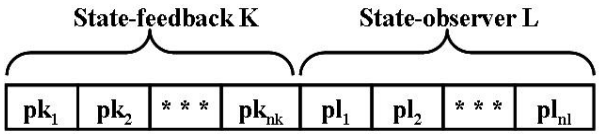

Fig. 3. Chromosome structure

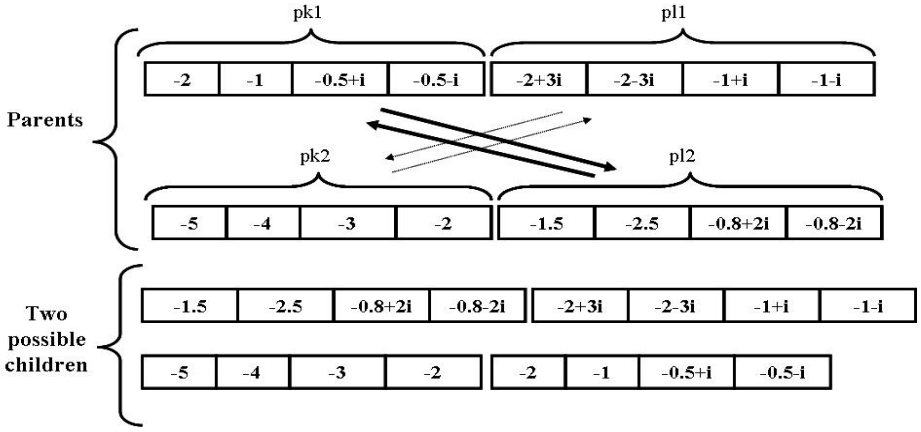

Fig. 4. First cross-over operator

3.4 Variation Operators

Two cross-over operators were implemented. The first (see figure 4) performs a "block" exchange between pk and pl, belonging to different individuals. The second (see figure 5) performs an uniform random cross-over [11].

Moreover, two mutation operators were implemented. The first (see figure 6) works like cross-over operator #1, but this time between pk and pl belonging to the same individual. The second (see figure 7) slightly moves the poles in random directions.

3.5 Multi-Objective Genetic Algorithm (MOGA)

MOGA has been widely applied to solve a number of practical applications [3]. This algorithm assigns the smallest rank value for all non-dominated individuals. The dominated ones are ranked according to the number of individuals that dominated them. The fitness value of each individual is computed by implementing a mapping inversely related to its rank. This fitness will be degraded based upon a sharing function, according to the distribution density in the feature space. The parameter α regulates the shape of the sharing function.

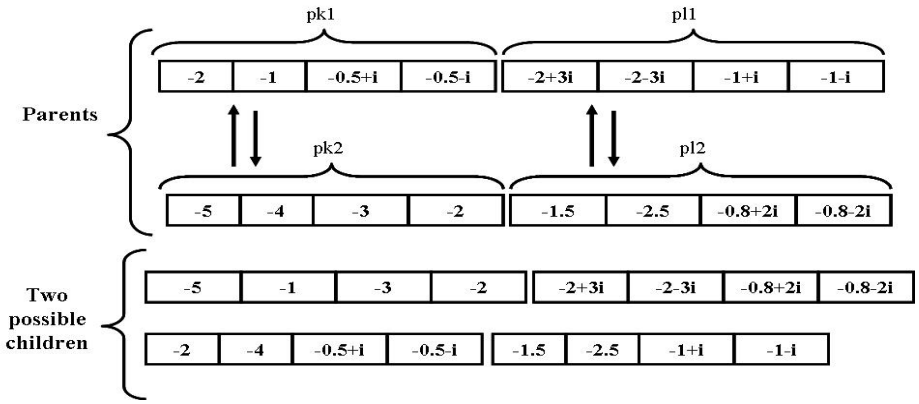

Fig. 5. Second cross-over operator

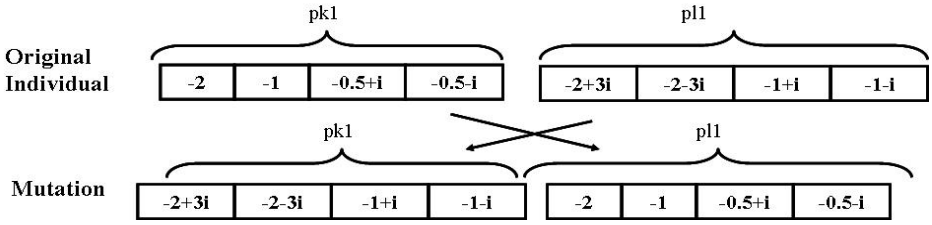

Fig. 6. First mutation operator

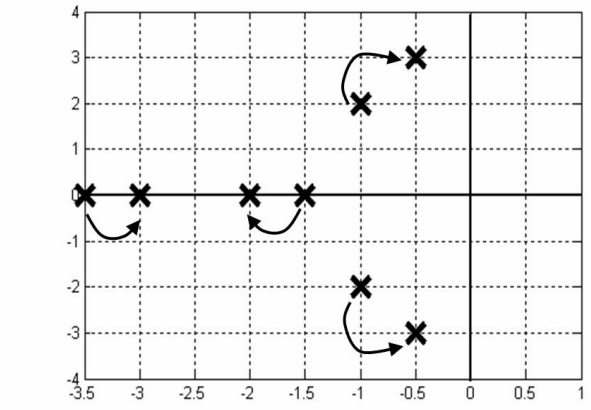

Fig. 7. Second mutation operator

The sharing distance σ_{share} determines the extent of the sharing region for each individual. In this work (two objectives case), the following equation was used:

$$\sigma_{share} = \frac{d}{2 \times N} \tag{12}$$

where d is the diameter of the trade-off curve (estimated from previous works [5] and [7]) and N is the population size.

A simple mating restriction mechanism was also implemented: only pairs of individuals that lie within a distance of σ_{mate} were allowed for mating [1].

Parents and survivor selection were implemented using the "Stochastic Universal Sampling" (SUS) algorithm, based on shared fitness values.

4 Design Example: a Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ Control Problem

The proposed design method was applied to solve the mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem (see equation 11) with the following state matrices:

$$\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{ccc|c} -21 & -120 & -100 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 150 & 0 \end{array}\right) \tag{13}$$

The same problem, with the same state matrices, was tackled in [5] and [7].

4.1 Experimental Results

All algorithms were coded in MATLAB, taking advantage of its control toolbox. Table 1 shows the parameters used during tests. Figure 8 shows the evolution of a typical attainment surface after 25, 50 and 100 generations. An attainment surface is the family of tightest goal vectors known to be attainable during the optimization process. Note that these results are comparable to those presented in [7].

5 Conclusions and Future Work

The proposed approach was able to find attainment surfaces as good as previous works. Its main advantage is that the stability restriction disappears from the optimization problem. However, the controller order is high and its structure cannot be optimized.

The goal of this paper was just to show experimentally that the proposed method is efficient enough. The focus was not really put on improving the optimization tool. Moreover, no other performance measure has been tested: it

Table 1. MOPPEA parameters

Initial Population	Randomly generated
Representation	Complex vectors
supr	10
supi	10
Cross-Over Recombination	Cross-over operators 1 and 2
Cross-Over Rate	0.8
Prob. Cross-Over Op. 1	0.3
Prob. Cross-Over Op. 2	0.7
Mutation Operator	Mutation operators 1 and 2
Mutation Rate	0.1
Prob. Mut. Op. 1	0.1
Prob. Mut. Op. 2	0.9
Population Size	100
σ_{share}	2.7733
α	2
σ_{mate}	100
Stop Condition	100 generations
Population Size	100
Offspring Size	100
Parents Selection	SUS $(s = 2)$
Survivor Selection	SUS $(s = 2)$

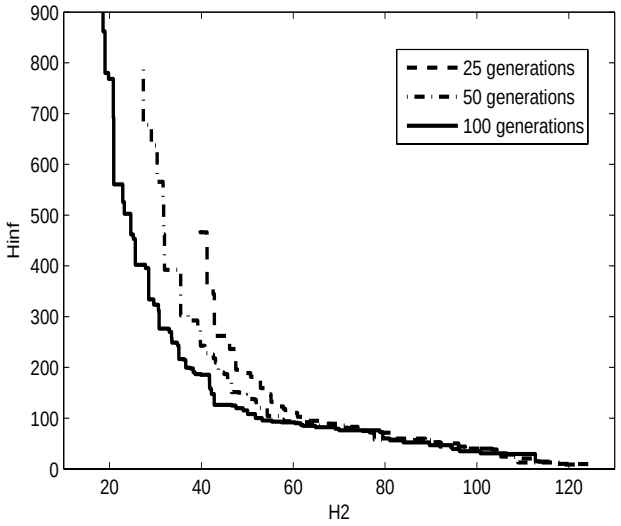

Fig. 8. Evolution of a typical attainment surface after 25, 50 and 100 generations

is well-known that assessing the performance of a multi-objective optimization algorithm is also a multi-objective problem [12].

Despite the simplicity of the proposed method, problems have been reported when using the pole placement technique. In high-order systems, certain pole locations result in very large gains [13]. This fact suggests caution during the optimization evolutionary process: a penalty mechanism can be used in order to avoid such locations.

Finally, it is clear that more tests are needed. The authors are currently developing a MATLAB toolbox, which allows using "state-of-the art" MOEA, in order to solve more complex control problems.

References

1. K.C Tan, E.F Khor, and T.H Lee. *Multiobjective Evolutionary Algorithms and Applications*. Springer-Verlag, 2005.
2. Peter J. Fleming and R.C. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice (10)*, pages 1223–1241, 2002.
3. Carlos Fonseca. *Multiobjective Genetic Algorithms with Application to Control Engineering Problems*. PhD thesis, Departement of Automatic Control and Systems Engineering. University of Sheffield, 1995.
4. J.F Whidborne, D.W Gu, and I. Postlethwaite. Algorithms for the method of inequalities: A comparative study. *American Control Conference, Washington*, 1995.
5. Alberto Herreros. Diseño de controladores robustos multiobjetivo por medio de algoritmos geneticos. *Tesis Doctoral, Departamento de Ingenieria de Sistemas y Automatica.Universidad de Valladolid, Espana.*, 2000.
6. V Zakian. *Control systems design: a new framework*. Springer-Verlag London Limited, 2005.
7. A Molina-Cristobal, I.A Griffin, P.J Fleming, and D.H Owens. Linear matrix inequalities and evolutionary computation. *International Journal of Systems Science. Vol. 37, No. 8*, pages 513–522, 2006.
8. Shinn-Jang Ho, Shinn-Ying Ho, Ming-Hao Hung, Li-Sun Shu, and Hui-Ling Huang. Designing Structure-Specified Mixed H2/Hinf Optimal Controllers Using an Intelligent Genetic Algorithm. *IEEE Transactions on Control System Technology, Vol. 13, No. 6*, pages 1119–1124, 2005.
9. A. Osyczka. *Multicriteria Optimization for Engineering Design*. Academic Press, Cambridge, MA, 1985.
10. J. Kautsky and N.K. Nichols. Robust Pole Assignment in Linear State Feedback. *Int. J. Control, 41*, 1985.
11. A.E Eiben and J.E Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.
12. E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms on Test Functions of Different Difficulty. In Annie S. Wu, editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 121–122, Orlando, Florida, 1999.
13. A.J Laub and M. Wette. Algorithms and Software for Pole Assignment and Observers. *UCRL-15646 Rev. 1, EE Dept., Univ. of Calif., Santa Barbara, CA*, 1984.

6 Appendix A: a MATLAB Recursive Function for Randomly Generating the Initial Population

```
function [ pol,nr,ni ] = genpol( n, supr, supi )
   %supr and supi are the limits of the complex region
    if n == 1,
       pol = -supr*[rand]; nr = 1; ni = 0;
       return
   end
    if n == 2,
        if rand <= 0.5,
           pol = -supr*[rand;rand];
           nr = 2;ni = 0;
        else
           im = 2*rand-1;re = -rand;
           pol = [complex(supr*re,supi*im);complex(supr*re,-supi*im)];
            nr = 0;ni = 1; end
   return
   end
   if n >= 3,
   est = rand;
       if est <= 0.5,
            polaux = -supr*[rand;rand];
            nr = 2;
            ni = 0;
        else
           im = 2*rand-1;
            re = -rand;
    polaux = [complex(supr*re,supi*im);complex(supr*re,-supi*im)];
           nr = 0;
           ni = 1;
       end
    [polaux1,nraux,niaux] = genpol(n-2,supr,supi);
    if est <= 0.5,
        pol = [ polaux ; polaux1];
    else
        pol = [ polaux1 ; polaux];
    end

    nr = nr + nraux;
    ni = ni + niaux;
    return
    end
```