산업기술연구(강원대학교 산업기술연구소 논문집), 제19집, 1999.
Research Report. The Institute of Industrial Technology, Kangwon Nat'l Univ., Korea, Vol. 19, 1999

Nelder-Mead, Dennis-Woods Method와 MATLAB의 FMINS의 비교실험

Comparative Experiment of FMINS with Nelder-Mead and Dennis-Woods Method

최 영 일* 현 창 헌**

Choe, Yeong-Il Hyun, Chang-Hun

Abstract

The Nelder-Mead simplex algorithm has become one of the most widely used methods for nonlinear unconstrained optimization, since 1965. Recently, this algorithm has been reevaluated and many papers on this algorithm are being published. The MATLAB computer software, highly renown in engineering, also provides the Nelder-Mead algorithm and the Denis-Woods modification with the name FMINS function. The authors made C++ code of these algorithms and compared with FMINS on the convergence behavior and the exactness of solutions. It shows that MATLAB's FMINS is inferior to author's C++ code. So, FMINS should be corrected for every user.

키워드 : *Nelder-Mead simplex algorithm, Dennis-Woods modification, MATLAB' FMINS*

1. 서론

현대에서 효율성의 문제는 공학, 경제학, 경영학 등에서 핵심적인 요소가 되고 있다. 효율성의 문제는 곧 최적화문제로 이어지며, 다양한 최적화용 알고리즘을 사용하여 그 해를 구할 수 있다.

최적화를 위한 알고리즘에는 크게 두가지 부류로 대별된다. Gradient method와 direct search method가 그것이다. Gradient method는 수렴속도는 빠르나 비선형문제에 있어 사용하기 어려운 단점이 있다. Direct search method는 함수값만 가지고 최적점을 찾아가는 알고리즘이기 때문에 선형과 비선형문제를 구분하지 않는 장점이 있으나 수렴속도가 느리다는 단점이 있다[1].

본 논문의 주제가 되는 Nelder-Mead Simplex 알고리즘[2]은 가장 유명한 direct search method로서 1965년에 발표된 이후로 현재까지 많은 비선형 최적화 문제를 해결하는데 적용되고 있다. 최근까지 Nelder-Mead Algorithm의 수정안이 계속해서 발표되고 있음은 Nelder-Mead Algorithm의 유용성과 적합성을 말해준다고 할 수 있다. 잘 알려진 공학용 소프트웨어 Matlab에서도 Nelder-Mead Algorithm을 사용한 FMINS함수[3]를 첨가시키고 있어 편리하게 사용할 수 있도록 하고 있다. 하지만 저자가 코딩한 original Nelder-Mead 알고리즘 및 Dennis-Woods의 알고리즘[4]과 이를 비교해본결과, 이렇게 널리 사용되고 있는 FMINS함수의 수행속도와, 해의 정확성에 문제가 있음을 발견하였다. 따라서 이에 대한 엄밀한 검토가 요구된다고 생각한다. 본 연구에서는 목표를 이러한 문제에 두고 수행해 나가기로 한다.

2. Nelder-Mead Simplex 알고리즘

Direct search method는 최적점의 위치를 찾기 위

* 강원대학교 대학원 기계공학과 석사과정
** 강원대학교 기계공학과 교수

하여 단지 함수값의 평가에만 의존한다. 이러한 방법은 미분요소가 필요하지 않으므로 선형문제와 비선형 문제를 구분하지 않고 사용된다. Direct search method에서 Nelder-Mead의 simplex 알고리즘은 가장 유명한 방법으로서[5] stochastic response function의 최적화를 해결하기 위한 Spendley, Hext, Himsworth[6]의 알고리즘을 deterministic 최적화를 위해 수정 개발된 알고리즘이다. Nelder-Mead Algorithm이 발표된 이후 지금까지도 많은 수정 알고리즘이 발표되고 있으며 여러 학문분야에서 사용되고 있다.

이렇듯 Nelder-Mead의 simplex 알고리즘이 유명하게 사용되고 있는 이유를 Lagarias, Reeds, M.H. Wright, P.E. Wright[7]는 다음의 세가지로 요약하고 있다.

1. Nelder-Mead 알고리즘은 최초 반복과정이 수행된 후 단지 몇번의 반복과정으로도 두드러지게 개선된 근사해(또는 최적점에 가까워진 점)에 접근한다.
2. 한번의 반복과정동안 적어도 목적함수의 차원(n) 만큼의 함수계산이 요구되는 방법들은 부단히 시간소모적이다. 만일 이러한 방법들이 성공했다고 가정한다면 이들 방법들 중 Nelder-Mead 알고리즘은 가장 함수평가를 적게 하려한다.
3. Nelder-Mead 알고리즘은 쉽게 설명될 수 있고, 컴퓨터 프로그래밍이 간단하다.

같은 계열로 구분되어지는 방법들 중 Nelder-Mead method는 빠른 수렴속도로 최적점을 찾는다. 이러한 장점으로 최근에는 Genetic 알고리즘과 Simplex 알고리즘의 교접(hybrid)방법[8], 또는 Evolutionary 알고리즘과 Simplex 알고리즘의 교접방법[9]이 소개되고 있다.

2.1 Original Nelder-Mead Simplex Method

Nelder-Mead simplex method는 simplex 꼭지점(vertex)들의 함수값들의 크기를 비교하여 simplex의 크기를 조정하는데 기초를 두고 있다.

(1) 초기화

N개의 변수들의 목적함수에서 초기 N-차원의 simplex를 갖추기 위해 N+1개의 극점을 선택한다. 각 극점에서의 함수값을 계산한다.

(2) 수렴판정조건

Reflection 반복과정은 N+1개의 극점에서의 함수값들의 표준편차가 정해진 미소값(10^{-8})보다 작게 되었을 때, 또는 한계 반복치를 넘어서면 중지한다.

$$SSE = [\ \sum(F(x_i)-\overline{F})^2 / (N+1)\ ]^{1/2},$$
$$\overline{F} \equiv \sum F(x_i)/(N+1)$$

(3) Reflect Worst Point

반복과정이 시작되면 각 극점에서의 함수값들 중에서 가장 높은점, 두번째 높은점, 그리고 가장 낮은점을 확인하여 각각 P_h, P_{sechi}, P_l라 놓고 이 점들에 대응하는 함수값들을 F_h, F_{sechi}, F_l결정한다. 가장 높은점 P_h를 제외한 나머지 극점들의 기하학적 중심점(centroid) P_o를 계산하여 P_h를 P_o방향으로 새로운 극점 P_r를 생성시킨다. 이러한 과정의 reflection은 다음의 식에 의하여 수행되는데, 여기서 α를 Reflection계수라 한다.

$$P_r = (1+\alpha)P_c - \alpha P_h$$

(4) 반전(Reflection)·확대·축소·수축 과정

(4.1) 반전(Reflection) 수용

Reflection된 P_r에서의 함수값을 F_r라 했을때, 만일 $F_l \le F_r \le F_{sechi}$이라면 P_h를 P_r로 대체한 후 새로운 반복과정을 시작한다(restart 2).

(4.2) 확대(expansion) 시도

만일 $F_r < F_l$이라면 같은 방향으로 확장하여도 좋은 결과값이 예상되므로 다음의 식에 의하여 확장된 점을 구한다. 여기서 γ는 확대계수라 한다.

$$P_e = \gamma P_r + (1-\gamma)P_o$$

확장된 극점 P_e에서의 함수값을 F_e라 했을때, 만일 $F_e < F_l$이라면 확장된 점을 수용한다. 즉, P_h를 P_e로 대체한 후 다시 새로운 반복과정을 시작한다. 그런데 $F_e \ge F_l$이라면 확장을 거부하고 이전의 Reflection을 수용한다.

(4.3) 축소(contraction) 시도

만일 reflection된 점의 함수값이 F_h보다 크다면, $F_r > F_h$, 축소를 시도한다. 여기서 만일 $F_r \le F_h$이라면 P_h를 P_r으로, F_h를 F_r으로 대체한다. 축소는 다음의 식에 의하여 수행이 되고 β를 축소계수라 한다.

$$P_e = \beta P_h + (1-\beta)P_o$$

만일 $F_c \leq F_h$이라면 축소를 수용하고 계속해서 반복과정을 수행한다.

(4.4) 수축(shrink)

축소된 새로운 극점의 함수값 F_c가 F_h보다 크다면, $F_c > F_h$, 축소 시도는 거부되고 simplex는 P_l 방향으로 수축을 한다. P_l을 제외한 나머지 극점들은 수축계수 δ에 의하여 수축정도가 결정된다.

$$P_i \leftarrow \delta P_i + (1-\delta) P_l$$

수축된 극점들의 함수값을 계산한 후 다시 반복과정을 수행한다.

2.2 수정 알고리즘

1965년 Nelder-Mead의 simplex 알고리즘이 발표된 이후로 지금까지 많은 수정 알고리즘이 제안되었다. 최근에 발표되는 수정알고리즘들은 deterministic function 최적화를 대상으로 하기보다는 stochastic perturbation을 극복하고자 하는 알고리즘이 대부분을 차지한다. 최근 Y. Huang, W. F. McColl[10]은 simplex reflection을 수행할 때 나머지 극점들의 기하학적 중심점의 연장선상으로 reflection시키지 않고 극점들간의 변의 길이에 대한 비중을 두고 비중에 대한 중심점(weighted-centre)의 연장선상으로 reflection을 시키는 방법으로 효과적인 결과를 얻었다는 보고가 전해진다.

본 연구에서 다루는 수정 알고리즘은 Dennis, Woods의 수렴판정조건에 대한 수정방법이다. Dennis와 Woods는 Nelder와 Mead의 함수값들의 표준편차로써 수렴판정을 했던 것과는 달리 simplex의 크기가 어떤 미소값 ν보다 작게 되면 reflection과정을 중지시키고 수렴하는 방법을 제안하였다.

$$(1/\Delta)\max_i \|P_i - P_l\| \leq \nu,$$
$$\Delta = \max(1, \|P_l\|)$$

여기서 $\|\cdot\|$은 Euclidean norm이며, Dennis와 Woods는 $\nu = 1\times 10^{-4}$을 사용하였다.

2.3 Matlab의 FMINS함수

Matlab의 FMINS함수는 기본적으로 Nelder- Mead 알고리즘을 따르고 있다. 수렴판정조건은 original Nelder-Mead method의 함수값들의 표준편차뿐만 아니라 Dennis, Woods의 simplex 크기에 기초를 두는 수렴판정조건도 함께 사용하고 있다.

3. Computational Experiments

3.1 실험대상 알고리즘

Matlab(ver 5.3)의 FMINS함수와 original Nelder-Mead method, 그리고 Dennis, Woods의 수렴판정조건의 simplex method, Nelder-Mead와 Dennis-Woods의 수렴판정조건이 모두 적용된 Nelder-Mead method를 대상으로 최적해를 찾기까지의 함수 평가수와 수렴된 해의 값, 그리고 좌표값을 중심으로 비교하였다. Table 1에서 이러한 실험대상 알고리즘들과 이하 글에서 사용될 약칭을 정리하였다.

3.2 시험함수(Test Functions)

More, Garbow, Hillstrom[11]은 최적화 알고리즘 및 다양한 수치해석 알고리즘들의 성능을 평가할 수 있는 시험함수들을 수집하였다. 본 연구에서도 FMINS, Nelder-Mead, 그리고 Dennis-Woods의 알고리즘의 성능을 비교하기 위하여 More 등이 수집하여 놓은 18개의 deterministic 함수들을 사용하여 실험을 하였다(Table 2). More 등은 이 함수들과 초기 시작점, 최적해 등을 완전히 정리해 놓았다. 이러한 18개 함수들은 난해한 비구속 최적화 문제를 위한 deterministic 목적함수들이며 NETLIB의 MINPACK Collection[12]을 통하여 이용이 가능하다.

Table 1. 실험 알고리즘의 명칭기호

기 호	설 명
FMINS	Matlab's FMINS function
NM	Original Nelder-Mead Simplex method
DW	Dennis-Woods의 수렴판정조건을 사용한 Nelder -Mead method
NM+DW	Nelder-Mead와 Dennis-Woods의 두가지 수렴판정조건을 모두 사용안 Nelder-Mead method

3.3 실험방법

지금까지 개발되어온 Direct search method 알고리즘들의 가장 단순하면서 기본적인 성능비교 방법은 함수값을 얼마나 많이 계산했는가에 있다. 수렴하기까지의 대부분의 시간은 함수값을 계산하는데 소모되기 때문이다. 따라서 본 실험에서도 FMINS, NM, DW, NM+DW 등의 방법들 간의 함수 평가수를 비교하였다. 또한 simplex의 붕괴에 따른 부적당한 수렴문제

를 고려하여 각각의 방법들에 의해 산출된 최적해 및 최적점을 비교하였다. 모든 알고리즘들의 Reflection 계수(α), 확장계수(γ), 축소계수(β), 수축계수(δ)는 각각 1.0, 2.0, 0.5, 0.5로 일치시켜 실험을 하였다. FMINS와 비교하기 위한 NM, DW, NM+DW 알고리즘들은 직접 C++언어를 사용하여 프로그래밍 하였다.

4. 실험결과 및 고찰

Simplex 알고리즘은 초기 시작점에 민감한 반응을 나타낸다. 초기 시작점의 선택은 수렴된 해의 전역성 또는 국부성, 수렴하기까지의 함수계산수 등에 영향을 준다[2]. 본 연구에서는 More 등[11]이 제안한 초기 시작점(부록의 Table 8 참조)을 사용하여 실험을 수행하였다. 수렴하여 최적해(부록의 Table 10 참조)를 구하기까지의 함수계산수를 정리하면 부록의 Table 9와 같다.

Table 2. MINPACK Test Function

	Function	NOV
1	Helical Valley Function	3
2	Big Exp6 Function	6
3	Gaussian Function	3
4	Powell Badly Scaled Function	2
5	Box 3-Dimensional Function	3
6	Variably Dimensioned Function	4
7	Watson Function	10
8	Penalty 1 Function	2
9	Penalty 2 Function	2
10	Brown Badly Scaled Function	2
11	Brown and Dennis Function	4
12	Gulf Research and Development Function	3
13	Trigonometric Function	2
14	Extended Rosenbrock Function	12
15	Extended Powell Singular Function	4
16	Beale Function	2
17	Wood Function	4
18	Chebyquad Function	2

NOV : 함수의 차원

Table 9에서와 같이 MINPACK의 18개의 함수에 대해서 Original Nelder-Mead method는 FMINS함수보다 12개 함수에서 빠른 수렴을 나타냈고 Dennis-Woods의 수렴판정조건 수정 알고리즘은 10개 함수에서, Nelder-Mead와 Dennis- Woods의 수렴판정조건을 모두 사용한 알고리즘은 11개 함수에서 빠른 수렴을 보였다.

그리고, FMINS에 대한 NM, DW, NM+DW의 상대적인 함수계산수 우위정도(식 1)를 계산하면 Table 3과 같고 이에 대한 평균치를 그래프로 나타내면 Figure 1과 같다.

그림에서 알 수 있듯이 NM, DW, NM+DW 방법들은 FMINS에 대해 상대적으로 평균 20%이상 함수계산수가 적으면서도 해를 도출하고 있다.

$$\text{상대적우위정도} = \frac{\textit{FMINS}\text{의 함수계산수} - \text{평가대상방법의 함수계산수}}{\text{평가대상방법의 함수계산수}} \quad \cdots\cdots\cdots\cdots (1)$$

Table 3. 함수계산수에서 FMINS에 대한 다른 방법들의 상대적 우위정도 (단위 %)

함수번호	NM	DW	NM+DW
1	-39.09	-41.73	-41.73
2	15.51	9.96	9.96
3	-36.08	-45.61	-45.61
4	102.29	-11.67	-11.67
5	67.25	51.42	51.42
6	70.64	69.09	69.09
7	119.22	25.88	25.88
8	-61.05	-73.04	-73.04
9	-1.45	-15.00	-15.00
10	1.85	43.23	14.11
11	16.03	69.04	54.88
12	62.43	253.46	253.46
13	0.00	-18.67	-18.67
14	12.12	11.20	11.20
15	22.98	-22.19	-22.19
16	46.58	42.67	42.67
17	79.25	79.86	79.86
18	-0.02	-4.48	-4.48

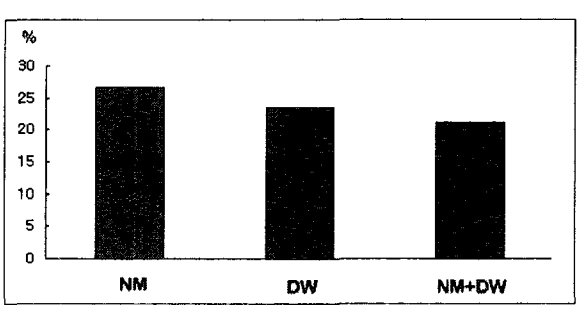

Figure 1. FMINS에 대한 평균 우위정도

MINPACK의 Test Function(Table 2 참조)을 대상으로 축소계수(β) 변화에 대한 함수계산수와 오차의 변화를 관찰한 결과 빠른 수렴은 오차의 증가를 유발한다는 것을 알 수 있었다. Box 3D 함수는 그 전형적인 예이다. 구체적으로는, Figure 2에 보여지고있다. 축소계수(β)가 0.35부근에서 함수계산수는 갑자기 높아지고 있는 반면 오차는 급격히 줄어들고 있

다.

NM, DW, NM+DW 방법들이 FMINS보다 상대적인 빠른 수렴으로 인한 오차의 증가를 예측하였으나 오히려 FMINS가 오차의 분포가 매우 넓게 나타났다. Figure 3은 이러한 오차값의 절대치 분포를 보여준다. Figure 3에서 함수 14(Extended Rosenbrock function)에 대한 절대치 오차의 분포는 생략되었다. Table 10에서와 같이, FMINS로 이 함수의 해를 구했을 때의 오차(14.3162)가 나머지 알고리즘의 사용으로 구한 해의 오차들(2.5080)에 비해 상대적으로 매우 크다. 하나의 동일 그림에 모든 오차들을 표현하기가 어려워 생략되었다. Figure 3에서 알 수 있듯이 FMINS의 오차분포는 나머지 오차분포 보다 5배 이상 넓게 분포하고 있다.

함수 2, 5, 7, 14 에서 FMINS 함수는 수렴해와 그 지점의 좌표에서 큰 오차가 발생하였으며, 특히 함수 5 에서는 함수의 좌표가 완전히 다르게 산출되었고 최적점의 함수값도 오차가 심했다. Figure 4는 이러한 함수 2, 5, 7, 14의 수렴해에 대한 LOG 오차의 분

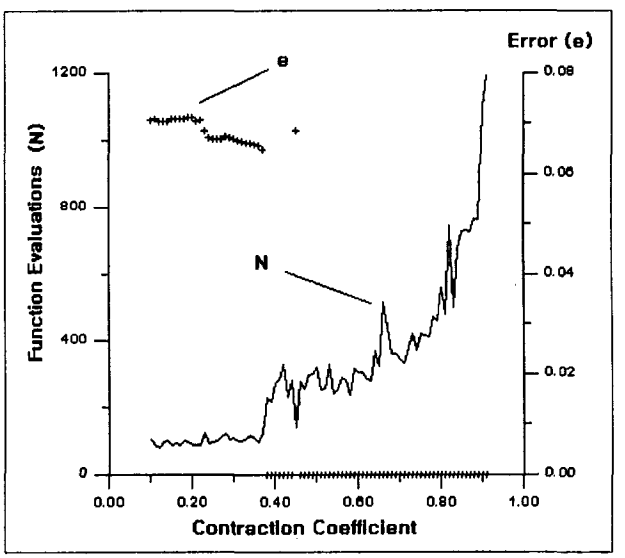

Figure 2. Box 3D 함수에 대하여 축소계수(β)의 변화에 따른 함수계산수(N)와 오차(e)의 변화

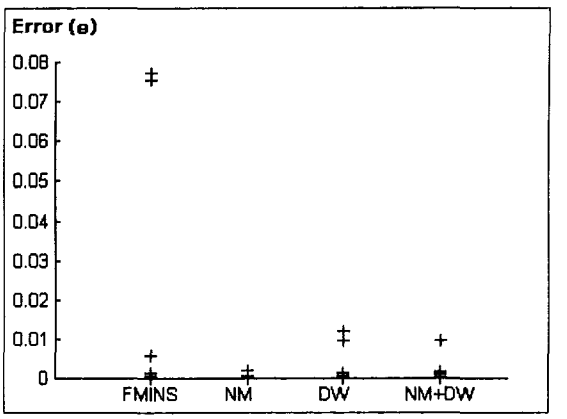

Figure 3. 절대오차의 분포

포를 보여주고 있다. 특히 함수 14를 제외한 경우, FMINS함수에 의한 오차들은 나머지 모든 방법들의 오차보다 위에 분포하고 있다. 함수 14, 즉 Extended Rosenbrock 함수의 경우 4개의 모든 알고리즘들이 진해(眞解)인 0에 가깝게 접근 하지 못했다. 그러나 상대적으로 FMINS가 다른 방법들보다 오차가 큰 함수값에서 수렴을 하였다. 함수 14의 진해(眞解)에 대한 각 수렴좌표들의 표준편차를 비교하면 Table 7과

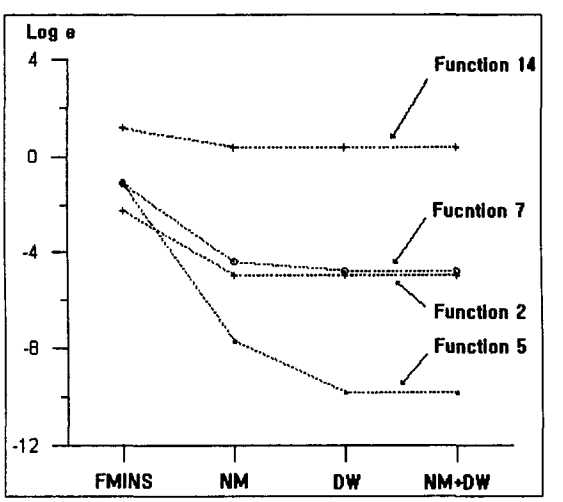

Figure 4. 함수 2, 5, 7, 14에 대한 수렴해의 LOG 오차분포

Table 7. 함수 14 비교

방법	함수계산수	수렴해	진해에 대한 표준편차
FMINS	6125	1.4316e+01	1.6424
NM	5463	2.5080e+00	0.6760
DW	5508	2.5080e+00	0.6760
NM+DW	5508	2.5080e+00	0.6760

같다. FMINS함수는 진해(眞解)에 대한 수렴 좌표값인 (0, 0, ..., 0)에 대한 표준편차가 나머지 방법들의 것보다 2.4배 이상 큰 값인 것을 알 수 있었다. 따라서 수렴된 함수값 뿐만 아니라 좌표값도 오차가 큰 것으로 나타났다.

함수 8, 즉 Penalty 1 함수인 경우 수렴하기까지의 함수평가수는 FMINS함수가 가장 적었으나 모든 방법들이 최적점의 좌표에 대한 오차는 크게 나타나고 있는 가운데 NM만이 수렴해의 좌표가 다르게 산출되었다. 2차원의 Penalty 1 함수는 진해(眞解)인 (0, 0) 근방에서 함수의 최소값의 차이가 거의 없는 부분이 넓게 분포하고 있다. Figure 5는 2차원의 Penalty 1 함수에 대한 그래프이다. 그림에서도 확인할 수가 있듯이 NM에 의해 구해진 최적점과 나머지 알고리즘에 의해 구해진 최적점을 구분한다는 것은 의미가 없음을 알 수 있었다.

모든 함수에 대해서 FMINS함수의 결과에 대한 오차가 큰 것은 아니었다. 함수 10, 12에 대해서는 FMINS함수의 결과가 다른 방법들의 결과보다 오차면에서는 우수하였지만 수렴되기까지의 함수계산수는 나머지 방법들에 비해 크게 나타났다. Figure 6은 이들 함수에 대한 LOG오차분포와 함수계산수를 도식한 그래프이다. 이 경우는 함수 2, 5, 7, 14에서 NM, DW, NM+DW가 오차와 함수계산수에서 FMINS의 결과보다 모두 우수했던 것과는 달리 수렴해 또는 수렴속도 등 어느것에 가치를 두느냐에 따라 우열이 나누어지므로 평가가 모호해 지는 경우이다.

지금까지의 실험을 토대로 판단하면 FMINS는 NM, DW, NM+DW보다 일반적으로 수렴속도면에서

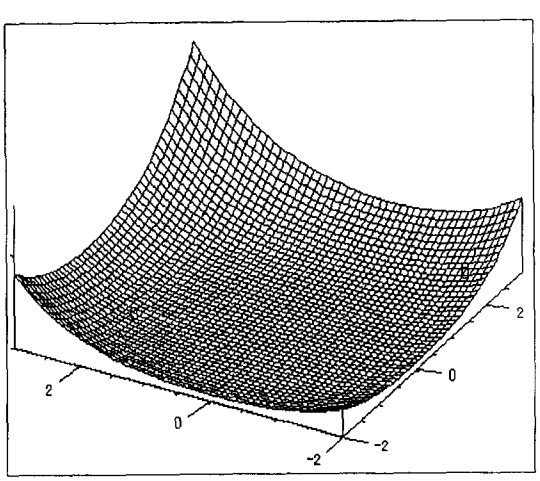

Figure 5. Test Function 8. Penalty 1 Function

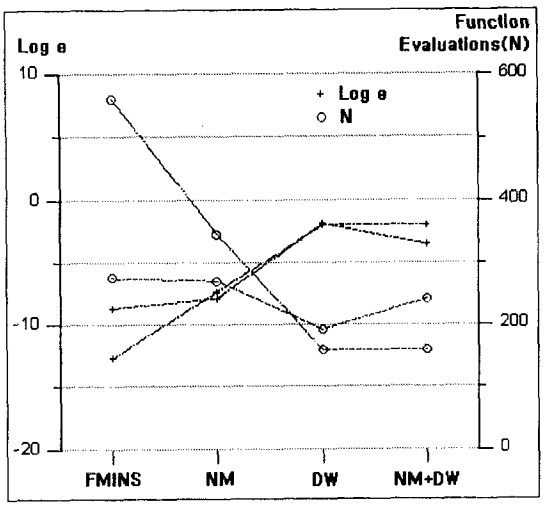

Figure 6. 함수 10, 12에 대한 수렴해와 수렴해에 대한 LOG 오차분포

열등한 것으로 나타났으며, 수렴한 결과인 최적값의 오차의 분포범위가 넓어 오차의 크기면에서도 열등하게 나타남을 알 수가 있었다.

5. 결 론

비선형 비구속 최적화문제를 위한 가장 많이 사용되어지는 Nelder-Mead의 Simplex 알고리즘이 공학분야에 많이 사용하고있는 Matlab (MathWorks사의 제품)에서도 FMINS 함수 이름으로 제공되고 있다. 이 함수의 사용시 문제가 있음을 발견한 저자는, 이의 수렴속도의 적합성 및 해의 정확성 평가가 요구된다고 생각되어 FMINS함수와 더불어 original Nelder-Mead Algorithm과 몇몇 수정 알고리즘 (Dennis-Woods, 그리고 Nelder-Mead 와 Dennis-Woods을 합친 알고리즘)과 함께 MINPACK의 Test Function으로써 비교·검토한 결과 다음의 결론을 얻었다.

첫째, 수렴하기까지의 상대적인 함수계산수를 산출한 결과 FMINS의 비교 대상이 되었던 방법들이 평균 20%이상 함수계산을 적게 하면서도 수렴을 하고 있다.

둘째, 수렴한 함수값을 비교해본 결과 FMINS함수의 수렴해의 값이 나머지 방법들의 결과보다 부정확했다. 특히 Bigs, Box 3D, Watson, Extended Rosenbrock 함수에 대해서 부정확했다.

수렴된 해의 좌표값의 비교에서는, 특히 Box 3D 함수인 경우가 상당히 달랐고 Extended Rosenbrock 함수에서는 FMINS함수가 상당히 열등한 것을 알 수

있었다.

셋째, FMINS함수의 경우 오차의 범위가 넓게 분포하고 있었다. 오차의 범위가 넓다는 것은 신뢰도가 낮다는 것을 의미하므로 FMINS함수의 결과를 항상 신용할 수는 없다.

종합적으로, FMINS는 Nelder-Mead, Dennis-Woods, 그리고, Nelder-Mead 와 Dennis-Woods을 합친 알고리즘보다 일반적으로 수렴속도면에서 열등하다. 그리고, 수렴한 결과값들의 오차의 분포범위가 넓어 해의 정확성의 신뢰도가 낮다.

따라서 FMINS함수는 주의깊게 사용하여야 하며 다른 방법들에 의한 결과와 항상 비교할 필요가 있다.

부 록

Table 8. Test Function과 초기시작점

Function	NOV	Starting Point
1	3	-1, 0, 0
2	6	1, 2, 1, 1, 1, 1
3	3	0.4, 1, 0
4	2	0, 1
5	3	0, 10, 20
6	4	-3, -7, -11, -15
7	10	0, 0, 0, 0, 0, 0, 0, 0, 0, 0
8	2	1, 2
9	2	0.5, 0.5
10	2	1, 1
11	4	25, 5, -5, -1
12	3	5, 2.5, 0.15
13	2	0.5, 0.5
14	12	-1.2, 1, -1.2, 1, -1.2, 1, -1.2, 1, -1.2, 1, -1.2, 1
15	4	3, -1, 0, 1
16	2	1, 1
17	4	-3, -1, -3, -1
18	2	1/3, 2/3

NOV : 함수의 차원

Table 9. 수렴하기까지의 함수계산수

함수번호	FMINS	NM	DW	NM+DW
1	148	243	254	254
2	916	793	833	833
3	62	97	114	114
4	265	131	300	300
5	480	287	317	317
6	372	218	220	220
7	1688	770	1341	1341
8	141	362	523	523
9	68	69	80	80
10	275	270	192	241
11	333	287	197	215
12	562	346	159	159
13	61	61	75	75
14	6125	5463	5508	5508
15	305	248	392	392
16	107	73	75	75
17	527	294	293	293
18	64	65	67	67

Table 10. 수렴한 함수값

함수번호	FMINS	NM	DW	NM+DW
1	3.2623e-09	1.9217e-08	7.9457e-10	7.9457e-10
2	5.7000e-03	1.1750e-05	1.1724e-05	1.1724e-05
3	1.1889e-08	3.9591e-08	1.1362e-08	1.1362e-08
4	9.9980e-01	9.9979e-01	9.9979e-01	9.9979e-01
5	7.5600e-02	2.0215e-08	1.5680e-10	1.5680e-10
6	4.0441e-09	2.6772e-08	6.8502e-09	6.8502e-09
7	7.7200e-02	4.0421e-05	1.7508e-05	1.7508e-05
8	8.3578e-09	8.9467e-06	8.3578e-06	8.3578e-06
9	8.0787e-07	9.2113e-07	8.0792e-07	8.0792e-07
10	2.0036e-09	1.2222e-08	1.1981e-02	2.8058e-04
11	8.5822e+04	8.5822e+04	8.5822e+04	8.5822e+04
12	1.8939e-13	4.3361e-08	9.4539e-03	9.4539e-03
13	9.5856e-10	5.5425e-08	4.5341e-10	4.5341e-10
14	1.4316e+01	2.5080e+00	2.5080e+00	2.5080e+00
15	1.3906e-06	4.8190e-08	5.5343e-15	5.5343e-15
16	1.3926e-10	3.9599e-09	2.7089e-09	2.7089e-09
17	1.9448e-09	2.3535e-08	1.1148e-08	1.1148e-08
18	1.9599e-09	9.3503e-09	2.5168e-09	2.5168e-09

참고문헌

[1] Swann, W. H., "Direct Search Methods," in W. Murray(Ed.), *Numerical Methods for Unconstrained Optimization*, Academic Press, London, 1972.

[2] Nelder, J. A. and R. Mead, "A Simplex Method for Function Minimization," *Computer J.*, 7 (1965), 308-313.

[3] *MATLAB user's guide,* Prentice Hall, Englewood Cliffs, N. J. 1995.

[4] Dennis, J. E., Jr. and D. J. Woods, "Optimization on Microcomputers: the Nelder-Mead Simplex Algorithm," in *New Computing Environments: Microcomputers in Large Scale Computing,* A. Wouk(Ed.), SIAM, Philadelphia, PA, 1987, 116-122.

[5] Barton, Russell R, Ivey Jr, John S., "Nelder-Mead simplex modifications for simulation optimization.", *Management Science*, Jul 96, vo l42 Issue7, p954, 20p.

[6] Spendley, W., G. R. Hext, and F. R. Himsworth, "Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation," *Technometrics,* 4 (1962), 441-461.

[7] J. C. Lagarias and J. A. Reeds and M. H. Wright and P. E. Wright, "Convergence properties of the Nelder-Mead simplex algorithm in low dimensions", *SIAM Journal on Optimization*, vol. 8, 1998.

[8] J. Yen, J. C. Liao, B. Lee, D. Randolph, "A Hybrid Approach to Modeling Metabolic Systems Using a Genetic Algorithm and Simplex Method," *IEEE Transactions on Systems, Man, and Cybernetics, Part B:Cybernetics,* vol 28, no 2, April 1998.

[9] Smith, Stephen, "Simplex method and evolutionary algorithms", *IEEE International Conference on Evolutionary Computation*, 1998, pp. 799-804.

[10] Huang, Yuguang; McColl, WF, "Improved simplex method for function minimization", *IEEE International Conference on Systems, Man and Cybernetics*, 1996, vol. 3, pp. 1702-1705.

[11] More, J. J., B. S. Garbow, and K. E. Hillstrom, "Testing Unconstrained Optimization Software," *ACM Trans. Math. Software,* 7 (1981), 17-41.

[12] Dongarra, J. J. and E. Grosse, "Distribution of Mathematical Software via Electronic Mail," *Comm. ACM,* 30 (1987), 403-407.