

## DETC99/DAC-8586

### A FUZZY ADAPTIVE SIMPLEX SEARCH OPTIMIZATION ALGORITHM

**Mohamed B. Trabia**

Associate Professor  
University of Nevada, Las Vegas  
Department of Mechanical Engineering  
Las Vegas, NV 89154-4027  
Telephone: (702) 895-0957  
E-mail: mbt@me.unlv.edu

**Xiao Bin Lu**

Graduate Student  
University of Nevada, Las Vegas  
Department of Mathematical Sciences

**Keywords:** Optimization Algorithms, Nonlinear Programming, Simplex, Fuzzy Logic

#### ABSTRACT

Optimization algorithms usually use fixed parameters that are empirically chosen to reach the minimum for various objective functions. This paper shows how to incorporate fuzzy logic in optimization algorithms to make the search adaptive to various objective functions. This idea is applied to produce a new algorithm for minimization of a function of  $n$  variables using an adaptive form of the simplex method. The search starts by generating a simplex with  $n+1$  vertices. The algorithm replaces the point with the highest function value by a new point. This process comprises reflecting the point with the highest function value in addition to expanding or contracting the simplex using fuzzy logic controllers whose inputs incorporate the relative weights of the function values at the simplex points. The efficiency of the algorithm is studied using a set of standard minimization test problems. This algorithm generally results in a faster convergence toward the minimum. The algorithm is also applied successfully to two engineering design problems.

#### INTRODUCTION

Fuzzy logic is a means for transforming linguistic knowledge into mathematical model. It has been applied extensively to the field of automatic control where it succeeded in modeling and control of many systems that cannot be accurately described using classical control techniques. While fuzzy logic has many applications in the control field, it is not widely used in the optimization area. Several researchers, e.g. Rao (1986), used fuzzy logic to describe the objective function and constraints. Lately, Mulkay and Rao (1997) used fuzzy logic to control the parameters of the sequential linear programming algorithm

recently proposed in a parallel approach to the one proposed in this paper. The parameters of most optimization techniques are usually rigid. The objective of this paper is to make these parameters adaptive by using fuzzy logic to produce more efficient search (lesser number of function evaluations). The paper specifically deals with making the simplex method adaptive.

The simplex method for minimization of a function of  $n$  variables was first introduced by Spendley *et al.* (1962). In this method, the search starts by generating a simplex with  $n+1$  vertices. The algorithm evaluates the function values at these points, and replaces the point of the highest function value with its reflection along a vector passing through the center of the remaining points. While this algorithm is simple, it can have problems such as slow rate of convergence or cycling. The simplex algorithm was further enhanced by the work of Nelder and Mead (1965) who proposed expanding or contracting the simplex based on evaluating the function values of the simplex vertices and the reflection point. Rekalitis *et al.* (1983) presented a variation on the algorithm of Nelder and Mead to make it more responsive to changes in the function values at the vertices of the simplex. The simplex algorithm uses only function evaluations to determine its search direction, which is especially useful when the function is highly nonlinear or has discontinuities. While Nelder and Mead algorithm was proposed several decades ago, it is still widely used by several commercial optimization programs.

The three simplex algorithms described above use crisp criteria to determine the amount of contraction or expansion of the new point that will replace the point of the highest function value. These criteria do not include the relative weights of the

function values at the simplex vertices. For example, it can be argued that if the function value at the point of the lowest function value is *much larger* than function value at the reflection point, a *significant* expansion of the simplex is recommended. On the other hand, if the function value at the point of the lowest function value is *little bit larger* than function value at the reflection point, a *moderate* expansion of the simplex should be recommended. These last two statements are linguistic statements, which lend themselves readily to fuzzy logic. This paper proposes using fuzzy logic to determine the movement of the simplex based on evaluating the function values at the simplex points (vertices). More specifically, these controllers will determine the direction of simplex reflection and control the expansion and contraction of the simplex to achieve faster conversion to the minimum.

Section 1 presents a detailed discussion of the fuzzy simplex algorithm. Section 2 includes testing the algorithm using standard test functions as well as discussion of the performance of the proposed algorithm performance. Engineering design examples are presented in Section 3. Conclusions and recommendations for future research are presented in Section 4.

## NOMENCLATURE

- $U_i$  unit vector in  $i^{th}$  coordinate axis
- $X_c$  the centroid of all the simplex points except  $X_h$
- $X_g$  simplex point with next highest function value
- $X_h$  simplex point with the highest function value
- $X_l$  simplex point with the lowest function value
- $X_0$  initial point of the simplex
- $\alpha$  simplex size factor
- $\epsilon$  termination accuracy value
- $\theta_c$  simplex reflection scaling factor
- $\theta_{co}$  simplex contraction scaling factor
- $\theta_{ex}$  simplex expansion scaling factor

## 1. FUZZY SIMPLEX ALGORITHM

Simplex algorithm attempts to minimize a function of  $n$  variables. The initial simplex is created according to Spendley et al. (1962) by generating  $n+1$  equally spaced points, based on an initial guess, according to the equation,

$$X_i = X_0 + \delta_1 U_i + \sum_{j=1, j \neq i}^n \delta_2 U_j \quad (1)$$

where,

$$\delta_1 = \frac{\sqrt{n+1} + n - 1}{n\sqrt{2}} \alpha \quad (2)$$

$$\delta_2 = \frac{\sqrt{n+1} - 1}{n\sqrt{2}} \alpha$$

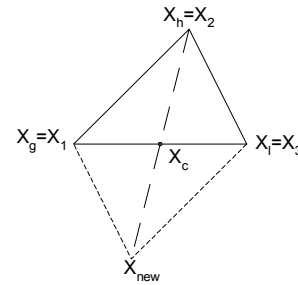
After the algorithm generates the first simplex, it tries to move the simplex toward the minimum. The first step in this process is to identify the simplex point with the highest function value,  $X_h$ , the simplex point with next highest function value,  $X_g$ ,

and the simplex point with the lowest function value,  $X_l$ . The simplex moves toward the minimum using three operations: reflection, expansion, and contraction. In this paper, each of these three operations uses fuzzy logic criteria.

### 1.1 Fuzzy Reflection

Reflection starts by calculating the coordinates of point  $X_c$ , which is the centroid of all the simplex points except  $X_h$ . In the original method of Spendley et al. (1962), a new point,  $X_{new}$  is created as a reflection of  $X_h$  along the  $X_h - X_c$  vector, Figure 1, such that,

$$X_{new} = 2X_c - X_h \quad (3)$$



**Figure 1. Reflection of the highest point in Nelder and Mead Simplex Algorithm**

This paper proposes moving the centroid toward the point of the lowest function value,  $X_l$ , as shown in Figure 2, to direct the simplex toward the minimum using a lesser number of steps. A fuzzy controller, *Reflect* (Appendix A), is used to determine the magnitude of the shift toward  $X_l$  according to this equation,

$$X_{cf} = (1 - \theta_c)X_c - \theta_c X_h \quad (4)$$

$\theta_c$  is the output of the fuzzy logic controller, *Reflect*. This controller has a single input,  $y_h$ , which is a non-dimensional variable that is defined as,

$$y_h = \left| \frac{f(X_h) - f(X_l)}{f(X_h)} \right| \quad (5)$$

The coordinates of the reflected point,  $X_{new}$ , is generated using the following equation,

$$X_{new} = 2X_{cf} - X_h \quad (6)$$

### 1.2 Decisions for the Simplex Movement

Based on comparing the function values at the simplex points with the function value at the new point,  $X_{new}$ , one of these decisions is taken,

- if  $f(X_{new}) \leq f(X_l)$ , Explore expansion by generating  $X_{new1}$
  - if  $f(X_l) < f(X_{new}) \leq f(X_h)$ , replace  $X_h$  by  $X_{new}$
  - if  $f(X_{new}) > f(X_h)$ , Explore contraction by generating  $X_{new2}$
- (7)

The two new points described in the above equation are generated as follows,

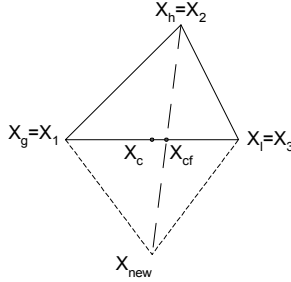
$$X_{new1} = X_{cf} + (\theta_{ex} + 1)(X_{cf} - X_h) \quad (8)$$

$$X_{new2} = X_{cf} + \theta_{co}(X_{cf} - X_h) \quad (9)$$

The function values at these two new points are evaluated and one of these decisions is taken:

- i. If  $f(X_{new1})$  is less than  $f(X_h)$ , replace  $X_h$  by  $X_{new1}$ . Otherwise, replace  $X_h$  by  $X_{new}$ .
- ii. If  $f(X_{new2})$  is less than  $f(X_h)$ , replace  $X_h$  by  $X_{new2}$ . Otherwise, generate a smaller simplex around  $X_l$  as follows,

$$X_i = \frac{X_i + X_l}{2} \quad i = 1..n+1 \quad (10)$$



**Figure 2. Reflection of the Highest Point in Fuzzy Simplex Algorithm**

### 1.3 Fuzzy Expansion and Contraction:

While Equation (7) gives a straightforward rationale on whether to expand or contract the simplex, the magnitude of these two operations is not however clear since each objective function has its own particular characteristics. Nelder and Mead (1965) tested their algorithm on several standard test functions. They experimented with several combinations of expansion and contraction factors,  $\theta_{ex}$  and  $\theta_{co}$ . They found out that the least number of function evaluations results from assigning values of 2 and -0.5 for  $\theta_{ex}$  and  $\theta_{co}$  respectively. These values may work well for some problems. There is no guarantee however that they will result in a fast conversion for every problem. Therefore, instead of using fixed factors for expanding and contracting the simplex, it is proposed to devise an adaptive approach to the problem. In this approach, the values of  $\theta_{ex}$  and  $\theta_{co}$  are determined by two separate fuzzy logic controllers, *Expand* (Appendix B) and *Contract* (Appendix C).

The input to *Expand* is a non-dimensional variable,  $y_{new}$ , which is defined as,

$$y_{new} = \left| \frac{f(X_h) - f(X_{new})}{f(X_h)} \right| \quad (11)$$

The output of *Expand* is  $\theta_{ex}$ , which is used in Equation (8). The input to *Contract* is  $y_h$ , Equation (5). The output of *Expand* is  $\theta_{co}$ , which is used in Equation (9).

### 1.4 Termination Criterion:

A termination criterion is evaluated at the end of each iteration. The termination criterion used in this paper is,

$$\text{if } \sqrt{\frac{\sum_{i=0}^n \left( f(x_i) - \frac{\sum_{i=0}^n f(x_i)}{n+1} \right)^2}{n+1}} < \varepsilon, \quad \text{stop} \quad (12)$$

Figure 3 shows the flowchart for the fuzzy simplex algorithm.

## 2. EVALUATION EXAMPLES

Several functions were used to test the proposed modification of the simplex search algorithm. Other researchers have previously used these functions to evaluate the efficiency of optimization algorithms. These functions are:

1. Powell's Quartic Function, Powell (1964)

$$f(X) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \quad (13)$$

Initial guess:  $(3, -1, 0, 1)^T$  Minimum:  $(0, 0, 0, 0)^T$

2. Fletcher and Powell's Helical Valley, Fletcher and Powell (1963)

$$f(X) = 100 \left( (x_3 - 10\theta(x_1, x_2))^2 + \left( \sqrt{x_1^2 + x_2^2} - 1 \right)^2 \right) + x_3^2 \quad (14)$$

where

$$2\pi\theta(x_1, x_2) = \begin{cases} \arctan \frac{x_2}{x_1} & \text{if } x_1 > 0 \\ \pi + \arctan \frac{x_2}{x_1} & \text{if } x_1 < 0 \end{cases}$$

Initial guess:  $(-1, 0, 0)^T$  Minimum:  $(1, 0, 0)^T$

3. Freudenstein and Roth Function, Freudenstein and Roth (1963)

$$f(X) = (-13 + x_1 + ((5 - x_2)x_2 - 2)x_2)^2 + (-29 + x_1 + ((x_2 + 1)x_2 - 14)x_2)^2 \quad (15)$$

Initial guess:  $(0.5, -2)^T$

Minimum:  $(5, 4)^T$  or  $(11.41\dots, -8968\dots)^T$

4. Brown's Badly Scaled Function, More *et al.* (1981)

$$f(X) = (x_1 - 10^6)^2 + (x_2 - 2(10^6))^2 + (x_1x_2 - 2)^2 \quad (16)$$

Initial guess:  $(1, 1)^T$  Minimum:  $(10^6, 2 \times 10^6)^T$

5. Beale's Function, More *et al.* (1981)

$$f(X) = (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2 \quad (17)$$

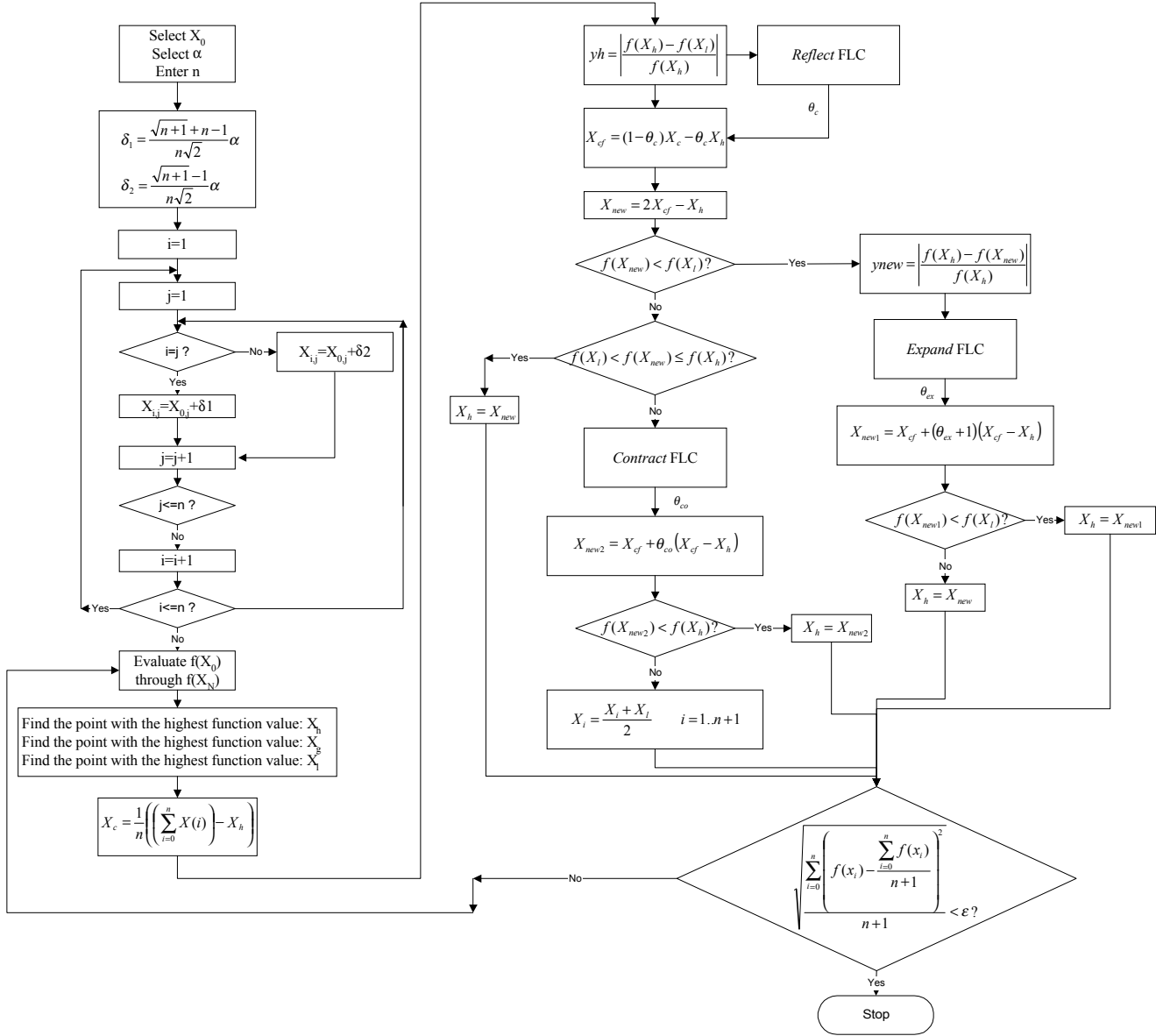
Initial guess:  $(1, 1)^T$  Minimum:  $(3, 0.5)^T$

6. Wood's Function, Rao (1996)

$$f(X) = (10(x_2 - x_1)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10(x_2 + x_4 - 2)^2 + 0.1(x_2 - x_4)^2) \quad (18)$$

Initial guess:  $(-3, -1, -3, -1)^T$  Minimum:  $(1, 1, 1, 1)^T$

$\epsilon$  is equals to  $2.5 \times 10^{-9}$  is used for all cases. To properly compare the algorithms, several original simplex size factor,  $\alpha$ , (Equation 2) are used. These results are listed in Table I through Table VI.



**Figure 3. Flowchart for the Fuzzy Simplex Algorithm**

Three algorithms are applied to each of these functions:

- Nelder and Mead Simplex
- Fuzzy Simplex I (fuzzy expansion and fuzzy contraction; no fuzzy reflection)
- Fuzzy Simplex II (fuzzy reflection, fuzzy expansion, and fuzzy contraction)

**Table I Number of Function Evaluation for Powell's Quartic Function**

Size factor, $\alpha$	Nelder and Mead Simplex	Fuzzy Simplex I	Fuzzy Simplex II
0.25	264	260	231
0.5	216	220	186
1	224	152	189
2	178	182	202
4	270	279	219
Av. No. of Fun. Eval.	230.4	218.6	205.4

**Table II Number of Function Evaluation for Fletcher and Powell's Helical Valley**

Size factor, $\alpha$	Nelder and Mead Simplex	Fuzzy Simplex I	Fuzzy Simplex II
0.25	262	265	263
0.5	410	241	243
1	208	214	207
2	190	167	187
4	164	269	247
Av. No. of Fun. Eval.	246.8	231.2	229.4

**Table III Number of Function Evaluation for Freudenstein and Roth Function**

Size factor, $\alpha$	Nelder and Mead Simplex	Fuzzy Simplex I	Fuzzy Simplex II
0.25	116	101	95
0.5	85	88	89
1	98	91	99
2	86	97	93
4	92	77	81
Av. No. of Fun. Eval.	95.4	90.8	91.4

**Table IV Number of Function Evaluation for Brown's Badly Scaled Function**

Size factor, $\alpha$	Nelder and Mead Simplex	Fuzzy Simplex I	Fuzzy Simplex II
0.25	263	264	268
0.5	296	257	268
1	230	258	258
2	254	218	269
4	269	237	245
Av. No. of Fun. Eval.	262.4	246.8	261.6

**Table V Number of Function Evaluation for Beale's Function**

Size factor, $\alpha$	Nelder and Mead Simplex	Fuzzy Simplex I	Fuzzy Simplex II
0.25	86	77	75
0.5	78	70	64
1	88	77	68
2	74	68	67
4	93	75	67
Av. No. of Fun. Eval.	83.8	73.4	68.2

**Table VI Number of Function Evaluation for Wood's Function**

Size factor, $\alpha$	Nelder and Mead Simplex	Fuzzy Simplex I	Fuzzy Simplex II
0.25	311	283	452
0.5	980	1179	271
1	1032	1289	1343
2	334	345	208
4	887	911	180
Av. No. of Fun. Eval.	708.8	801.4	490.8

Reviewing the results of Table I through Table VI shows that Simplex II generally provides less average number of iterations than Nelder and Mead simplex. The average number of iterations of Simplex I is either of the same order of Simplex II or in between it and Nelder and Mead simplex. The best result is an improvement in the number of function evaluations by 80% for the Simplex II. The average improvement is 4% for Fuzzy Simplex I and 12% for Fuzzy Simplex II over Nelder and Mead algorithm for the six tested functions, which proves that Fuzzy Simplex II represents an attractive substitute for Nelder and Mead algorithm.

To help visualize the search algorithms, consider Figure 4 and Figure 5, which show typical progression of the solution toward a minimum for Beale's function when size factor,  $\alpha$ , is equal to four using Nelder and Mead Simplex and Simplex II respectively. Comparing these two figures shows that the fuzzy reflection succeeded in reducing the number of function evaluations by moving simplex to a point with a lower function value at the fourth iteration. On the other hand, Nelder and Mead Simplex algorithm maintained the simplex points  $(1, 1)$  and  $(0, -3)$  stationary for five iterations, Figure 4. Fuzzy contraction was effective in directing the search toward the minimum at the end of the search.

The instances when the fuzzy Simplex needs more function evaluations to reach the minimum can be explained by several factors:

- i. Fuzzy reflection controller, *Reflect*, may produce values of  $\theta_c$  that is close to zero, which limits its role in the search.



$$\begin{aligned}
\text{subject to, } & g_1(x) = \tau_d - \tau(x) \geq 0 \\
& g_2(x) = \sigma_d - \sigma(x) \geq 0 \\
& g_3(x) = x_4 - x_1 \geq 0 \\
& g_4(x) = P_c(x) - F \geq 0 \\
& g_5(x) = 0.25 - \delta(x) \geq 0 \\
& 0.125 \leq x_1 \leq 10.0 \\
& 0.10 \leq x_2 \leq 10.0 \\
& 0.10 \leq x_3 \leq 10.0 \\
& 0.10 \leq x_4 \leq 10.0
\end{aligned} \tag{24}$$

where,

$$c_3 = 0.10471 \text{ \$/in}^3$$

$$c_4 = 0.04811 \text{ \$/in}^3$$

$$\tau(x) = (\tau'^2 + 2\tau'\tau''\cos\theta + \tau''^2)^{0.5}$$

$$\tau' = \frac{F}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{MR}{J}$$

$$M = F \left( L + \frac{x_2}{2} \right)$$

$$R = \left( \frac{x_2^2}{4} + \left( \frac{x_3 + x_1}{2} \right)^2 \right)^{0.5}$$

$$J = 2 \left( 0.707x_1x_2 \left( \frac{x_2^2}{12} + \left( \frac{x_3 + x_1}{2} \right)^2 \right) \right)$$

$$\cos\theta = x_2 / 2R$$

$$\sigma(x) = \frac{6FL}{x_4x_3^2}$$

$$P_c(x) = \frac{4.013\sqrt{EI\alpha}}{L^2} \left( 1 - \frac{x_3}{2L} \sqrt{\frac{EI}{\alpha}} \right)$$

$$I = \frac{x_4x_3^3}{12}$$

$$\alpha = \frac{Gx_3x_4^3}{3}$$

$$\delta(x) = \frac{4FL^3}{Ex_4x_3^3}$$

The material characteristics are,  $E=30 \times 10^6$  psi,  $G=12 \times 10^6$  psi,  $\sigma_d=30,000$  psi, and  $\tau_d=13,600$  psi. The beam length,  $L$ , is equal to 14 inch while the load,  $F$ , is equal to 6000 pound.

$R$  was assigned value of 100 for all constraints. Constraints  $g_1$ ,  $g_2$ , and  $g_4$  are divided by  $10^4$  to make them of the same order as the other constraints. The initial guess is,  $(1, 7, 4, 2)^T$ . The program converged to,  $(0.25165, 6.2715, 8.2997, 0.24384)^T$ . The final function value is 2.4219, which is reached after 622 objective function and constraint evaluations when  $\alpha$  equal to 2. The method of multipliers (MOM) used 1173 objective function

and 1177 constraint evaluations respectively to reach the minimum, Rekalitis et al. (1983).

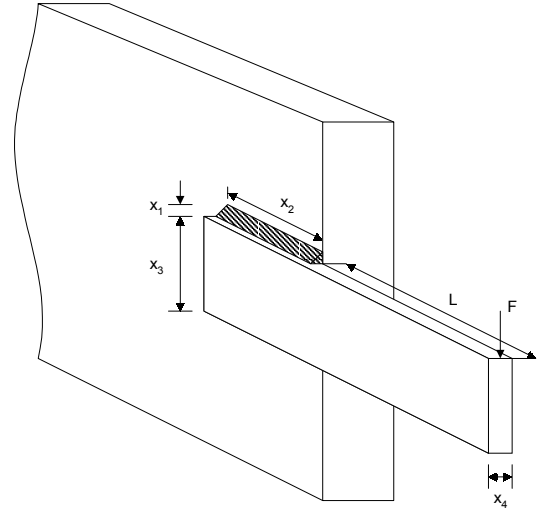


Figure 6. Welded Cantilever Beam

#### 4. CONCLUSIONS

The paper presents an algorithm to minimize a function of  $n$  variables using an adaptive simplex search. The proposed method incorporates the relative weight of the function value at the simplex points in the simplex movement decisions through fuzzy logic, which allows an easy mean for tuning the algorithm based on linguistic logic. The algorithm uses fuzzy logic controller, *Reflect*, to attract the reflection vector toward the point with the lowest function value. The input to this controller is a variable that describes the relative weight of the function values at the points with the highest and lowest function values. The function value of the reflected point is compared with the function values at the simplex points to determine whether to use the reflected point, expand, or contract the simplex. The magnitude of simplex expansion is determined by a fuzzy logic controller, *Expand*, whose input is a non-dimensional variable that describes the relative weight of the function value at the reflected point with respect to the function value at the point with highest function value. Similarly, a fuzzy logic controller, *Contract*, determines the amount of the simplex contraction. The input to *Contract* is the same as the input to *Reflect* but described using different membership sets.

The algorithm is evaluated using several standard functions to compare Fuzzy Simplex I (fuzzy expansion and fuzzy contraction) and Fuzzy Simplex II (fuzzy reflection, fuzzy expansion, and fuzzy contraction) to Nelder and Mead simplex. The results show a general reduction of the number of function evaluations that are needed to reach the minimum, on the order of 12%, when compared to the original Nelder and Mead algorithm. These results can be further enhanced by more systematically tuning the three fuzzy logic controllers. The paper also discusses the performance of the algorithm.

The algorithm is further extended to constrained minimization. The minimum cost of air tank and welded cantilever beam are determined using the proposed algorithm.

The paper shows that fuzzy logic can provide a rational approach for tuning optimization algorithms. The ideas presented here shows that fuzzy logic can be used to increase the efficiency of other pattern search minimization algorithms.

**REFERENCES**

1. F. Freudenstein and B. Roth, "Numerical Solution of Systems of Nonlinear Equations," Journal of ACM, Vol. 10, pp. 550-556, 1963.
2. R. Fletcher and M. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6, pp. 163-168, 1963.
3. C. Lo and P. Papalambros, "A Deterministic Global Design Optimization Method for Nonconvex Generalized Polynomial Problems," Advances in Design Automation, pp. 41-49, ASME, New York, 1990.
4. J. More, B. Garbow, and K. Hillstorn, "Testing Unconstrained Optimization Software," ACM Transactions on Mathematical Software, Vol. 7, pp. 17-41, 1981.
5. E. Mulkey and S. Rao, "Fuzzy Heuristics for Sequential Linear Programming," Proceedings of the 1997 ASME Design Engineering Technical Conferences.
6. J. Nelder and R. Mead, "A Simplex Method for Function Minimization," Computer Journal, Vol. 7, pp. 308-313, 1965.
7. M. Powell, "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives," Computer Journal, Vol. 7, pp. 303-307, 1964.
8. S. Rao, "Engineering Optimization: Theory and Practice," Wiley-Interscience, New York, 1996.
9. S. Rao, "Description and Optimum Design of Fuzzy Mechanical Systems", Journal of Mechanisms, Transmissions, and Automation in Design, pp. 1-7, 1986.
10. V. Rekalitis, A. Ravindaran, and K. M. Ragsdell, "Engineering Optimization: Methods and Applications," Wiley-Interscience, New York, 1983.
11. W. Spendley, G. Hext, and F. Himsworth, "Sequential Application of Simplex Designs in Optimization and evolutionary Operation," Technometrics, Vol. 4, pp. 441, 1962.

**APPENDIX A. REFLECTION FUZZY LOGIC CONTROLLER**

The first question in the process of designing a fuzzy logic controller is related to the choice of membership sets which describes the degree of membership of the linguistic terms that describe a variable. Gaussian curve membership functions are used in this paper. These functions are in the form of,

$$\mu(z, \sigma, c) = e^{-\frac{(z-c)^2}{2\sigma^2}} \tag{A.1}$$

Gaussian curve membership function are chosen since they have the advantage of being described using only two parameters, which makes tuning controllers easier than when using trapezoidal membership functions that are described using four independent variables. These two parameters are  $c$  determines the center of the function while  $\sigma$  controls its shape. The ideas presented here can be easily modified to deal with membership functions of other shapes.

The input and output are described using membership sets as shown in Figure A.1 and Figure A.2 respectively. The range of  $\theta_c$  is selected to be between zero and 0.05, which moves the  $X_h-X_{cf}$  vector slightly toward  $X_l$ . Experimenting with this controller showed that high values of  $\theta_c$  would result in reducing the role of the simplex vertices other than  $X_l$  and  $X_h$ . The controller's rules are,

1. If ( $y_h$  is  $Z_{yr}$ ) then ( $\theta_c$  is  $Z_r$ )
2. If ( $y_h$  is  $M_{yr}$ ) then ( $\theta_c$  is  $M_r$ )
3. If ( $y_h$  is  $B_{yr}$ ) then ( $\theta_c$  is  $B_r$ )

Figure A.3 shows the relation between the input and the output of the controller. The membership sets are chosen in such a way that there are three ranges of  $y_h$  that correspond to 0.0 to 0.5, 0.7 to 0.85, and 0.92 to 1.0. At each of these three ranges,  $\theta_c$  has values around 0.007, 0.03, and 0.048 respectively. The value of  $\theta_c$  gradually increases between these ranges. Figure A.3 shows that significant shifting of the reflection vector occurs at high values of  $y_h$  only.

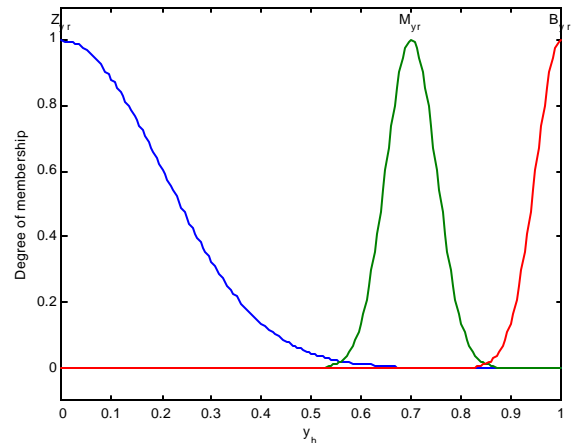


Figure A.3 shows the relation between the input and the output of the Reflect

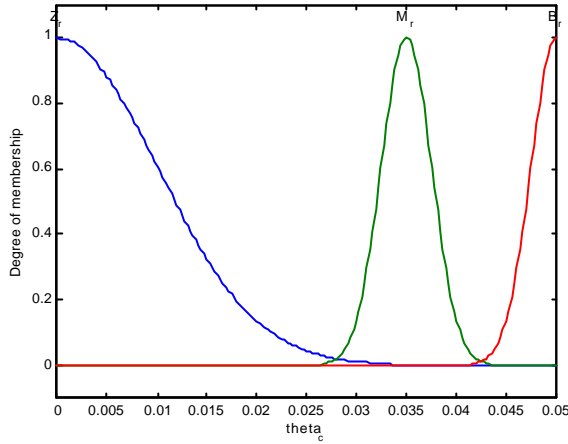


Figure A.2. Membership Sets for the Output of the Fuzzy Controller, *Reflect*

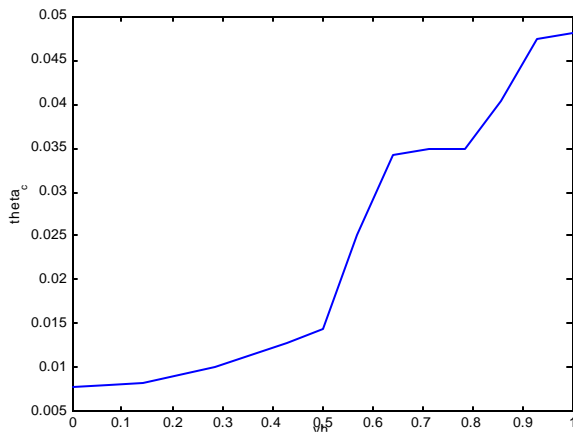


Figure A.3. Relation between the Input and the Output of the Fuzzy Controller, *Reflect*

## APPENDIX B. EXPANSION FUZZY LOGIC CONTROLLER

The input and output of this controller are described using membership sets as shown in Figure B.1 and Figure B.2 respectively. The range of  $\theta_{ex}$  varies around 2.0, which was the best value for this factor according to Nelder and Mead (1965). Larger values of  $\theta_{ex}$  were found to lead the simplex away from the minimum in many cases.

The rules of *Expand* are based on the observation that the least amount of simplex expansion should be when  $f(X_{new})$  is slightly less than  $f(X_h)$ . Similarly, the biggest amount of simplex expansion should be when  $f(X_{new})$  is very small when compared to  $f(X_h)$ . These rules are,

1. If ( $y_{new}$  is  $Z_{ye}$ ) then ( $\theta_{ex}$  is PM)
2. If ( $y_{new}$  is  $M_{ye}$ ) then ( $\theta_{ex}$  is PL)
3. If ( $y_{new}$  is  $B_{ye}$ ) then ( $\theta_{ex}$  is PXL)

The relation between the input and the output of this controller is shown Figure B.3. The membership sets and the rules

are selected such that  $\theta_{ex}$  is equal to 2 when  $y_{new}$  is between 0.1 and 0.9. Outside this range, the value of  $\theta_{ex}$  decreases to 1.8 when  $y_{new}$  is equal to 0.0 and increases to 2.25 when  $y_{new}$  is equal to 1.0. The location and size of the membership sets of the input and output variables of *expand* are tuned using several examples with the objective of reducing the number of function evaluations.

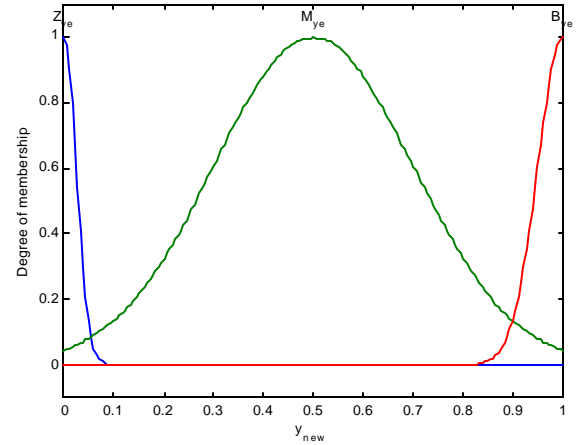


Figure B.1. Membership Sets for the Input of the Fuzzy Controller, *Expand*

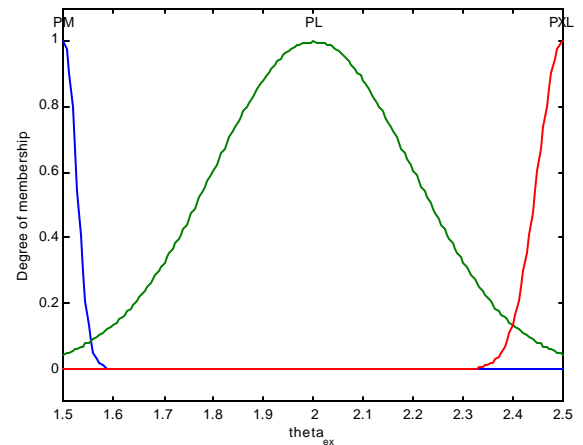


Figure B.2. Membership Sets for the Output of the Fuzzy Controller, *Expand*

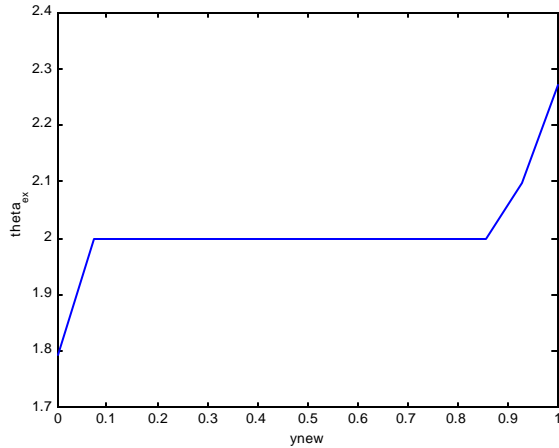


Figure B.3. Relation between the Input and the Output of the Fuzzy Controller, *Expand*

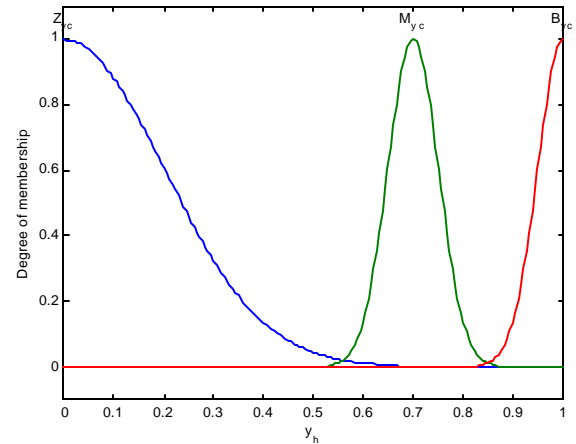


Figure C.1. Membership Sets for the Input of the Fuzzy Controller, *Contract*

### APPENDIX C. CONTRACTION FUZZY LOGIC CONTROLLER

The fuzzy logic controller *Contract* controls contracting the simplex. The input to *Contract* is  $y_h$ , which is defined in Equation 5. The membership sets of  $y_h$ , Figure C.1, are however modified than those of Figure A.1 to fit the purposes of this controller. The output of *contract* is  $q_{co}$  whose membership sets are shown in Figure C.2. The rules of *Contract* are,

1. If ( $y_h$  is  $Z_{yc}$ ) then ( $q_{co}$  is NB)
2. If ( $y_h$  is  $M_{yc}$ ) then ( $q_{co}$  is NM)
3. If ( $y_h$  is  $B_{yc}$ ) then ( $q_{co}$  is NS)

The relation between the input and the output of the controller is shown Figure C.3. This membership sets of the input and output variables to are manipulated based on the following ideas:

- i. If the value of  $y_h$  is low, i.e., the values of  $f(X_h)$  and  $f(X_l)$  are close to each other, it is reasonable to contract the simplex toward its center. Therefore, let the value of  $q_{co}$  be close to  $-0.5$ .
- ii. If the value of  $y_h$  is high, i.e.,  $f(X_h)$  becomes significantly bigger than  $f(X_l)$ , it is reasonable contracts the simplex toward its lowest point. Therefore, let the value of  $q_{co}$  be close to  $-0.4$ .

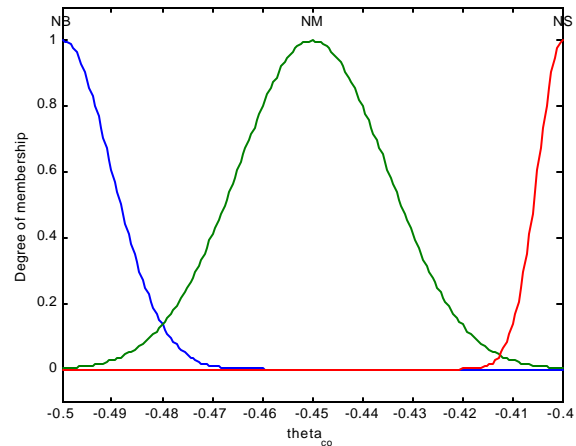


Figure C.2. Membership Sets for the Output of the Fuzzy Controller, *Contract*

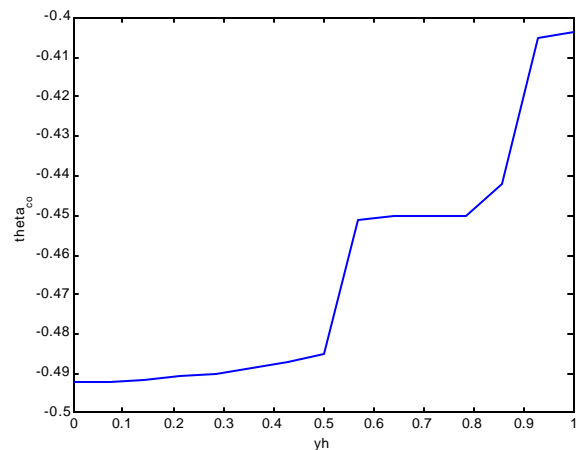


Figure C.3. Relation between the Input and the Output of the Fuzzy Controller, *Contract*