한국CDE학회 논문집
Korean Journal of Computational Design and Engineering

Vol. 21, No. 4, pp. 443-452. December 2016
DOI http://dx.doi.org/10.7315/CDE.2016.443

〈학술논문〉

pISSN 2508-4003
eISSN 2508-402X

고비용 블랙박스 함수의 RBF기반 근사 최적화 기법

박상근†
한국교통대학교 기계공학과

A Method for RBF-based Approximate Optimization of Expensive Black Box Functions

Sangkun Park†
Dept. of Mechanical Engineering, Korean Nat'l Univ. of Transportation

Received 18 May 2016; received in revised form 21 June 2016; accepted 21 June 2016

ABSTRACT

This paper proposes a method for expensive black box optimization using radial basis functions (RBFs). The proposed algorithm is a computational strategy that uses a RBF model approximating the expensive black box function to predict an optimum. First, a RBF-based approximation technique is introduced and a sampling plan for estimation of the black box function is described. Then the proposed algorithm is explained, which presents the pseudo-codes for implementation and the detailed description of each step performed in the optimization process. In addition, numerical experiments will be given to analyze the performance of the proposed algorithm, by investigating computation accuracy, number of function evaluations, and convergence history. Finally, geometric distance problem as application example will be also presented for showing the algorithm applicability to different engineering problems.

Key Words: Expensive black box optimization, Radial basis functions, Sampling plan

1. 서 론

공학 분야에서 다수 개의 설계 변수(혹은 파라미터)에 의해 정의되는 대규모, 대용량의 엔지니어링 설계 문제를 컴퓨터 시뮬레이션을 통하여 해결하려는 예를 흔히 찾아볼 수 있다. 유한요소 해석 기술을 사용한 구조해석 시뮬레이션, 유동해석 및 가시화를 위한 전산유체 모의실험 등이 그 대표적인 예가 될 것이다. 그런데 대부분의 경우에 최적의 설계 구성을 찾는데 상당한 노력과 비용이 소요되어 거의 불가능하다. 즉, 오늘날 컴퓨터 성능의 눈부신 발전에도 불구하고, 컴퓨터 시뮬레이션 수행에 필요한 막대한 계산 비용으로 말미암아, 다양한 설계 대안을 시도해 보려는 엔지니어링 설계자는 단지 몇몇의 시뮬레이션 결과에만 의존할 수 밖에 없다. 일반적으로 산업 현장에서 다루는 설계 최적화 문제는 목적함수 1회 계산을 위해 상당한 계산 비용을 요구하며 나아가 대부분의 경우에 최적화 수행에 필요한 구배 정보를 구할 수 없다. 이러한 이유로 수많은 엔지니어들은 시행착오 과정을 거쳐 현 설계 문제의 단순한 개선 결과에 만족한다.

이러한 고비용의 목적함수로 기술되는 최적화

†Corresponding Author, skpark@ut.ac.kr

문제를 해결하기 위해 지난 수 년 전부터 고비용의 목적함수 대신에 근사 모델을 사용하여 그 해를 구하려는 근사 최적화 기법이 연구되어 왔다. 여기서 근사 모델은 실제 목적함수를 대신한다는 의미로 보통 대안 모델(surrogate model)이라 부르며, 메타 모델(metamodel) 혹은 반응표면 모델(response surface model)이라 부르기도 한다. 한편 최적화하려는 목적함수에 관하여 그 어떠한 사전적 정보 혹은 경험없이 최적 해를 탐색하려는 시도가 함께 연구되어 왔는데, 이를 블랙박스(black box) 최적화라 부른다. 본 연구는 근사 모델로서 RBF(radial basis function)[1,2] 모델을 활용하여 이러한 고비용 블랙박스 함수의 최적 해를 근사적으로 구할 수 있는 최적화 알고리즘을 제시하고자 한다.

관련 연구를 살펴보면 다음과 같다. 대표적인 근사 모델 혹은 대안 모델로서 저차 다항식(low degree polynomial)[3], 크리깅(Kriging)[4], 방사함수(radial basis function)[5], 서포트 벡터 회귀(support vector regression)[6] 등이 있다. 또한 가감 혹은 스케일링을 통하여 모델 함수를 교정하는 다충실도(variable-fidelity or multi-fidelity)[7] 모델 방식이 있는데, 이는 데이터 점 근처에서 저충실 근사 모델의 함수값이 고충실 실제 모델과 같아지도록 교정하는 방식이다. 이러한 다충실도 모델 방식의 대표적인 예로 공간 매핑법(space mapping technique)[8], 다양체 매핑법(manifold mapping)[9] 기법이 있다. 공간 매핑법은 저충실 모델과 고충실 모델 사이의 입력 공간(input space) 매핑 함수의 교정을 통하여 최적 해 근처에서 모델 함수 값이 같아지도록, 다양체 매핑법은 저충실 모델의 출력 공간(output space)을 아핀 변환(affine transformation)하여 최적 해 근처에서 고충실 모델의 출력 공간과 접하도록, 매핑 관계를 개선해 나가는 방식이다. 이밖에 순차적 근사 최적화 과정 중 수치적으로 불량 상태(ill-conditioned)가 발생치 않도록 근사 모델의 수렴성을 관리해 나가는 모델 관리 기법(model management framework)이 있는데, 대표적인 방법으로 신뢰영역(trust-region) 모델 관리법[10], 패턴탐색(pattern-search) 관리법[11]이 있다. 이 두 관리 기법의 공통점은 모두 다충실 모델 방식을 사용하나, 차이점으로 신뢰영역의 경우 고충실 모델의 도함수를 필요로 하나 패턴탐색은 그렇지 않다.

한편, 본 연구에서 다루는 최적화 문제의 유형은 다음과 같이 3가지 형태로서 본 연구에서 제시하는 알고리즘에 의해 해결하고자 한다.

1) 비선형 연립 방정식: 함수 $f_i: \mathbf{R}^n \rightarrow \mathbf{R}$ ($i = 1, \ldots, n$)가 주어졌을 때 식 (1)을 만족하는 $\mathbf{x}$를 구한다.

$$f_i(\mathbf{x}) = 0, \quad 1 \le i \le n, \quad \mathbf{x} \in \mathbf{R}^n \tag{1}$$

2) 비선형 최소자승: 함수 $f_i: \mathbf{R}^n \rightarrow \mathbf{R}$ ($i = 1, \ldots, m$)가 주어졌을 때 식 (2)를 만족하는 $\mathbf{x}$를 구한다 (단, $m \ge n$).

$$\mathbf{min} \sum_{i=1}^{m} f_i^2(\mathbf{x}), \ \mathbf{x} \in \mathbf{R}^n \tag{2}$$

3) 비음수 함수의 최소화: 함수 $f: \mathbf{R}^n \rightarrow \mathbf{R}$ (≥ 0)가 주어졌을 때 식 (3)을 만족하는 $\mathbf{x}$를 구한다.

$$\mathbf{min}\ f(\mathbf{x}), \ \mathbf{x} \in \mathbf{R}^n, f(\mathbf{x}) \ge 0 \tag{3}$$

식 (3)에서 비음수 함수 $f(\mathbf{x})$를 제곱하면 그 결과는 식 (2)의 형태($m = 1$인 경우)를 가진다. 즉 식 (2)의 계산 알고리즘에 의해 식 (3)의 해를 구할 수 있음을 의미한다. 참고로 식 (3)에서 $f(\mathbf{x}) > -a$ ($a > 0$)라면, $f(\mathbf{x}) + a > 0$의 형태로 변환한 후 제곱하여 처리하면 된다. 한편, 식 (2)에서 $m = n$인 경우에 식 (2)는 식 (1)의 형태를 갖는다. 즉 식 (1)은 식 (2)의 특별한 경우이며, 이는 식 (2)의 계산 알고리즘에 의해 식 (1)의 해를 찾을 수 있음을 의미한다. 참고로, 식 (1)의 해가 존재하지 않는다면, 식 (2)의 계산 알고리즘은 식 (1)의 벡터 $\mathbf{f} = \{f_i(\mathbf{x}) | 1 \le i \le n\}$의 노름(norm)이 최소가 되는 해를 구한다. 결국, 식 (1)과 (3)은 식 (2)의 계산 알고리즘에 의해 그 해를 구할 수 있으며, 이때 흔히 사용되는 대표적인 알고리즘으로 LM(Levenberg-Marquardt)[12] 방법이 있다.

2. RBF기반 근사 최적화

본 연구는 앞에서 언급한 바와 같이, 고비용 목적함수의 경우에 식 (1)과 (3)의 문제를 식 (2)의 문제로 변환하며, 식 (2)의 해를 저렴하게 구하기 위해 근사 모델을 사용하여 식 (2)을 근사화한다. 이때 본 연구 알고리즘은 비선형성 및 불량상태 조건의 문제들을 효과적으로 해결하기 위해 기존 LM알고리즘을 수정하며, 더불어 사용자 설계 목표를 최적화 문제에 반영시키기 위해 목적함수 형태를 변경한다. 결국, 본 연구에서 다루는 실제 최

적화 문제는 식 (4)와 같으며, 이 문제의 근사 해를 구하기 위한 근사 모델 기반의 최적화 문제는 식 (5)와 같이 작성한다.

○ 실제 최적화 문제

$$\textbf{min } F(\mathbf{x}) = \|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|^2 = \sum_{i=1}^{m} (f_i(\mathbf{x}) - y_i)^2 \qquad (4)$$

여기서 $\mathbf{y} \in \mathbf{R}^m$는 사용자 설계 목표를 표시하며, 함수 $F(\mathbf{x})$는 원래의 실제 모델 $\mathbf{f}(\mathbf{x})$와 설계 목표 $\mathbf{y}$ 사이의 불일치를 의미하는 목적함수이다. 여기서 고충실 모델인 $\mathbf{f}(\mathbf{x})$는 고비용 블랙박스 함수로서 최적화에 필요한 자코비안(Jacobian) 행렬 $\mathbf{J_f(x)} = \mathbf{df/dx}$를 구할 수 없다라고 가정한다.

○ 근사 최적화 문제

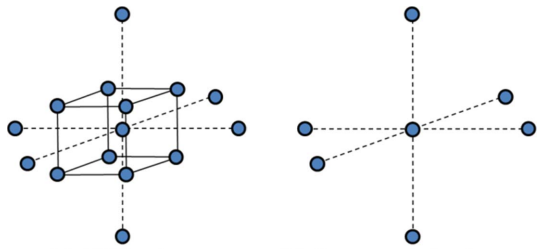
$$\textbf{min } S(\mathbf{x}) = \|\mathbf{s}(\mathbf{x}) - \mathbf{y}\|^2 = \sum_{i=1}^{m} (s_i(\mathbf{x}) - y_i)^2 \qquad (5)$$

여기서 저충실 모델인 $\mathbf{s}(\mathbf{x})$는 실제 모델 $\mathbf{f}(\mathbf{x})$의 근사 모델로서 저렴하게 계산되며, 자코비안 행렬 $\mathbf{J_s(x)} = \mathbf{ds/dx}$은 저렴하게 계산할 수 있다라고 가정한다.

일반적으로 식 (2), (4), 또는 (5)와 같은 비선형 최소자승 문제의 해를 구하기 위해선 먼저 자코비안 행렬의 계산이 필요하다. 이 행렬 계산은 앞에서 가정한 바와 같이 컴퓨터 시뮬레이션 기반 엔지니어링 최적화 문제의 경우에 대부분 불가능하다. 본 연구에서는 이상과 같이 목적 함수 계산에 큰 비용이 요구되며, 목적 함수 1차 미분에 의해 정의되는 자코비안 행렬 계산이 불가능한 고비용 블랙박스 함수의 최소화 문제에 관하여, 근사 최적 해를 저렴하게 구할 수 있는 RBF기반의 근사 최적화 알고리즘을 제시하고자 한다. 먼저 블랙박스 함수의 전체 형상을 어림잡기 위해 블랙박스 함수가 존재하는 설계 공간 상에서 임의의 점을 샘플링 하고(2.1절), 이 샘플링 점과 그 점에서 계산된 함수 값을 가지고 $\mathbf{s}(\mathbf{x})$에 해당하는 RBF모델을 생성한다(2.2절). 마지막으로 생성된 RBF모델을 반복적으로 개선하여 최적 해에 접근한다(2.3절).

2.1 샘플링 방법

샘플링에 의해 특정 모델 혹은 시스템을 표현 모델에 의해 효과적으로 근사하기 위해 샘플 점의 개수와 그 분포가 매우 중요하다. 실험 계획법 또는 컴퓨터 모의 실험까지 포함하는 샘플링 계획(sampling plan)이 바로 이러한 샘플링 문제의 해법을 제시하기 위해 연구되어 왔다. 자주 언급되는 샘플링 방법 가운데 대표적 방법으로 CCD(central composite design)가 있다. 이 방법은 설계 변수의 개수가 큰 경우, 거대한 양의 샘플 점을 요구하는 문제를 가지고 있다. 일반적으로 설계 공간 상에 샘플 점은 균등하게 분포되어야 하며, 각 설계 방향으로의 투상 점 분포 또한 균등하고 같은 값에서 중첩되어서는 안 된다. 이를 위한 대표적인 방법으로 OA(orthogonal arrays)[13]와 LHS(Latin Hypercube sampling)[14] 기법이 있다.



(a) CCD scheme (b) the proposed scheme

Fig. 1 Sampling plan

본 연구에서는 고비용 함수의 저렴한 최적화를 위하여, 즉 고비용 함수의 계산 회수를 최대한 줄이기 위하여, 다른 요소들(예: 데이터 균일성, 직교성, 회전성)보다 샘플 점의 개수를 가장 중요한 요소로 판단하고 이를 최대한 줄이기 위한 방안으로 Fig. 1과 같은 샘플링 방법을 제안한다. 이 방법은 $(2^n + 2n + 1)$개의 CCD 설계 점에서 큐브(factorial cube)의 꼭지점 상에 존재하는 2^n개의 점을 제거한 CCD의 축소형으로 $(2n + 1)$개의 설계 점을 가진다. 본 제시 방법은 비록 데이터 회전성은 보장하지 못하나, 함수 계산 회수의 감소를 통해 효율적인 근사 최적화를 기대할 수 있다.

2.2 RBF기반 근사 기법

본 연구에서 사용한 근사 모델은 RBF함수이다. RBF는 다항식 혹은 스플라인 보간 기법에 의해 다루기 힘든 다차원 공간 상의 비구조화 샘플 데이터를 데이터 특성에 무관하게 보간(근사)하는 표현 모델로서, 그 수학적 정의가 간단하고, 구현하기 쉬우며, 근사 품질 또한 우수하기 때문에 다양한 분야에서 널리 사용된다. 벡터 함수 $\mathbf{f}(\mathbf{x}) \in \mathbf{R}^m$를 보간하는 RBF모델 $\mathbf{s}(\mathbf{x})$를 살펴보면 식 (6)과

같다.

$$\mathbf{s}(\mathbf{x}) = \sum_{i=1}^{N} \boldsymbol{\beta}^i \phi(\| \mathbf{x} - \mathbf{x}^i \|),\ \mathbf{x} \in \mathbf{R}^n,\ \boldsymbol{\beta}^i \in \mathbf{R}^m \quad (6)$$

여기서 $\|\mathbf{x} - \mathbf{x}^i\|$는 임의의 점($\mathbf{x}$)과 샘플 점($\mathbf{x}^i$) 사이의 유클리드 거리를 나타내고, $\phi(\cdot) : \mathbf{R}+ \rightarrow \mathbf{R}$는 방사형의 기저함수(basis function)로서 대표적인 예로 thin-plate spline, multiquadric, Gaussian RBFs 등이 있고, $\boldsymbol{\beta}^i$는 RBF모델의 계수 벡터이다.

N개의 샘플 점 $\mathbf{x}^i \in \mathbf{R}^n$와 이들의 함수 벡터 값 $\mathbf{f}^i = \mathbf{f}(\mathbf{x}^i) \in \mathbf{R}^m$가 주어졌을 때, 이를 보간하는 RBF 모델의 생성은 식 (7)로부터 구해진다.

$$\mathbf{s}(\mathbf{x}^j) = \sum_{i=1}^{N} \boldsymbol{\beta}^i \phi(\| \mathbf{x}^j - \mathbf{x}^i \|) = \mathbf{f}(\mathbf{x}^j) \quad (7)$$

여기서 $j = 1, \ldots, N$이며, 이 식은 미지수 $\boldsymbol{\beta}^i$ ($i = 1, \ldots, N$)를 구하는 식 (8)의 행렬식 형태로 작성할 수 있다.

$$[\Phi] \begin{pmatrix} \boldsymbol{\beta}^1 \\ \vdots \\ \boldsymbol{\beta}^N \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{x}^1) \\ \vdots \\ \mathbf{f}(\mathbf{x}^N) \end{pmatrix} \quad (8)$$

여기서 $[\Phi] \in \mathbf{R}^{N \times N}$는 식 (9)에 의해 정의된다.

$$[\Phi] = \begin{pmatrix} \phi(\| \mathbf{x}^1 - \mathbf{x}^1 \|) & \cdots & \phi(\| \mathbf{x}^1 - \mathbf{x}^N \|) \\ \vdots & \ddots & \vdots \\ \phi(\| \mathbf{x}^N - \mathbf{x}^1 \|) & \cdots & \phi(\| \mathbf{x}^N - \mathbf{x}^N \|) \end{pmatrix} \quad (9)$$

일반적으로 샘플 점의 개수가 많을수록 식 (9)의 계산 비용이 크게 증가하며, 또한 식 (9)의 수치적 불량 상태는 샘플 점의 분포 상태에 따라 그 크기가 달라지나 대부분의 경우 매우 심하다. 본 연구의 경우, 2.1절에 제시된 샘플링 방법에 의하여 샘플 점의 개수를 가능한 최소화하며, 불량 상태 처리를 위해 의사 역행렬(pseudo-inverse) 기법을 사용한다.

한편, RBF 기저함수 중에 본 연구에서는 가우시안(Gaussian) RBF을 선택하였다. 이 RBF을 선택한 이유는 샘플 점의 분포가 균등하지 않더라도 중첩되지 않는다면 식 (9)의 $[\Phi]$가 정칙 행렬(nonsingular matrix)임을 보장하기 때문이다. 또한 다른 기저함수에 비해 도함수 유도 및 계산이 간결하여 본 연구 최적화 수행에 필요한 목적함수의 구배(gradient) 및 헤시안(Hessian) 계산이 용이하다. 그러나 2.3절의 본 연구 알고리즘은 가우시안이 아닌 다른 RBF함수의 사용을 허용하며, 나아가 RBF모델이 아닌 다른 형태의 근사 모델 사용도 가능하다.

2.3 근사 최적화 알고리즘

본 연구에서 제시하는 근사 최적화 알고리즘은 고비용 블랙박스 함수를 근사하는 가우시안 RBF 모델을 사용하여 최적 해 근처에서 점차적으로 개선해 나가는 반복순환 구조를 가지고 있다. 먼저 본 연구에서 제시한 샘플링 방법에 의해 생성된 샘플 점 위치에서 고비용 함수를 계산하고, 이 데이터를 기반으로 가우시안 RBF모델을 생성한다. 그 다음은 RBF모델에 의해 정의된 목적함수에 대하여 최적화 기법을 적용하여 근사 해를 구하고, 이를 샘플링 데이터에 추가하여 RBF 근사 모델을 개선한다. 이러한 근사 모델의 최적화 및 개선 과정은 종료조건이 만족될 때까지 반복하여 수행된다.

Algorithm: A RBF-based Approximate Optimization.

Inputs:
(1) The maximum number of function evaluations: $k_{\max}$.
(2) Two tolerances used for stopping criteria: ε_x and ε_F.
(3) The feasible domain: $\mathbf{x}^l$ and $\mathbf{x}^u$.
(4) The design aims (or specification): $\mathbf{y}$

Output: The best solution $\mathbf{x}^*$ obtained by the algorithm.

// Build an initial RBF model.
Step 1. Generate a sampling data set $\mathbf{X}^0$ in $[\mathbf{x}^l, \mathbf{x}^u]$.
Step 2. Build a RBF model $\mathbf{s}^0(\mathbf{x})$ with $\mathbf{X}^0$.
Step 3. Find a start point to used when $k = 0$ at step 4.

// Optimizing and improving the RBF model.
For $k = 0, \ldots, k_{\max}$, do the following steps (4~6):
Step 4. Minimize $\|\mathbf{s}^k(\mathbf{x}) - \mathbf{y}\|^2$ to find the next iterate $\mathbf{x}^{k+1}$.
Step 5. Test for convergence with ε_x and ε_F.
Step 6. Improve $\mathbf{s}^k(\mathbf{x})$ with $\mathbf{X}^{k+1} = \mathbf{X}^k \cup \{(\mathbf{x}^{k+1}, \mathbf{f}(\mathbf{x}^{k+1}))\}$.

Fig. 2 The proposed algorithm for RBF-based optimization

Algorithm: The Proposed Sampling Scheme.

Input: The feasible domain, $\mathbf{x}^l$ and $\mathbf{x}^u$.
Output: The sampled point set $\{ \mathbf{x}^i \mid i = 0, \ldots, 2n \}$
Set $\mathbf{x}^0 = (\mathbf{x}^l + \mathbf{x}^u) / 2$
Set $\alpha = 0.6180339887\ldots$ (or golden ratio conjugate)
For $i = 1, \ldots, 2n$, we set $\mathbf{x}^i = \mathbf{x}^0$
For $j = 1, \ldots, n$, we compute (1) and (2),

(1) $\mathbf{x}^i_j = \mathbf{x}^0_j - \alpha(\mathbf{x}^u_j - \mathbf{x}^l_j)/2$ where $i = 2j - 1$

(2) $\mathbf{x}^i_j = \mathbf{x}^0_j + \alpha(\mathbf{x}^u_j - \mathbf{x}^l_j)/2$ where $i = 2j$

Fig. 3 The proposed sampling scheme

이러한 본 연구 알고리즘의 구현을 위한 가상 코드는 Fig. 2와 같다. 총 6개의 Step으로 구성되며 이 중 에서 Step 4~6은 반복 형태로 진행된다. 각 과정 별로 상세히 살펴보면 다음과 같다.

○ **Step** 1: 샘플 데이터 $\mathbf{X}^0 = \{(\mathbf{x}^i, \mathbf{f}(\mathbf{x}^i)) | i = 0, \ldots, 2n\}$를 생성한다. 여기서 n은 설계변수의 개수이고, $\mathbf{x}^i$는 Fig. 3의 샘플링 알고리즘에 의해 결정되며, $\mathbf{f}(\mathbf{x})$는 고비용 블랙박스 함수를 의미한다.

○ **Step** 2: 앞 과정에서 구한 샘플 데이터 $\mathbf{X}^0$를 가지고 2.2절에서 소개한 RBF 근사 기법을 사용하여 가우시안 RBF모델을 생성한다. 여기서 근사 정밀도를 높이기 위해 가우시안 RBF함수 $\phi(r) = \exp(-cr^2)$의 형상 파라미터 c를 교차 검증(cross validation)을 통해 최적의 값을 구할 수 있으나, 본 연구에서는 $c = 0.017/r_{\max}^2$[15]를 사용한다. $r_{\max}$는 샘플 영역에서 이웃 점 간의 최대 거리를 가리킨다.

○ **Step** 3: Step 4에서 $k = 0$일 때 수행되는 첫 번째 최적화의 초기치 $\mathbf{x}^*$를 구한다. 이 초기치는 Step 1에서 구한 샘플 점 $\{\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^{2n}\}$ 중에 목적함수 값이 최소인 점으로 결정한다. 즉 다음의 식에 의해 초기치 $\mathbf{x}^*$를 결정한다. 그리고 다음 과정을 위해 $\mathbf{x}^*$를 $\mathbf{x}^0$로 설정한다.

$$\text{Find } \mathbf{x}^* = \text{argmin } \{\|\mathbf{f}(\mathbf{x}^i) - \mathbf{y}\|^2\}$$

where $\mathbf{x}^i \in \{\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^{2n}\}$.

○ **Step** 4: 기존 LM알고리즘의 약점을 보안한 LM알고리즘을 사용하여 아래 최적화 문제의 해를 구한다.

$$\text{Find } \mathbf{x}^{k+1} = \text{argmin } \|\mathbf{s}^k(\mathbf{x}) - \mathbf{y}\|^2 \text{ with a start point } \mathbf{x}^k.$$

적용된 LM알고리즘은 초기치가 해로부터 원거리에 있을 때 수렴 속도를 향상시키기 위한 것으로, 원거리에서 기존 LM알고리즘은 거의 급강하(steepest descent) 법에 의해 해를 탐색하는데 그 진행 크기가 너무 작아 낮은 수렴 속도를 보인다. 이를 개선하기 위해 헤시안을 사용하는 뉴튼 혹은 준뉴튼(quasi-Newton) 법을 사용하여 해의 탐색 속도를 높인다. 헤시안 행렬이 양의 정부호이면 뉴튼 법을 사용하고, 그렇지 않으면 준뉴튼 법을 사용한다. 여기서 준뉴튼 법의 헤시안은 BFGS update 방법을 사용하여 근사적으로 구한다.

위의 LM 최적화 수행을 위해 $S(\mathbf{x}) = \|\mathbf{s}(\mathbf{x}) - \mathbf{y}\|^2$의 구배 벡터와 헤시안 행렬이 필요한데, 근사 모델 $\mathbf{s}(\mathbf{x})$ 및 그 도함수를 사용하여 식 (10) 및 (11)과 같이 유도되므로 이를 사용하여 손쉽게 계산한다.

1) Gradient vector of $S(\mathbf{x})$:

$$\frac{\partial S}{\partial x_i} = \sum_{k=1}^{m} 2(s_k - y_k)\frac{\partial s_k}{\partial x_i} \quad i = 1, \ldots, n \tag{10}$$

2) Hessian matrix of $S(\mathbf{x})$:

$$\frac{\partial^2 S}{\partial x_i \partial x_j} = \sum_{k=1}^{m} 2\left(\frac{\partial s_k}{\partial x_i}\frac{\partial s_k}{\partial x_j} + (s_k - y_k)\frac{\partial^2 s_k}{\partial x_i \partial x_j}\right)$$

$$i = 1, \ldots, n \text{ and } j = 1, \ldots, n \tag{11}$$

○ **Step** 5: 본 연구 알고리즘은 다음 세 가지 종료조건 중에 하나가 만족되면 종료된다.

1) Stepsize-based criterion: $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \le \varepsilon_x$
2) Function-based criterion:
   $\|F^{k+1} - F^k\| / \max\{\|F^k\|, \|F^{k+1}\|, 1\} \le \varepsilon_F$
   where $F^k = F(\mathbf{x}^k) = \|\mathbf{f}(\mathbf{x}^k) - \mathbf{y}\|^2$
3) Iteration-based criterion: $k > k_{\max}$

이밖에 구배 조건을 종료조건으로 사용할 수 있으나, 본 연구의 경우는 어떠한 미분도 필요로 하지 않는 비도함수 방법(derivative-free method)이기 때문에 구배 조건을 설정할 수 없다. 일반적으로 비도함수 최적화 방법의 경우에 위에서 기술한 종료조건을 사용한다. 한편 종료조건에서 사용된 ε_x와 ε_F는 사용자가 미리 설정하는 값으로, 추후 소개할 예제에서 $\varepsilon_x = \varepsilon_F = 1.0\text{e-}6$로 설정하였다. 그리고 무한 반복을 막기 위해 $k_{\max} = 20$을 사용하였다.참고로 반복회수가 20회를 넘길 경우 그동안 구한 계산 결과가 반환된다.

○ **Step** 6: Step 4에서 구한 $\mathbf{x}^{k+1}$와 해당하는 함수 값 $\mathbf{f}(\mathbf{x}^{k+1})$을 아래와 같이 기존 샘플 데이터 $\mathbf{X}^k$에 첨가하여 새로운 샘플 데이터 $\mathbf{X}^{k+1}$을 구성한다.

$$\mathbf{X}^{k+1} = \mathbf{X}^k \cup \{(\mathbf{x}^{k+1}, \mathbf{f}(\mathbf{x}^{k+1}))\}$$

그리고 이 샘플 데이터 $\mathbf{X}^{k+1}$를 기반으로 2.2절의 RBF기반 근사법을 사용하여 RBF모델 $\mathbf{s}^{k+1}(\mathbf{x})$을 생성한다. 여기서 RBF모델을 재건하는데 드는 계산 비용은 고비용 함수 $\mathbf{f}(\mathbf{x})$을 계산하는데 요구되는 것보다 훨씬 작다. 이렇게 생성된 RBF모델 $\mathbf{s}^{k+1}(\mathbf{x})$은 이전 RBF모델 $\mathbf{s}^k(\mathbf{x})$보다 해 근방에서 $\mathbf{f}(\mathbf{x})$에 가

깝기 때문에 개선되었다고 보며, 다음 반복 과정 (k+1)에서 보다 정밀한 해($\mathbf{x}^{k+2}$)를 기대할 수 있게 된다. 이것은 고비용 함수 **f**(**x**)의 계산 회수 1회 증가를 통해, **s**(**x**)와 **f**(**x**) 사이의 불일치가 감소됨을 의미하고, 식 (5)의 근사 최적화 문제가 식 (4)의 고비용 실제 문제로 접근함을 의미한다.

3. 수치결과 및 성능평가

본 연구에서 제시한 근사 최적화 알고리즘의 계산 성능을 평가하기 위해 6개의 수학 함수를 사용하였고, 다 분야로의 응용 가능성을 확인하기 위해 CAD/CAM 등의 분야에서 흔히 요구하는 3차원 점과 자유 곡선 간의 최소 거리 문제를 다루었다.

3.1 테스트 함수

2.3절의 최적화 알고리즘의 계산 성능을 평가하기 위해 최적화 분야에서 널리 알려진 아래의 6개 수학 함수[16]를 테스트하였다. 각 테스트 함수는 다음의 형식으로 기술된다.

○ 함수이름 [라벨]
(a) 차원(dimension)
(b) 함수 정의(function definition)
(c) 최소값(minima)

○ ***Ackley function*** **[ACK]**
(a) n variable, m = 1
(b) $f(\mathbf{x}) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$
(c) $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, \ldots, 0)$

○ ***Trid function*** **[TRI]**
(a) n variable, m = 1
(b) $f(\mathbf{x}) = \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
(c) $f(\mathbf{x}^*) = -n(n+4)(n-1)/6$ at $x_i^* = i\ (n - i + 1)$
e.g., $f(\mathbf{x}^*) = -50$ at $\mathbf{x}^* = (6, 10, 12, 12, 10, 6)$ for n = 6

○ ***Bohachevsky function*** **[BOH]**
(a) n = 2, m = 3
(b) $f_1(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

$f_2(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$

$f_3(\mathbf{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$

(c) $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (0, 0)$

○ ***Box three-dimensional function*** **[BOX]**
(a) n = 3, $m \geq n$ variable
(b) $f_i(\mathbf{x}) = \exp[-t_i x_1] - \exp[-t_i x_2] - x_3\left(\exp[-t_i] - \exp[-10\ t_i]\right)$
where $t_i = (0.1)i$
(c) $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (1, 10, 1)$, $(10, 1, -1)$ and whenever ($x_1 = x_2$ and $x_3 = 0$)

○ ***Helical valley function*** **[HEL]**
(a) n = 3, m = 3
(b) $f_1(\mathbf{x}) = 10[x_3 - 10\theta(x_1, x_2)]$

$f_2(\mathbf{x}) = 10[(x_1^2 + x_2^2)^{1/2} - 1]$

$f_3(\mathbf{x}) = x_3$

where

$$\theta(x_1, x_2) = \frac{1}{2\pi}\arctan\left(\frac{x_2}{x_1}\right) \ \textit{if}\ x_1 > 0$$
$$= \frac{1}{2\pi}\arctan\left(\frac{x_2}{x_1}\right) + 0.5 \ \textit{if}\ x_1 < 0$$

(c) $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (1, 0, 0)$

○ ***Extended Rosenbrock function*** **[ERO]**
(a) n variable but even, $m = n$
(b) $f_{2i-1}(\mathbf{x}) = 10\,(x_{2i} - x_{i-1}^2)$

$f_{2i}(\mathbf{x}) = 1 - x_{2i-1}$

(c) $f(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (1, \ldots, 1)$

여기서 n은 설계 변수의 개수이고, m은 벡터 함수 $\mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$의 벡터 크기이다. 그리고 $m \neq 1$일 때 목적함수 F(**x**)는 식 (12)와 같이 정의한다.

$$\mathrm{F}(\mathbf{x}) = f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) + \cdots + f_m^2(\mathbf{x}) \qquad (12)$$

한편, 이 테스트 함수들은 다음과 같이 3가지 문제 유형 중 하나에 속한다. 즉

○ ACK, TRI : 비음수 함수 최소화 문제 (식 (3))
○ BOH, BOX : 비선형 최소자승 문제 (식 (2))
○ HEL, ERO : 비선형 연립 방정식 문제 (식 (1))

그러나 이미 앞에서 기술한 바와 같이 모두 비선형 최소자승 문제로 변환하여 그 해를 구한다.

Table 1 Performance results of mathematical test problems. (The 'k' denotes the final iteration or func. evaluation number.)

Label	n	m	$\|x^k - x^*\|$	$\|\mathbf{F}(x^k) - \mathbf{F}(x^*)\|$	k	N_f
ACK	3	1	0.000090	1.0e-12	5	12
TRI	6	1	0.000669	1.0e-12	6	19
BOH	2	3	0.001468	1.0e-12	2	7
BOX	3	10	0.000006	1.0e-12	3	10
HEL	3	3	0.000025	1.0e-12	6	13
ERO	20	20	0.000003	1.0e-12	3	44

3.2 알고리즘 성능 분석

3.1절의 각 테스트 함수에 대해, 본 연구 알고리즘의 수행 결과는 Table 1과 같다. 이 결과를 바탕으로 본 알고리즘의 계산 성능을 평가를 위해 아래의 평가지표를 사용하였다.

1) 계산 정확성(computation accuracy)
2) 함수 계산 회수(number of function evaluations)
3) 알고리즘 수렴성(algorithm convergence)
4) 근사모델과 실제함수 간의 유사성(similarity)

1) 계산 정확성: 아래의 두 가지 평가지표를 사용하여 계산 정확성을 측정한다.

○ 해의 정확성
Solution accuracy = $\|\mathbf{x}^k - \mathbf{x}^*\|$

○ 목적함수 정확성
Objective accuracy = $\|\mathbf{F}(\mathbf{x}^k) - \mathbf{F}(\mathbf{x}^*)\|$

여기서 $\mathbf{x}^*$는 참값에 해당하는 최적 해를, $\mathbf{x}^k$는 본 알고리즘에서 계산한 근사 해를 가리키며, $\mathbf{F}(\mathbf{x})$는 고비용 목적함수를 나타낸다. Table 1에서 보는 바와 같이, 테스트 문제의 평균 해 정확성은 대략 0.000377이며 목적함수 정확성은 거의 0.0이다. 이 결과는 비록 테스트 문제의 경우이지만 본 알고리즘이 매우 정확한 근사 해를 제공할 수 있음을 보여주는 것이다.

2) 함수 계산 회수: 최적화 과정이 종료할 때까지 소요된 계산 비용을 컴퓨터 성능과 무관하게 나타내기 위한 평가지표로서, 최적화 분야에서 특히 고비용 함수의 최적화 문제에서 알고리즘의 품질을 평가하는 주요 지표로 사용된다. 본 연구의 함수 계산 회수 N_f는 식 (13)과 같이 계산된다.

$$N_f = (2n+1) + k \tag{13}$$

여기서 $(2n+1)$은 Fig. 2의 Step 1~3에서 초기 RBF 근사모델을 생성하기 위해 사용된 샘플 점의 개수이고, k는 본 연구 종료 시의 반복 회수로서 Step 4~6에서 1회 반복 수행 때마다 첨가되는 샘플 점의 개수를 가리킨다. Table 1에서 반복 수행에 의한 함수 계산 회수 k가 매우 작음을 확인할 수 있고, N_f 는 n의 증가에 의해 증가하나 m의 변화에 무관함을 확인할 수 있다. 더불어 $k < (2n+1)$이므로 샘플링 방법의 개선을 통해 본 연구 알고리즘의 성능을 높일 수 있으리라 판단된다.

3) 알고리즘 수렴성: Fig. 4와 같이 반복회수 k가 증가함에 따라, k번째 근사 해인 $\mathbf{x}^k$와 참값에 해당하는 $\mathbf{x}^*$ 사이의 거리가 어떻게 변화하는지를 살펴봄으로써 알고리즘 수렴성을 확인할 수 있으며, 목적 함수 $\mathbf{F}(\mathbf{x}^k)$와 $\mathbf{F}(\mathbf{x}^*)$ 사이의 거리 변화를 보여주는 Fig. 5을 통해서도 수렴성을 확인할 수 있다. 한

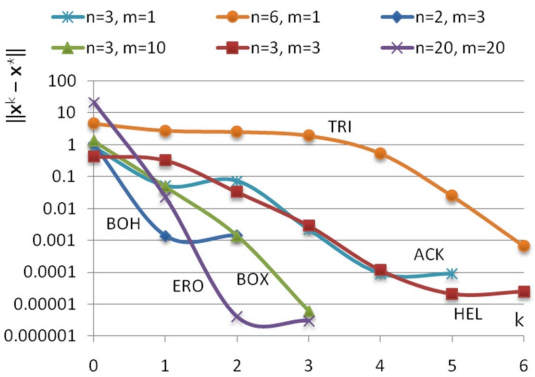


Fig. 4 Convergence history of $\|\mathbf{x}^k - \mathbf{x}^*\|$where $\mathbf{x}^k$ denotes the k-th iterate and $\mathbf{x}^*$ the true solution

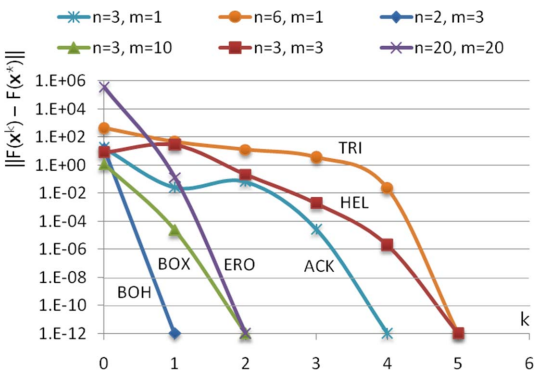


Fig. 5 Convergence history of $\|\mathbf{F}(\mathbf{x}^k) - \mathbf{F}(\mathbf{x}^*)\|$where $\mathbf{F}(\mathbf{x})$ denotes the objective function

편 Fig. 4와 5의 수치 결과를 바탕으로 해의 수렴 속도를 살펴보면,

○ ACK의 경우: 선형(linear) 수렴
○ HEL의 경우: 초선형(superlinear) 수렴
○ TRI, BOX, ERO의 경우: 이차(quadratic) 수렴

임을 확인할 수 있다. 여기서 BOH의 경우, 반복회수가 너무 적어 판단할 수 없다. 그리고 목적 함수의 수렴 속도는 BOH 경우를 제외하고 모두 이차 수렴임을 확인할 수 있다. 이상의 수치 예제를 통하여 본 연구 알고리즘은 비록 이론적으로 증명되지 않았지만 이차 수렴에 준하는 수렴 성질을 갖고 있다고 판단된다.

4) 근사모델과 실제함수 간의 유사성: 고비용 함수 **f**(**x**)와 근사 모델 **s**(**x**) 간의 유사성는 최적 해 **x***로부터 원거리에 있는 곳보다 근거리에서 더욱 중요하다. 이를 확인하기 위해, 최적 해 **x*** 근방에서 **s**(**x**)와 **f**(**x**) 사이의 유사성을 식 (14)와 (15)에 의해 측정하여 본 알고리즘의 품질을 평가한다.

$$\mathrm{RMSE}_k = \sqrt{\frac{\sum_{i=1}^{M} \| \mathbf{f}(\mathbf{x}^i) - \mathbf{s}^k(\mathbf{x}^i) \|^2}{M}} \qquad (14)$$

$$\mathrm{MAXE}_k = max(\|\mathbf{f}(\mathbf{x}^i) - \mathbf{s}^k(\mathbf{x}^i)\|) \qquad (15)$$

여기서 RMSE_k와 MAXE_k는 **x*** 근방에서 M개의 무작위 샘플 점 $\{\mathbf{x}^i \mid i = 1, \ldots, M\}$을 가지고 측정하는데, RMSE_k는 평균오차를 나타내고 MAXE_k는 최대오차를 의미한다. 그리고 아래첨자 k는 반복회수를 가리킨다. 본 연구에서 M은 $M = min(1000, 5^n)$으로 설정하였고, 무작위 샘플 점은 식 (16)에 의해 정의되는 영역 [$\mathbf{b}^u$, $\mathbf{b}^l$]에서 추출하였다.

$$\mathbf{b}^u = \mathbf{x}^* + \alpha(\mathbf{x}^u - \mathbf{x}^l)$$
$$\mathbf{b}^l = \mathbf{x}^* - \alpha(\mathbf{x}^u - \mathbf{x}^l) \qquad (16)$$

여기서 $\alpha = 0.01$로 설정하였고, 상수 1000, 5, 0.01은 RMSE_k와 MAXE_k 측정 결과에 큰 영향을 주지 않는다.

Fig. 6과 7는 RMSE_k와 MAXE_k의 거동을 보여주고 있는데, 반복회수 k가 증가함에 따라 그 값이 감소되고 있음을 보여주고 있다. 그리고 이 그림에서 최초의 근사모델 $\mathbf{s}^0(\mathbf{x})$가 실제 함수의 분지 위치(basin location)를 가깝게 근사할수록, 적은 반복회수로 정밀한 근사 해를 얻을 수 있음을 확인

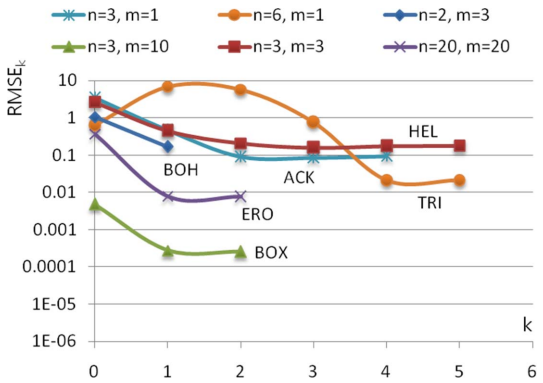


Fig. 6 RMS-based error between **f**(**x**) and $\mathbf{s}^k(\mathbf{x})$ near the true solution

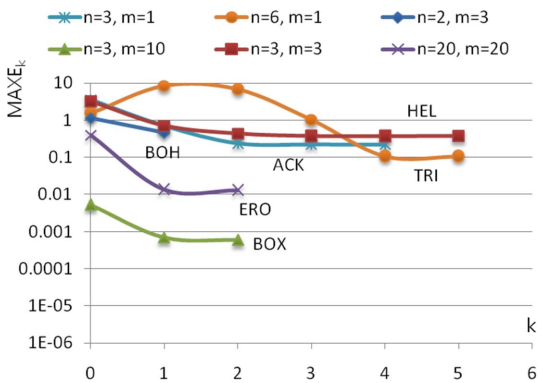


Fig. 7 Maximum error between **f**(**x**) and $\mathbf{s}^k(\mathbf{x})$ near the true solution

할 수 있다. 이는 $\mathbf{s}^0(\mathbf{x})$에 의해 계산 정확성과 함수 계산 회수 모두를 개선할 수 있음을 보여주는 것이다. 그림에서 $\mathbf{s}^0(\mathbf{x})$의 유사성이 높은 BOX, ERO가 낮은 HEL, TRI보다 적은 반복회수에 의해 수렴됨을 알 수 있다.

3.3 적용 예제: 최소 거리 문제

본 연구 알고리즘이 다양한 분야에 활용될 수 있음을 보이기 위한 적용 예제로서, 주어진 3차원 점과 B-스플라인 곡선 사이의 기하 거리(geometric distance)가 최소가 되는 곡선 상의 점을 찾는 문제를 다루고자 한다. B-스플라인 곡선[17]은 CAD/CAM 및 컴퓨터 그래픽스 분야에서 널리 사용되는 대표적인 곡선 모델이며, 이 곡선과 점 사이의 최소 거리 계산은 형상 모델링 커널 소프트웨어를 개발하는데 있어 필수 기능으로서 그 활용 범위가 넓다.

주어진 점 **P**(x, y, z)와 B-스플라인 곡선 **C**(t) 사이의 기하 거리가 최소가 되는 곡선 상의 파라미

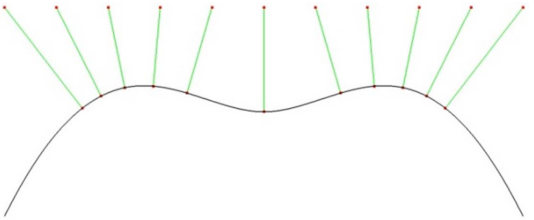


Fig. 8 Minimum distances between 3D points and a B-spline curve

터 t^*는 식 (17)과 같이 나타낼 수 있다.

$$t^* = \text{argmin} \|\mathbf{C}(t) - \mathbf{P}\|^2 \tag{17}$$

식 (17)는 본 연구에서 제시한 비선형 최소자승 문제와 일치하며 이때 $n = 1$, $m = 3$이다.

본 적용 예제에서는 Fig. 8과 같이 일정한 간격으로 배열된 다수의 점을 가지고 테스트하였다. 사용된 곡선 $\mathbf{C}(t)$의 차수는 3이고, 조정점의 개수는 5이며, 절점 벡터는 $\mathbf{T} = \{0,0,0,0,1/2,1,1,1,1\}$이다. 그리고 본 알고리즘의 종료를 위해 $\varepsilon_x = \varepsilon_F = 1.0e\text{-}6$을 사용하였다. 한편 최소 거리에 관한 해의 정확도는 식 (18)에 의해 계산할 수 있다.

$$\theta_t = \cos^{-1} \frac{\|\mathbf{C}'(t)\cdot(\mathbf{C}(t)-\mathbf{P})\|}{\|\mathbf{C}'(t)\|\|\mathbf{C}(t)-\mathbf{P}\|} \tag{18}$$

여기서 $\mathbf{C}'(t)$는 $\mathbf{C}(t)$를 t에 관한 미분한 것으로 접선 벡터를 의미하므로 식 (18)은 접선 벡터 $\mathbf{C}'(t)$와 차벡터 $\mathbf{C}(t) - \mathbf{P}$ 사이의 각도를 나타낸다.

총 11개의 점을 가지고 이 각도를 측정한 결과, 평균 각도가 $\theta_t = 89.998$도이며 이 값은 참값 90도에 비해 오차가 0.002에 해당한다. 그리고 식 (17)의 해를 구하기 위해 사용된 함수 계산 회수는 평균 5.8회(초기 근사 모델 생성하는데 3회, 반복 과정에서 평균 2.8회)이고, 계산 시간은 3.06 GHz Intel Xeon CPU 컴퓨터에서 평균 0.003초이다. 이는 1초에 300개의 최소 거리 문제를 풀 수 있는 수준이며 이때의 평균 오차는 0.002도임을 나타낸다. 이상의 결과로부터 본 연구에서 사용한 RBF 근사 모델 $\mathbf{s}(\mathbf{x})$가 최적 해 근처에서 B-스플라인 곡선 $\mathbf{C}(t)$을 정확하게 근사하였음을 확인할 수 있다.

4. 결 론

본 연구는 고비용 블랙박스 함수의 최소값을 RBF모델을 사용하여 저렴하게 계산하는 근사 최적화 알고리즘을 제시하였다. 이 알고리즘은 먼저 함수 계산 회수를 최대한 줄이기 위해 제시된 샘플링 방법을 사용하여 샘플 점을 생성하고, 이를 기반으로 가우시안 RBF모델을 생성한다. 그리고 LM알고리즘 최적화 과정을 수행하여 근사 해를 구하고, 이를 활용하여 RBF모델을 개선한다. 개선된 RBF모델은 다시 LM 최적화에 사용됨으로써 점차 최적 해에 접근해 나간다.

이상의 알고리즘은 6개의 수학 함수 문제를 통해 그 계산 성능(계산 정확성, 함수 계산 회수, 알고리즘 수렴성 등)의 우수성을 확인하였고, 1개의 적용 예제를 통해 타 분야로의 응용 가능성을 확인하였다. 이러한 수치 결과로부터 본 연구 알고리즘의 주요 특징을 살펴보면 다음과 같이 요약된다.

- 본 연구에서 제시하는 근사 최적화 알고리즘은 문제 특성 및 근사 모델링 기법에 의존하지 않는 계산 구조를 제공한다.
- 비선형 연립 방정식, 비선형 최소자승 문제, 비구속 최소화 문제에 적용 가능하다.
- 이차 수렴속도에 준하는 수렴성을 가지고 있다.
- 샘플 점에 의해 정의되는 목적함수의 최적화를 지원한다.
- 고비용 목적함수의 최적 값을 저렴하게 계산할 수 있다.
- 목적함수의 미분(예: 구배, 자코비안, 헤시안 등)을 요구하지 않는다.
- 목적함수 계산 회수의 추가를 통해 계산 정확성을 높일 수 있다.
- 목적함수가 불연속일지라도 이를 완만한 목적함수로 간주하여 해를 구한다.

본 연구의 향후 과제로서 본 알고리즘의 수렴성에 관한 이론적 고찰이 필요하며, 더불어 현 알고리즘의 적용 범위 확대를 위해 아래의 최적화 문제에 관한 추가 연구가 필요하다.

- 노이지(noise)가 있는 목적함수의 최적화 문제
- 정수형 변수에 의해 기술되는 최적화 문제
- 제약조건을 가진 다목적 최적화 문제

감사의 글

이 논문은 2016년 한국교통대학교 지원을 받아 수행하였음.

References

1. Park, S., 2015, An Approximate Optimization Technique Using Radial Basis Functions, *Transactions of the Korean National University of Transportation*, 50 (scheduled for publication).
2. Buhmann, M.D., 2003, *Radial Basis Functions*, Cambridge University Press.
3. deBoor, C. and Ron, A., 1992, Computational Aspects of Polynomial Interpolation in Several Variable, *Mathematics of Computation*, 58, pp.705-727.
4. Lebensztajn, L., Marretto, C.A.R., Costa, M.C., and Coulomb, J.-L., 2004, Kriging: A Useful Tool for Electromagnetic Device Optimization, *IEEE Trans Magn*, 40, pp.1196-1199.
5. Gutmann, H.M., 2001. A Radial Basis Function Method for Global Optimization, *Journal of Global Optimization*, 19(3), pp.201-227.
6. Clarke, S.M., Griebsch, J.H., and Simpson, T.W., 2005, Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses, *Journal of Mechanical Design*, 127, 1077-1087.
7. Forrester, A.I.J. and Keane, A.J., 2009, Recent Advances in Surrogate-based Optimization, *Progress in Aerospace Sciences*, 45, pp.50-79.
8. Bandler, J.W., Chen, Q.S., Dakroury, S.A., Mohamed, A.S., Bakr, M.H., Madsen, K., and Sondergaard, J., 2004, Space Mapping: The State of the Art, *IEEE Trans on Microwave Theory and Techniques*, 52(1), pp.337-361.
9. Hemker, P.W. and Echeverria, D., 2007, A Trust-region Strategy for Manifold-mapping Optimization, *Journal of Computational Physics*, 224, pp.464-475.
10. Alexandrov, N., Dennis, J.E., Lewis, R.M., and Torczon, V., 1998, A Trust Region Framework for Managing the Use of Approximation Models in Optimization, *Structural Optimization*, 15, pp.16-23.
11. Booker, A., Dennis, J., Frank, P., Serafini, D., Torczon, V., and Trosset, M., 1999, A Rigorous Framework for Optimization of Expensive Functions by Surrogates, *Structural Optimization*, 17, pp.1-13.
12. Madsen, K., Nielsen, H.B., and Tingleff, O., 2004, *Methods for Non-Linear Least Squares Problems (2nd ed.)*, Informatics and Mathematical Modelling, Technical University of Denmark, DTU.
13. Hedayat, A., Sloane, N., and Stufken, J., 1999, *Orthogonal Arrays: Theory and Applications*, Springer, Series in Statistics, Berlin: Springer.
14. McKay, M., Conover, W., and Beckman, R., 1979, A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code, *Technometrics*, 21, pp.239-245.
15. Wang, J.G. and Liu, G.R., 2002, On the Optimal Shape Parameters of Radial Basis Functions Used for 2D Meshless Methods, *Computer Methods in Applied Mechanics and Engineering*, 191, pp.2611-2630.
16. http://www.gamsworld.org/performance/selconglobal/selcongloballib.htm.
17. Piegl, L. and Tiller, W., 1995, *The NURBS Book*, Springer-Verlag.

박 상 근

1991년 포항공과대학교 학사
1993년 서울대학교 기계설계학과 석사
1997년 서울대학교 기계설계학과 박사
1997년~1999년 삼성SDS 정보기술연구소 책임연구원
2000년 서울대 BK21 기계분야사업단 계약교수
2000년~2002년 ㈜K&I 테크놀로지 책임연구원
2003년~현재 국립한국교통대학교 기계공학과 교수
관심분야: Computational Geometry, CAD/CAM/CAE, Design Optimization, Scientific Visualization