論 文

最適 設計를 위한 3점 탐색 알고리즘의 제안

正會員 金 周 弘* 正會員 孔 徽 植**

A Proposal of 3 Point Search Algorithm for Optimal Design

Ju Hong KIM*, Whue Sik KONG** *Regular Members*

要 約 最適 設計를 위한 최적치 탐색 알고리즘으로 直接 探索法의 일종인 3점 탐색 알고리즘을 제안하였다.

본 알고리즘은 N차원 탐색범위 내에 있는 數空間의 3^N점에서 함수의 최소치를 탐색하고, 점차로 탐색범위를 축소하여 동일한 탐색 과정을 반복 수행하는 방법이다. 그러므로, 1회 탐색시에 성능지표의 계산횟수는 3^N(N는 매개변수의 수)이다. 또한 3^N점 탐색법을 대신한 3N점에 대한 탐색법으로 단순 3N점 탐색법을 기술하였으나, 이것은 서로 다른 매개변수가 乘除項을 갖는 성능지표의 경우에는 불확실함이 발견되었다.

제안된 알고리즘은 2차 형식이나 선형 함수로 구성되는 성능지표에 적용이 가능하며, 안정하고 신뢰도가 높은 특성을 갖고 있음이 확인되었다.

ABSTRACT In the paper, the three-point search algorithm used direct search method for optimal design is proposed.

Proposed algorithm is composed of two iteration procedure to determine the minimum value of performance index. The minumum value of three-point existing in the inner N-order searching is firstly determined and next searching space is then reduced by the result of first procedure. To compute performance index, 3^N iteration for a searching is necessary. 3N searching method is also described and confirmed with exception of parameter included multiplicator and divisor.

Proposed algorithm have good stability and reliability when perfomance index is linear or quadratic function.

I. 서 론

계통(system)이나 장치의 설계에 있어서 그 성능(성능지표)의 최적치(max 또는 min)는 수치계산에서 막강한 능력을 갖는 컴퓨터의 출현과 극치문제에 관한 수학적 이론 및 알고리즘의 개발로 최적설계의 실현을 가능하게 하였다.

최적 설계 과정에서 가장 중요한 것은 매개변수(parameter)의 설정에 있으며, 주어진 성능지표가 최대 또는 최소로 되도록 매개변수의 값을 결정하는 문제이다[7-9].

최적치 문제에 관한 알고리즘은 gradient descent 법[6-10]과 search(탐색)법[3-5]으로 대분되고, 전자는 성능지표의 gradient(기울기 또는 경도)와 기울기의 방향을 산출하여 최적치를

*東國大學校 電子工學科
Dept. of Electronics Engineering Dongguk University
**東宇專門大學 電子計算科
Dept. of Computer Science Dong-u Junior College
論文番號 : 91-60(接受1991. 2. 26)

구하는 방법으로, 계산식이 많아서 프로그램이 복잡하고, 안정도도 낮으나 성능지표를 계산하는 횟수가 적은 장점을 갖고 있다[3-4]. 후자는 성능지표만을 계산하여 최적치를 탐색하는 방법이므로 계산의 반복횟수가 많아지는 단점은 있으나, 계산은 간단하게 이뤄지고, 안정도가 높으며, 프로그램이 용이한 장점이 있다[4].

이중에서 탐색법에는 단일 변수를 갖는 성능지표의 탐색법으로 탐색구간을 2등분하여 최적치를 탐색하는 dichotomous 탐색법[3]과 golden section 탐색법[4]이 있으며, 이들은 알고리즘이 단순하나 최적치를 갖는 성능지표가 단일 변수로 구성되어야 하는 문제점이 있으며, 다차원 탐색법은 Hooke 와 Jeeves가 제안한 pattern 탐색법[3-5], Lawrence와 Steiglitz의 randomized pattern 탐색법[2], Rosenbrock의 직접 탐색법[3] 및 Burhardt의 ASA[1]등 많은 알고리즘이 발표되어 있으나, 스텝사이즈(step size)와 탐색의 방향(direct)을 결정하기 위한 계산이 복잡하고, sharp corner, curving vally, ridge에서의 처리와 신뢰도 및 적용 가능한 함수등에서 각각 장단점을 갖고 있다.

본 논문에서는 최적 설계분야에서 가장 높은 빈도로 나타나는 선형 및 2차형식을 갖고, 단일 변수 또는 다변수로 구성되는 성능지표의 최적치 탐색을 위한 직접 탐색 알고리즘으로 3점 탐색법을 제안하고, 프로그램을 작성하여 성능을 검토하였다.

Ⅱ. 3점 탐색 알고리즘의 제안

Ⅱ.1 3^N점 탐색 알고리즘의 제안

통신계나 전기계 및 기계계나 장치등의 최적 설계에 적용되는 성능지표(performance index 또는 cost function) J는 일반적으로 식(2.1)과 같은 2차형식함수(quadratic function)로 구성되는 경우가 많다.

$$J=X\cdot R\cdot x,\quad x=[x_1,x_2,\cdots,x_n] \tag{2.1}$$

여기서, x는 설계에 필요한 독립변수 $x_1,x_2,\cdots,x_n$로 구성된 벡터(vector)이며, 변수가 n개일 때는 n차의 벡터이고, R은 n×n개의 변수를 갖는 정정행렬(positive definition matrix)이다. 그리고, 취급한 함수는 설정한 탐색범위 내에서 極値가 1개만 존재하는 unimodality 형인 경우의 단일변수와 다변수(4개 변수)로 된 성능지표이며, 적용하는 3점 탐색법은 다음과 같다.

편의상 탐색하려는 벡터 x가 1차인 경우와 2차인 경우에 대하여 설명하고, 3차 이상인 다차의 경우는 類推 適用하기로 한다.

(1) 1차 벡터인 경우

함수(성능지표) J의 변수 x가 1차 $x=[x_1]$인 경우이며, 그림 1과 같이 탐색의 반복횟수를 w라하고, 각 회수에 따라 설명하면 다음과 같다.

(가) 초기탐색(w=0)

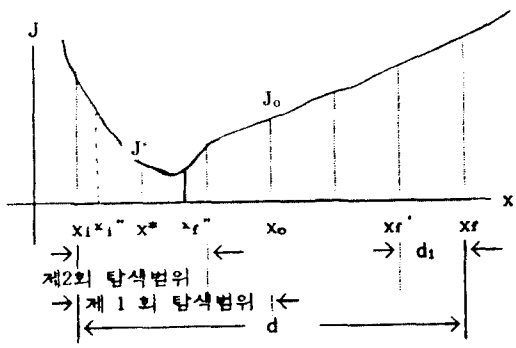

그림 1. 1차원 함수의 탐색법.
Fig 1. Search algorithm of one dimensional function.

그림 1과 같이 탐색범위를 x_l, x_f로 하고, 그 크기를 $d=x_f-x_l$로 표시하면 그 중앙점 x_0에서 성능지표 J를

$$J_0=f(x_0) \tag{2.2}$$

로 계산하여서, J_0와 x_0를 기억시킨다.

(나) 제1회 계산(w=1)

제1회의 分割區間

$$d_1 = \frac{d}{2} \times \frac{1}{3} \tag{2.3}$$

를 계산하고, x_0부터 $\pm 2d_1$점을 각각 x_1', x_f'로 표시하여, $J_1=f(x_1)$과 $J_f=f(x_f)$을 계산하여 J_0, J_1및 J_f의 세 값에서 최소인 것을 $J^*=f(x^*)$로 기술하고, J^*및 x^*를 기억시킨다.

(다) 제2회 계산(w=2)

다시, 분할구간 d_2

$$d_2=\frac{d_1}{3} \tag{2.4}$$

을 계산하고, x^*점을 중심으로 $\pm 2d_2$의 곳에 위치한 x_1'', x_f''의 점에서 J의 값 $J_1=f(x_1'')$과 $J_f=f(x_f'')$를 계산하여서 J^*와 J_1, J_f를 비교하여 이들 중에서 최소치 J를 $J^*=f(x^*)$로 기술하고 J^* 및 x^*를 기억시킨다. 그리하여, 제1회 탐색범위의 1/3로 축소된 탐색범위 내에서 탐색하게 된다.

(라) 3회이상의 계산(w>3)

제i번째의 탐색은 분할구간

$$d_i=\frac{d_{i-1}}{3}=\frac{d}{2\cdot 3^i} \tag{2.5}$$

이므로, 전항과 같은 방법으로 J의 계산을 반복하여서, 설정한 계산횟수로 종료한다.

(다)항의 방법을 반복하여 J^*를 계산하였을 때 J의 총 계산회수 y는

$$y=3 \times w \tag{2.6}$$

이며, 探索精度(즉, 최종 계산시의 探索範圍)는

$$d_w=\frac{d}{2}\times\left(\frac{1}{3}\right)^w = \frac{d}{2\times 3^w} \tag{2.7}$$

으로 w=10인 경우에는 $8.467\times10^{-6}\times d$가 되어 精密한 탐색정도가 얻어진다.

특히, 최소치 x^*가 경계점 x_1나 x_f상의 極限점에 위치하는 경우에 x가 다음 식(2.8)과 같은 반복계산에 의하여 x_0로 부터 출발하여서 x_f에 收斂함을 알 수 있다. 즉, 최초의 중심점 x_0로 부터, w회의 반복계산으로 도달 할 수 있는 거리 x_d는

$$x_d=\frac{d}{2}\times\left(\frac{2}{3}+\frac{2}{3}+\cdots+\frac{2}{3^w}\right) = \frac{d}{2}\cdot\frac{2}{3}\cdot\left(\frac{1-(1/3)^w}{1-(1/3)}\right) \tag{2.8}$$

이고, $w=\infty$일때 $x_d=(d/2)$로 되어서, x_1점 또는 x_f점에 도달함을 알 수 있다.

이상의 과정을 검토하면, 탐색범위로 설정한 구간내의 모든 점에 접근하여 최소값의 탐색이 가능하고, 탐색구간을 벗어나는 경우는 존재하지 않으므로, 정확한 값을 구할 수 있어 신뢰도가 인정되고, 안정한 상태의 유지가 가능하다.

(2) 2차 벡터의 경우

독립변수가 2개인 경우에 x점은 그림 2에서 평면상에 표시되므로

$$x=[x_1,\ x_2] \tag{2.9}$$

이 평면상의 점에 대응한 J의 값에서 최소치를 구하는 문제가 된다.

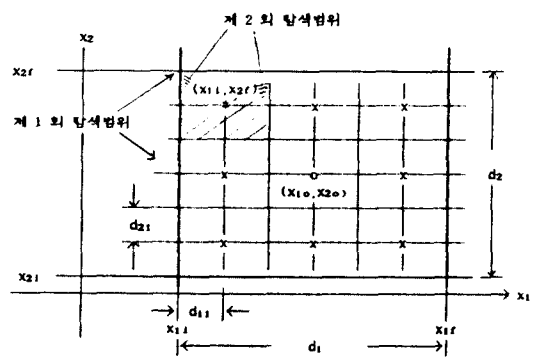

그림 2. 2변수 벡터의 탐색과정
Fig 2. A search procedure of 2-variable vector

(1)에서 설명한 것을 2차원으로 擴張하여 탐색범위 x_{1l}, x_{2l}와 x_{1f}, x_{2f}의 평면 사각형의 중앙점(그림 3에서 ○점으로 표시) x_{1o}, x_{2o}와 그림 2에서 **x**로 표시된 8개의 점을 포함한 9점에서 **J**를 계산하고, 그 중에서 **J**의 최소치

$$\mathbf{J}^*=f(x_1^*,\ x_2^*) \tag{2.10}$$

를 탐색하여 $\mathbf{J}^*$ 및 x_1^*, x_2^*를 기억시키고 제1회(w=1)의 탐색을 마친다.

제2회는 $\mathbf{J}^*$를 중심으로 하여 길이가 1/3로 축소된 탐색범위에서 같은 방법으로 최소치를 계산하여서, 설정된 w회의 반복계산을 한다.

(3) n차 벡터의 경우

탐색변수를 식(2.11)의 벡터로 표시하고,

$$\mathbf{x}=[x_1,\ x_2,\cdots,x_n] \tag{2.11}$$

이에 대응하는 탐색법은 (2)절의 방법을 유추 적용하여 축차적으로 범위를 축소하여 설정한 반복횟수 w에서 종료한다.

이때, 성능지표의 계산횟수 y는

$$y=3^N\cdot w \tag{2.12}$$

이고, 精度 Z(최후의 탐색범위)는

$$Z_k=\frac{d_k}{2}\cdot\left(\frac{1}{3}\right)^{w-1} \tag{2.13}$$

이다. 여기서 d_k는 **x**의 k번째 성분에 대한 초기 탐색범위의 길이이다.

Ⅱ.2 단순 3점 탐색 알고리즘의 제안

Ⅱ.1절의 알고리즘은 1회 탐색시에 **J**의 계산횟수가 3^N번 수행되므로 탐색변수인 벡터 **x**의 차수가 커지면 N승에 비례하여 막대한 계산을 필요로 하는 알고리즘의 단점이 있다. 특히 실시간 제어계에 이용하는 경우는 계산시간이 많이 소요되어 적용이 불가능한 경우가 발생할 수 있으므로 계산 횟수를 줄이기 위한 3N점 탐색법을 아래에 제시한다.

앞의 (2)항에서 2차벡터인 경우에 $3^2=9$점 대신 $3\times2=6$개의 점을 탐색하는 방법으로 N차원인 계에서는 $3\cdot N$개의 점에서 탐색을 수행하게 되어 계산횟수가 현저하게 감소하는 방법으로 그림 3과 같은 위치에서 탐색하되, 이것을 다차원에 유추 적용시킨 방법이다.

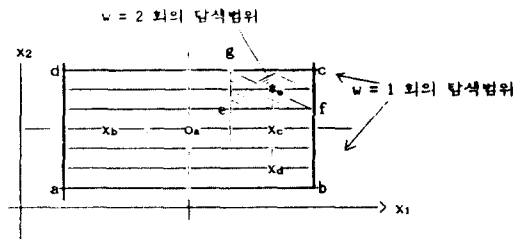

그림 3. 단순 3N점 탐색법의 설명도
Fig 1. Description of simple 3N point search algorithm

그림 3에서 탐색범위의 중앙점 o_a에서 **J**를 구하고, x_b점과 x_c점의 **J**값과 비교하여, 그 중에서 최소치 $\mathbf{J}^*$를 구하고(그림 3에서는 x_o점으로 가정), 이 점을 통과한 x_2축에 수직인 점에 대하여서 같은 방법으로 3점(x_d, ✳$_e$점)에 대한 **J**의 최소치를 구하고(그림 3에서 ✳$_e$점으로 표시), 이들 $\mathbf{J}^*$와 $\mathbf{x}^*$값을 기억시킨다. 그림 3에서 사각형 abcd는 제1회 탐색범위이고, 사각형 cefg는 제2회의 탐색범위이며, 이와같은 과정을 반복 실행하는 알고리즘이다. 탐색을 1회 수행하는 동안의 계산횟수는 3N회로 계산횟수가 현저하게 줄어드는 장면이 있다.

이 2개의 제안한 알고리즘을 구분하기 위하여 Ⅱ.1절을 3^N점 탐색법, 그리고 후자 Ⅱ.2절을 3N점 탐색법으로 명명하기로 한다.

Ⅲ. 성능의 비교 검토

본 알고리즘은 계산과정에서 벡터 **x**의 어느 한 성분만을 변수로 취급하고, 다른 변수는 상수로 취하여서 반복계산하는 소위 univariate 탐색

법에 속한 것이다.

3^N점 탐색법을 수행하기 위한 주프로그램의 신호 흐름도는 그림 4와 같고, 부프로그램의 신호 흐름도는 그림 5이다. 벡터 x의 탐색범위를 설정하여 행렬 Sr의 초기치로 입력하고, 성능지표계산은 부프로그램으로 작성하고, $J^*=f(x_i^*)$의 탐색은 그 차원 수 N에 따라 해당하는 부프로그램을 작성하였다.

제안한 알고리즘을 Turbo C 언어로 프로그램을 작성하고 IBM PC/AT 및 386기종으로 실행하였으며, 결과를 기존의 다른 탐색법과 그 성능을 비교 검토한 것이다.

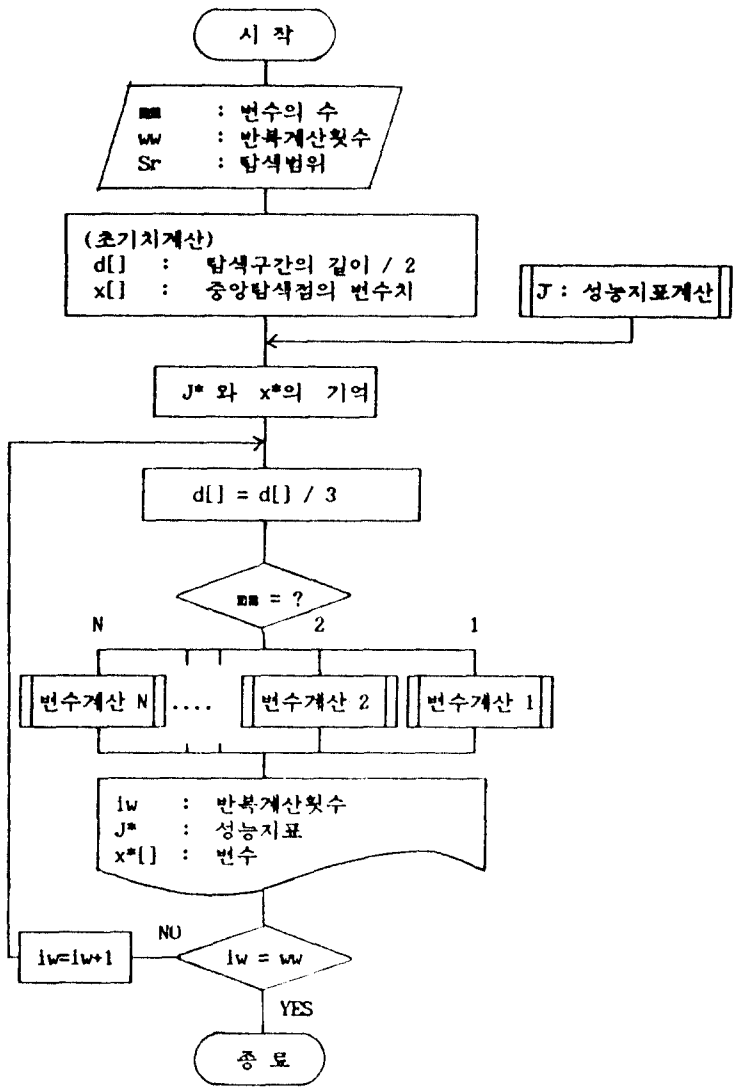

그림 4. 제안 알고리즘의 주 프로그램 흐름도
Fig 4. Main flowchart of the proposal algorithm

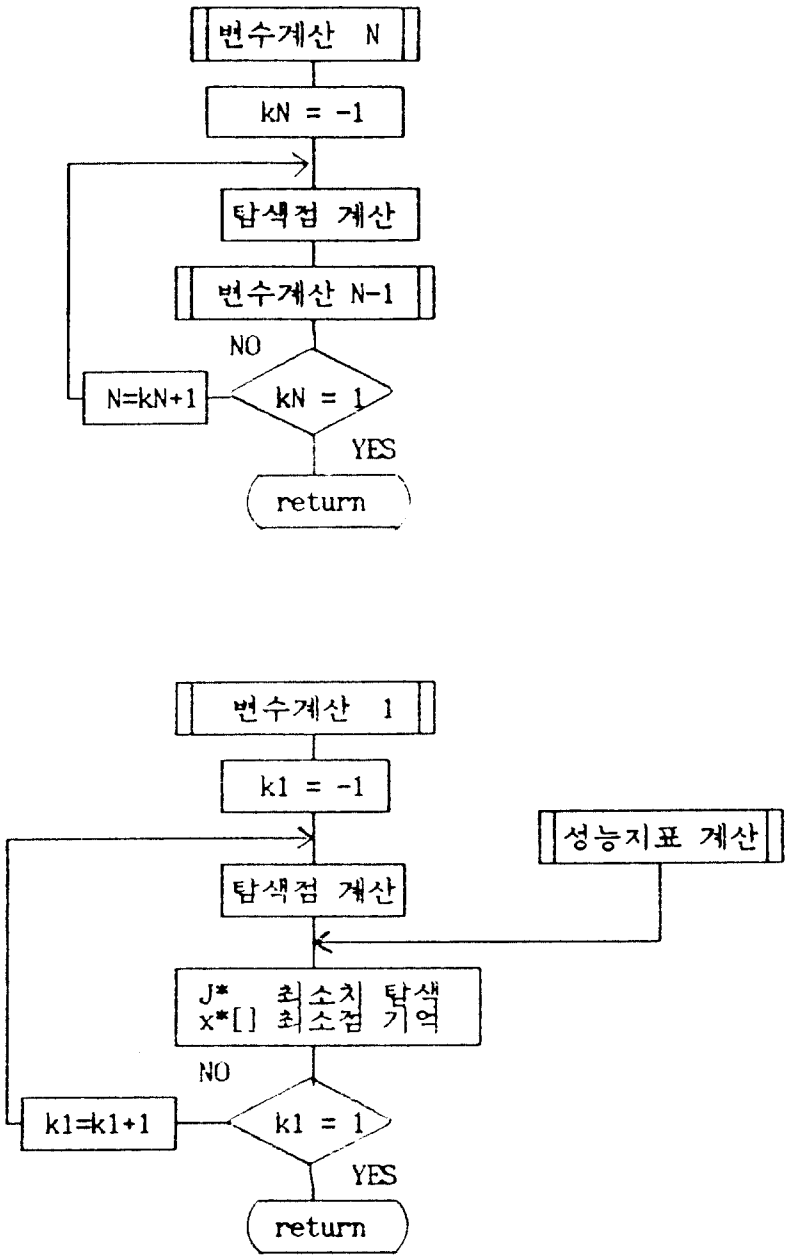

그림 5. 부프로그램의 흐름도
Fig 5. Flowchart of subprogram

Ⅲ.1 2분 탐색법과의 비교

단일 변수로된 성능지표의 최적치를 계산하기 위하여 탐색구간을 설정하고, 탐색구간의 중앙점에서 $\pm\delta$만큼 떨어진 두점에서 성능지표를 계산하여 계산값이 작은 부분으로 탐색방향을 결정하여 최소치(최적치)를 탐색하는 dichotomous 탐색법 및 탐색구간을 2분하는 golden section 탐색법을 이용하여 Rao의 저서에서 사용한 test 함수[3]

$$J=x\cdot(1.5-x) \tag{3.1}$$

을 적용하여 3^N점 탐색법으로 계산한 결과는 표1과 같다. 표 1은 −10에서 10까지를 탐색범위로 설정하여 처리한 결과를 비교한 것으로, dichotomous 탐색법에서는 3회 반복수행시에 탐색 방향이 잘못 결정되어 정확한 값이 출력되지 않았으며, golden section 탐색법에서는 탐색의 계산 값은 안정하게 감소하고 있으나, 17회의 탐색에서 정확한 값이 출력되었으며, 본 논문에서 제안한 3N 점 탐색 알고리즘 및 3^N탐색 알고리즘에서는 9회의 탐색에서 정확한 값을 출력하였을 뿐만 아니라, 계산값이 안정하게 감소하고

표 1. 다른 탐색법과 비교
Table 1. The comparasion with other search algorithms

탐색	제안 3점알고리즘		dichotomous탐색		golden section 탐색	
회수	계산값(J)	탐색점	계산값(J)	탐색점	계산값(J)	탐색점
1	0.0	0.0	0.5002	0.5005	9.1138	2.3606
2	−0.0	0.0	−0.5624	0.7502	2.0317	2.3606
3	−0.5504	0.64	−0.5470	0.8711	−0.5253	0.5572
4	−0.5504	0.64	−0.5586	0.8121	−0.5253	0.5572
5	−0.5624	0.7424	〃	〃	0.5253	0.5572
:	:	:			:	:
6	−0.5625	0.7500			0.5615	0.7198

* 참 고 : 참값은 $J^*=-0.5625$, $x^*=0.75$
golden section 탐색법은 17회의 실행가 참값으로 계산되어 출력되었음.

표 2. 3^N탐색 알고리즘에 의한 신뢰도
Table 2. Reliability of search rainge for 3^N search algorithm

탐색	$-2 < x_1 < 8$			$-3 < x_1 < 7$			$-5 < x_1 < 5$		
범위	$-3 < x_2 < 5$			$-7 < x_2 < 3$			$-5 < x_2 < 5$		
	J	x_1	x_2	J	x_1	x_2	J	x_1	x_2
1	−0.7778	−0.3333	1.0000	−0.8889	−1.3333	1.3333	0.0000	0.0000	0.0000
2	−1.0494	−1.4444	1.8889	−0.8889	−1.3333	1.3333	−0.9877	−1.1111	1.1111
3	−1.2442	−1.0741	1.5926	−1.2318	−0.9630	1.3333	−1.2209	−1.1111	1.4815
4	−1.2457	−0.9506	1.4938	−1.2486	−0.9630	1.4568	−1.2498	−0.9877	1.4815
5	−1.2500	−0.9918	1.4938	−1.2499	−1.0041	1.4979	−1.2498	−0.9877	1.4815
6	−1.2500	−1.0055	1.5048	−1.2499	−1.0041	1.4979	−1.2500	−1.0014	1.4952
7	−1.2500	−1.0009	1.5011	−1.2500	−0.9995	1.4979	−1.2500	−1.0014	1.4998
8	−1.2500	−0.9994	1.4999	−1.2500	−0.9995	1.4975	−1.2500	−0.9998	1.4998
9	−1.2500	−1.9999	1.4999	1.2500	−1.0001	1.5000	−1.2500	−0.9998	1.4998
10	−1.2500	−1.0001	1.5001	−1.2500	−1.0001	1.5000	−1.2500	−1.0000	1.4999

있음이 확인되었다.

그러므로, 성능지표의 계산횟수가 적어 수렴속도가 빠르고, 참 값과의 오차가 적어 신뢰도가 높으며, 탐색구간내에서만 탐색을 수행하므로 안정도가 높다.

Ⅲ.2 2차형식 성능지표에 대한 기존 탐색법과의 비교

Hooke와 Jeeves 및 Rosenbrock 탐색 알고리즘을 이용하여 Rao의 저서[3]에서 사용한 2차 형식의 성능지표

$$J(x_1,x_2)=x_1-x_2+2\cdot x_1^2+2\cdot x_1\cdot x_2+x_2^2 \quad (3.2)$$

를 test 함수로 적용한 계산 결과는 표 2와 같다.

표 2는 탐색범위를 달리한 여러가지 경우의 정확도를 시험한 것으로, unimodality의 범위를 벗어나지 않는 탐색범위에서는 참 값과의 오차가 ±5/10000의 이내에 있음을 알 수 있었다.

다변수로된 성능지표의 탐색법으로는 pattern 탐색법인 Hooke와 Jeeves의 탐색법과 pattern과 탐색 방향을 변경하는 Rosenbrock의 탐색법 등이 있다. 전자는 탐색의 시작점을 임의로 설정하고, 매 탐색시에 탐색의 방향을 결정하여야 하고, 스텝사이즈(step size)를 결정하기 위한 보조 계산식을 필요로 한다. 또한 후자 역시 탐색의 방향과 스텝사이즈의 계산을 위한 보조 계산식을 필요로 할 뿐만 아니라, 최적치가 탐색되는 방향으로 좌표축을 변경하게 되어 steep vally의 현상이 발생하게 된다.

표 3은 같은(식 3.2) 성능지표의 최적치를 Hooke와 Jeeves의 탐색법과 Rosenbrock 탐색법의 결과[3]를 비교하여 신뢰도와 정확도를 검증한 결과로 표에 나타난 바와 같이 제안한 3^N점 탐색법이 안정하게 감소되는 상태를 유지하고 있을 뿐만 아니라, 탐색을 위한 반복 탐색횟수가 많을수록 오차가 적고, 정확한 값을 출력하고 있음을 알 수 있다.

다음은 참고문헌[4,5,9]에서 발췌한 여러가지 형식과 이를 변형하여 구성한 성능지표등으로,

$$J1=-x1\times\cos(x1)^{(9)} \quad (3.3)$$
$$J2=(x1-1.5)^2+(x2-1.5)^{2\ (4)} \quad (3.4)$$
$$J3=3\cdot x1^2+2\cdot x1\cdot x2+2\cdot x2^2+7^{(4)} \quad (3.5)$$
$$J4=x1^2+2\cdot x2^2-4\cdot x1-2\cdot x1\cdot x2^{(5)} \quad (3.6)$$
$$J5=(x1-1)^2+(x2-2)^2+(x3-3)^2+(x4-4)^2 \quad (3.7)$$
$$J6=(x1-0.5)^2\times(x2+3)^2\times(x3+0.2)^2 \quad (3.8)$$
$$J7=(x1-x2^2)^2+3\cdot(x2+2)^2+(x3-3)^2+(x4+4)^2-10 \quad (3.9)$$

이에 대한 최소치를 제안한 알고리즘으로 탐색

표 3. 다른 탐색법과의 성능 비교
Table 3. The comparison of performance with other search method

반복 횟수	3^N점 탐색법 ($-5<x_1<5, -5<x_2<5$)			Hooke and Jeeves Search Method			Rosenbroke's Search Method		
	J	x_1	x_2	J	x_1	x_2	J	x_1	x_2
1	0.0	0.	0	0	0.	0	0	0	0
2	−0.9877	−1.11	1.11	−1.21	−0.8	1.3	−0.88	−0.4	0.8
3	−1.2209	−1.11	1.48	−1.235	−0.975	1.410	−1.156	−1.14	1.13
4	−1.2498	−0.98	1.48	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
11	−1.2500	−1.00	1.50	:	:	:	:	:	:

한 결과를 표 4에 제시하였다.

표 4의 검토 결과 J1(식 3.3)은 3N점 탐색법 알고리즘에 의한 결과이며, J2(식 3.4)부터 J8(식 3.9)의 결과는 3^N점 탐색 알고리즘의 결과이다. 이들 본 논문에서 제안한 알고리즘과 유사한 golden section 탐색 알고리즘의 결과와 비교 검토한 결과는 golden section 탐색법에서는 불안정한 결과가 출력되었으며, 특히 2변수 이상에서는 탐색이 불가능하였다.

그러므로 본 논문의 알고리즘이 단일 변수 및 다변수계에도 적용이 가능하며, 탐색되어진 값이 계속 감소하는 안정된 상태를 유지하고, 최종 탐색의 결과가 참고문헌에서 제시되어진 값과 정확하게 일치되거나, 또는 오차가

표 4. 여러가지 성능지표에 대한 최적치 결과
Table 4. The product of optimum value for various performance index

탐색 횟수	식 (3.3)		식 (3.4)			식 (3.5)		
	J1	x1	J2	x1	x2	J3	x1	x2
	(0 < x1 < 3.141593)		(-1 < x1 < 10, -1 < x2 < 10)			(-3 < x1 < 10, -2 < x2 < 15)		
1	-.000002	.5235	7.555556	4.1666	0.8333	9.083333	-0.8333	0.8333
2	-.560938	.8726	0.913580	2.2777	2.0555	9.058641	0.6111	-1.0555
3	-.560938	.8726	0.043895	1.6481	1.6481	7.186213	0.1296	0.2037
4	-.560938	.8726	0.003962	1.4382	1.5123	7.003315	-0.0308	-0.0061
5	-.560938	.8597	0.000220	1.5082	1.5123	7.001333	0.0226	-0.0061
6	-.561082	.8610	0.000075	1.5082	1.4972	7.000086	0.0048	-0.0061
7	-.561082	.8626	0.000005	1.5004	1.5022	7.000005	-0.0011	0.0016
8	-.561092	.8621	0.000000	1.5001	1.5006	7.000002	0.0008	-0.0009
9	-.561092	.8619	0.000000	1.5005	1.5000	7.000000	0.0001	-0.0001
10	-.561093	.8619	0.000000	1.4998	1.5000	7.000000	-0.0000	0.0001

계 속(continue)

탐색 횟수	식 (3.6)				식 (3.7)				
	J4	x1	x2	x3	J5	x1	x2	x3	x4
	(0 < x1 < 10, -2 < x2 < 12, 2 < x3 < 10)				(0 < x1 < 12, -2 < x2 < 15, 10 < x3 < 10, -15 < x4 < 5)				
1	-6.82330	1.6667	1.1667	4.0000	16.80556	2.0000	0.8333	0.0000	1.6667
2	-8.26896	6.1111	2.6111	2.6667	1.25000	0.6667	2.7222	2.2222	3.8889
3	9.92831	1.6296	2.1296	3.1111	0.03464	1.1111	2.0926	2.9629	3.8889
4	-9.94736	1.5062	2.1296	2.9629	0.02367	0.9629	2.0926	2.9629	3.8889
5	-9.98148	4.3415	2.0761	3.0123	0.00287	1.0123	2.0226	2.9629	3.9711
6	-9.98727	1.2866	2.0582	2.9958	0.00011	0.9959	1.9993	2.9993	3.9986
7	-9.98878	1.2684	2.0582	3.0013	0.00000	1.0013	1.9993	2.9995	3.9986
8	-9.98937	1.2623	2.0543	2.9995	0.00000	0.9995	1.9993	2.9995	3.9986
9	-9.98954	1.2602	2.0543	3.0001	0.00000	1.0001	2.0001	2.9995	3.9996
10	-9.98959	1.2595	2.0543	2.9999	0.00000	0.9998	1.9998	2.9998	3.9999

계속(continue)

탐색 횟수	식 (3.8)				식 (3.9)				
	J6	x1	x2	x3	J7	x1	x2	x3	x4
	$0 < x1 < 19$, $-7 < x2 < 22$, $0 < x3 < 20$				$-1 < x1 < 9$, $-7 < x2 < 5$, $-4 < x3 < 10$, $-5 < x4 < 7$				
1	234.65700	3.166	−4.833	3.333	−5.88888	0.667	−1.000	3.000	−3.000
2	0.13364	1.055	−2.277	−1.111	−9.44444	5.111	−2.333	3.000	−4.333
3	0.00002	0.351	−3.129	−0.370	−9.94680	3.629	1.888	3.000	−3.888
4	0.00001	0.586	−3.129	−0.123	−9.99383	4.123	−2.037	3.000	−4.037
5	0.00001	0.508	−3.031	−0.205	−9.99932	3.958	−1.987	3.000	−3.987
6	0.00000	0.508	−3.003	−2.205	−9.99992	4.013	−2.004	3.000	−4.004
7	0.00000	0.499	−3.003	−0.196	−9.99999	3.995	−1.998	3.000	−3.998
8	0.00000	0.499	−2.999	−0.199	−9.99999	4.001	−2.000	3.000	−4.000
9	0.00000	0.499	−2.999	−0.199	−9.99999	3.999	−1.999	3.000	−3.999
10	0.00000	0.499	−2.999	−0.200	−9.99999	4.000	−2.000	3.000	−4.000

±5 / 10000 이내에 있었다.

Ⅳ. 결 론

unimodality와 2차원 형식의 탐일변수와 다변수로 구성되는 성능지표의 최소치를 탐색하여 최적설계를 실현하기 위한 알고리즘으로 3점 탐색법을 제안하고, test함수를 설정하여 기존 알고리즘의 결과와 제안한 알고리즘의 결과를 비교 검토한 결과는 다음과 같다.

(1) 3^N점 탐색 알고리즘은 일반적인 2차형식 및 선형으로 표시되는 성능지표에 대하여 安全性과 正確度가 매우 좋은 탐색 성능을 갖고 있다.

(2) 3N점 탐색법은 서로 다른 변수가 乘除算項으로 결합되지 않는 성능지표에 대하여서는 信賴度가 양호하고, 수렴속도가 빠르며, 간단한 프로그램으로 신속한 탐색이 가능하다.

또한, 본 알고리즘은 프로그램이 간결하여 유연성이 좋으며, 탐색범위로 설정된 범위 내에서 성능지표의 값이 계속 감소되어 안정한 상태를 유지하며, 탐색의 방향 및 스텝사이즈의 계산을 위한 별도의 계산식을 필요로하지 않는다.

계속 연구 검토하여할 문제로는 비선형 특성으로된 성능지표의 최적치 탐색의 경우에는 불안정한 결과가 나타나는 경우와 전체영역(nonunimodal) 에서의 최적치 탐색을 위한 방법, 그리고 설계시방에 따라서 제한 조건이 있는 경우의 성능지표에 대한 최적치 탐색 알고리즘의 연구가 필요하다.

특히, 본 알고리즘으로 통신 시스템, 각종 기기 및 부품, 제어계등의 최적설계에서 가장 빈도가 높게 나타나는 일반적인 성능지표에 대한 최적치 탐색에 널리 이용할 수 있을 것으로 사료 된다.

參 考 文 獻

1. Krzysztof K. Burhardt, "An Adaptive Search Optimization Algorithm", IEEE Trans., Vol. c-23, No.9, pp 890-897, Sept 1974.
2. J.P.Lawrence Ⅲ, Kenneth Steiglitz, "Randomized Pattern Search", IEEE Trans. on Computers, pp 382-385, Apr 1972.
3. S.S.Rao, "Optimization theory and applications", Halsted Press, pp 215-338, 1984.
4. William H. Press외 3인, "Numerical Recipes", Cambridge(Pang Han), pp 166-176, pp 274-334, 1986.
5. Charles S. Beightler외 2인, "Foundations of Optimiz-

ation", pp 171~265, Prentice Hall(탑출판사), 2'nd ed., 1982.
6. Lawrence Hasdorff, "Gradient Optimization and Nonlinear Control", A Wileyinterscience Pub. 1976.
7. Donald.E. Kirk, "Optimal Control Theory", Prentice-Hall Inc., 1970.
8. E.Bryson, Yu-Chi Ho, "Applied Optimal Control", pp 212~245, A Halsted Press Book, 1975.
9. Byron S.Gottfried, "Introduction to Optimization Theory", pp 26~293, 탑출판사, 1973.
10. Andrew P.Sage, Chelswa C.White, "Optimum systems control", pp 287~382, Prentice Hall Inc., 1977.
11. 情報數學研究會篇, "コンピュータ基礎數學", pp 85~101, 日本理工出版社刊, 昭和 63.

金 周 弘(Ju Hong KIM) 正會員
1929年 1月 5日生
1952年：서울大學校 工科大學 卒業
現在：東國大學校 電子工學科 教授(工博)

孔 徽 植(Whue Sik KONG) 正會員
1954年 1月 26日生
1980年 2月：東國大學校 電子工學科 卒業(工學士)
1985年 2月：漢陽大學校 産業大學院 卒業(工學碩士)
1991年 2月：東國大學校 大學院 電子工學科 博士課程 修了
1985年 3月~現在：東亞專門大學 電子計算科 助教授