SOME COMPUTING RESULTS FOR NEWTON METHOD AND QUASI - NEWTON METHOD FOR UNCONSTRAINED OPTIMIZATION PROBLEM

Nguyen Dinh Dung
TNU - University of Information and Communication Technology

ARTICLE INFO

Received: 22/8/2022
Revised: 31/8/2022
Published: 31/8/2022

KEYWORDS

Convex optimization
Newton
Quansi-Newton
Hessen Matrix
Global optimization

ABSTRACT

The optimization problem has many effective and widespread applications in resource planning, machine design, automatic control, business administration, urban architecture that creating decision support systems in management and development of large systems. Nowadays, there are many effective algorithms published to solve optimization problems, including iterative algorithms such as Gradient algorithm, Newton's algorithm and variations of this algorithm that are applied in learning machine, deep learning, regression,... In this paper, we introduce an algorithm based on Newton method and Quansi - Newton method, they are effective methods of finding solutions for optimal problems when the objective function is a convex, then we give some computing results for the algorithm to illustrate the convergence of the method.

MỘT SỐ KẾT QUẢ TÍNH TOÁN ĐỐI VỚI THUẬT TOÁN NEWTON VÀ TỰA NEWTON CHO BÀI TOÁN TỐI ƯU KHÔNG RÀNG BUỘC

Nguyễn Đình Dũng
Trường Đại học Công nghệ thông tin và Truyền thông - ĐH Thái Nguyên

THÔNG TIN BÀI BÁO

Ngày nhận bài: 22/8/2022
Ngày hoàn thiện: 31/8/2022
Ngày đăng: 31/8/2022

TỪ KHÓA

Tối ưu lồi
Newton
Tựa newton
Ma trận Hessen
Tối ưu toàn cục

TÓM TẮT

Bài toán tối ưu có nhiều ứng dụng hiệu quả và rộng rãi trong quy hoạch tài nguyên, thiết kế chế tạo máy, điều khiển tự động, quản trị kinh doanh, kiến trúc đô thị, trong việc tạo nên các hệ hỗ trợ ra quyết định trong quản lý và phát triển các hệ thống lớn. Hiện nay, có nhiều thuật toán hữu hiệu được công bố nhằm giải quyết các bài toán tối ưu, có thể kể đến các thuật toán lặp như thuật toán Gradient, thuật toán Newton và các biến thể của thuật toán này được ứng dụng trong học máy, học sâu, hồi quy,... Trong phạm vi bài báo này, chúng tôi giới thiệu một thuật toán dựa trên phương pháp Newton và tựa Newton, đây là một phương pháp hiệu quả tìm nghiệm cho bài toán tối ưu khi hàm mục tiêu là phiếm hàm lồi và chúng tôi đưa ra một số kết quả tính toán đối với thuật toán để minh họa sự hội tụ của phương pháp.

DOI: https://doi.org/10.34238/tnu-jst.6386

Email: nddung@ictu.edu.vn

1. Giới thiệu

Xét bài toán tối ưu không ràng buộc

$$\min_{\mathbf{x}\in\mathbf{R}^n} f(\mathbf{x}). \tag{1}$$

Trong đó, $f(\mathbf{x})$ là phiếm hàm lồi có đạo hàm bậc hai trên $\mathbf{R}^n$. Bài toán tối ưu (1) cũng là bài toán được dẫn về từ nhiều bài toán thuộc các lĩnh vực khác nhau trong kinh tế, kỹ thuật. Để giải Bài toán (1) hiện nay có nhiều phương pháp giải khác nhau tuỳ thuộc vào hàm mục tiêu. Một trong những phương pháp đơn giản là phương pháp đường dốc nhất hay còn gọi là phương pháp Gradient descent [1], phương pháp này đơn giản và áp dụng cho một lớp khá rộng các hàm mục tiêu, nội dung phương pháp là đưa ra dãy lặp $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \lambda_k \nabla f(\mathbf{x}^{(k)})$, $\lambda_k > 0$, trong đó λ_k là độ dài bước được xác định theo quy tắc Armijo, phương pháp này có nhược điểm là tốc độ hội tụ tuyến tính. Để tăng tốc độ hội tụ của thuật toán thì phương pháp Newton là một lựa chọn khá tốt ([2] - [5]). Hiện nay, phương pháp này được mở rộng để tìm lời giải tối ưu cho hàm nhiều biến trên cơ sở khai triển Taylor, nghiệm của bài toán tối ưu được thực hiện theo dãy lặp $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left[\nabla^2 f(\mathbf{x}^{(k)})\right]^{-1} \nabla f(\mathbf{x}^{(k)})$, nếu hàm mục tiêu không có dạng toàn phương thì dãy lặp trên có thể phân kỳ hoặc hội tụ về cực tiểu địa phương hay hội tụ về điểm yên ngựa. Một biến thể của phương pháp Newton được giới thiệu trong [6] là phương pháp Newton suy rộng, nghiệm của bài toán tối ưu được tính toán theo dãy lặp $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \lambda_k \left[\nabla^2 f(\mathbf{x}^{(k)})\right]^{-1} \nabla f(\mathbf{x}^{(k)})$, trong đó λ_k được gọi là độ dài bước và được xác định nhờ các phương pháp tìm kiếm một chiều theo hướng $-\left[\nabla^2 f(\mathbf{x}^{(k)})\right]^{-1} \nabla f(\mathbf{x}^{(k)})$. Tuy nhiên trong một số bài toán thực tiễn khi dẫn về bài toán tối ưu mà hàm mục tiêu không tồn tại đạo hàm cấp 2, khi đó việc áp dụng phương pháp Newton là không khả thi. Để khắc phục hạn chế này, gần đây một số kết quả trong [7] và [8] đã đề xuất thuật toán tựa Newton khi tìm lời giải cho bài toán tối ưu phi tuyến cho thấy tính hiệu quả của phương pháp. Trong bài báo này, chúng tôi xây dựng thuật toán lặp dựa trên tư tưởng phương pháp Newton và tựa Newton khi tìm nghiệm tại mỗi bước lặp, trong đó có sự kế thừa thông tin nghiệm thành phần đã tính được ở lần lặp hiện tại thay vì sử dụng nghiệm đã tính được ở lần lặp trước đó. Các kết quả tính toán minh hoạ cho thuật toán được đưa ra để khẳng định cho sự hội tụ của thuật toán.

2. Phương pháp

2.1. Phương pháp Newton

Gọi $\mathbf{x}^*$ là điểm cực tiểu của phiếm hàm thì điều kiện cần là $\frac{\partial f}{\partial x_i}(\mathbf{x}^*) = 0$. Để xác định dãy lặp, ta sử dụng khai triển Taylor cho $f(\mathbf{x})$, ta có

$$f(\mathbf{x}) = f(\mathbf{x}^{(k)}) + \nabla f_k(\mathbf{x} - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)})^2 \mathbf{J}_k, \tag{2}$$

trong đó,

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}\right),$$

$\mathbf{J}_k$ là ma trận Hessen và được xác định là

$$\nabla^2 f(\mathbf{x}) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{i=1,...,n; j=1,...,n}.$$

Từ (2) ta có

$$\nabla f = \nabla f_k + (\mathbf{x} - \mathbf{x}^{(k)})\mathbf{J}_k. \tag{3}$$

Vậy ta có dãy lặp

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{J}_k)^{-1}\nabla f_k,\ k = 1, 2, 3, ..., n. \tag{4}$$

Công thức (4) được gọi là công thức lặp Newton có tốc độ hội tụ cấp 2. Thuật toán được thực hiện như sau:

Thuật toán 1

function $\mathbf{x}$=*newton*$(\mathbf{x}^{(1)}, \varepsilon)$;
k=1;
while*(*$\|\nabla f_k\| > \varepsilon$*)*
$\boldsymbol{d}^{(k)} = -(\boldsymbol{J}_k)^{-1}\nabla f_k$*;*
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$;
$k = k + 1$;
end;
$\mathbf{x} = \mathbf{x}^{(k+1)}$;

Theo thuật toán này, tại bước tính $\mathbf{d}^{(k)} = -(\mathbf{J}_k)^{-1}\nabla f_k$ và $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$ chúng tôi thực hiện kế thừa thành phần phần $x_i^{(k+1)}$ đã tính được để tính $x_j^{(k+1)}$, $j = i+1, ..., n$. Vậy (4) được thay thế bằng công thức $x_i^{(k+1)} = x_i^{(k)} + d_i^{(k)}$, trong đó

$$d_1^{(k)} = -\sum_{j=1}^{n} \left[(\mathbf{J}_k)^{-1}\right]_{i,j} \nabla f(x_j^{(k)}),$$

$$\begin{aligned} d_i^{(k)} = &-\sum_{j=1}^{i-1} \left[(J_k)^{-1}\right]_{i,j} \nabla f(x_j^{(k+1)}) \\ &-\sum_{j=i+1}^{n} \left[(J_k)^{-1}\right]_{i,j} \nabla f(x_j^{(k)}),\ i = 2, ..., n. \end{aligned} \tag{5}$$

Vậy từ đó thuật toán Newton được cập nhật như sau:

Thuật toán 2

function $\mathbf{x}$=*newton*$(\mathbf{x}^{(1)}, \varepsilon)$;
k=1;
n=length($\boldsymbol{x}^{(0)}$*);*
while*(*$\|\nabla f_k\| > \varepsilon$*)*
$\mathbf{d}^{(k)} = \mathbf{0}$;
for *j=1:n*
$d_1^{(k)} = d_1^{(k)} - \left[(J_k)^{-1}\right]_{1,j} \nabla f(x_j^{(k)})$;
end;
$x_1^{(k+1)} = x_1^{(k)} + d_1^{(k)}$;
for *i=2:n*
for *j=1:i-1*
$d_i^{(k)} = d_i^{(k)} - \left[(J_k)^{-1}\right]_{i,j} \nabla f(x_j^{(k+1)})$;
end;
for *j=i:n*
$d_i^{(k)} = d_i^{(k)} - \left[(J_k)^{-1}\right]_{i,j} \nabla f(x_j^{(k)})$;
end;
$x_i^{(k+1)} = x_i^{(k)} + d_i^{(k)}$;
end;
$k = k + 1$;
end;
$\mathbf{x} = \mathbf{x}^{(k+1)}$;

Trong trường hợp hàm mục tiêu không khả vi bậc 2, khi đó thay vì sử dụng phương pháp Newton ta sẽ sử dụng phương pháp tựa Newton.

2.2. Phương pháp tựa Newton

Ý tưởng của phương pháp tựa Newton [8] là xuất phát từ công thức (4), ta xấp xỉ ma trận Hessen $[\mathbf{J}_k]$ bởi ma trận $[\mathbf{B}_k]$.
Vậy từ (2) ta có dãy lặp

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \lambda_k \left[\mathbf{B}_k\right]^{-1} \nabla f_k,\ k = 1..n, \tag{6}$$

trong đó, λ_k là độ dài bước được xác định theo hướng $\mathbf{S}_k = -\left[B_k\right]^{-1} \nabla f_k$, đại lượng này có thể thay đổi ở mỗi bước lặp và thoả mãn điều kiện Wolfe [8],

$$\begin{aligned} &f(\mathbf{x}^{(k)} + \lambda_k \mathbf{S}_k) \le f(\mathbf{x}^{(k)}) + c_1 \lambda_k \mathbf{S}_k^T \nabla f(\mathbf{x}^{(k)}), \\ &-\mathbf{S}_k^T \nabla f(\mathbf{x}^{(k)} + \lambda_k \mathbf{S}_k) \le -c_2 \mathbf{S}_k^T \nabla f(\mathbf{x}^{(k)}), \\ &0 < c_1 < c_2 < 1. \end{aligned} \tag{7}$$

Thuật toán λ_k xác định như sau:
Thuật toán 3

function λ_k=*LineSearch*$(f, \mathbf{x}^{(k)}, \mathbf{S}_k)$;
Khởi tạo các hằng số c_1, c_2, β *thỏa mãn*

$$0 < c_1 < c_2 < 1; 0 < \beta < 1;$$

$\lambda = \lambda_0$;
while(λ *chưa thỏa mãn (7)*)
$\lambda = \beta\lambda$;
end;
$\lambda_k = \lambda$;

Trong trường hợp hàm mục tiêu là hàm lồi thì điểm tối ưu thu được chính là nghiệm tối ưu toàn cục của Bài toán (1). Dễ thấy trong (6), nếu $[\mathbf{B}_k] = \mathbf{I}$ thì công thức lặp (6) chính là phương pháp đường dốc nhất đã được công bố trong [9].
$\mathbf{B}_k$ là ma trận xấp xỉ cho ma trận Hessen và thỏa mãn điều kiện

$$\nabla f(\mathbf{x}^{(k)}) = \nabla f(\mathbf{x}^{(k-1)}) - [\mathbf{B}_k]\,(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}). \tag{8}$$

Tại điểm $x^{(k+1)}$, ta có

$$\nabla f(\mathbf{x}^{(k+1)}) = \nabla f(\mathbf{x}^{(k)}) - [\mathbf{B}_{k+1}]\,(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}), \tag{9}$$

hay có thể viết

$$[\mathbf{B}_{k+1}]\,\mathbf{d}_k = \mathbf{g}_k, \tag{10}$$

trong đó

$$\mathbf{d}_k = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)},\ \mathbf{g}_k = \nabla f_{k+1} - \nabla f_k.$$

Công thức (10) có thể viết lại như sau:

$$\mathbf{d}_k = \left[\mathbf{B}_{k+1}\right]^{-1} \mathbf{g}_k, \tag{11}$$

ở đây, $\mathbf{B}_{k+1}$ là ma trận đối xứng xác định dương và được cập nhật theo công thức

$$[\mathbf{B}_{k+1}] = [\mathbf{B}_k] + C\mathbf{Z}\mathbf{Z}^T. \tag{12}$$

Từ (10), ta có

$$C\mathbf{Z} = \frac{\mathbf{g}_k - [\mathbf{B}_k]\,\mathbf{d}_k}{\mathbf{Z}^T \mathbf{d}_k}.$$

Chọn $\mathbf{Z} = \mathbf{g}_k - [\mathbf{B}_k]\,\mathbf{d}_k$, khi đó

$$[\mathbf{B}_{k+1}] = [\mathbf{B}_k] + \frac{(\mathbf{g}_k - [\mathbf{B}_k]\,\mathbf{d}_k)\,(\mathbf{g}_k - [\mathbf{B}_k]\,\mathbf{d}_k)^T}{(\mathbf{g}_k - [\mathbf{B}_k]\,\mathbf{d}_k)^T\,\mathbf{d}_k}. \tag{13}$$

Ngoài cách tiếp cận theo (12), ta cũng có thể sử dụng cách tính

$$[\mathbf{B}_{k+1}] = [\mathbf{B}_k] + C_1\mathbf{Z}_1\mathbf{Z}_1^T + C_2\mathbf{Z}_2\mathbf{Z}_2^T.$$

Theo cách tiếp cận này, thì việc cập nhật ma trận B_{k+1} được dẫn về công thức lặp

$$[\mathbf{B}_{k+1}] = [\mathbf{B}_k] + \frac{\mathbf{g}_k\mathbf{g}_k^T}{\mathbf{g}_k^T\mathbf{d}_k} - \frac{([\mathbf{B}_k]\,\mathbf{d}_k)\,([\mathbf{B}_k]\,\mathbf{d}_k)^T}{\mathbf{d}_k^T\,[\mathbf{B}_k]\,\mathbf{d}_k}. \tag{14}$$

Như vậy, thuật toán tìm nghiệm của Bài toán (1) được thực hiện như sau:

Thuật toán 4

function $\mathbf{x}$=*QNewton*$(\mathbf{x}^{(1)}, \varepsilon)$;
k=1;
$[\boldsymbol{B}_k] = \boldsymbol{I}$;
while$(\|\nabla f_k\| > \varepsilon)$
$\mathbf{S}_k = -\,[\mathbf{B}_k]^{-1}\,\nabla f_k$;
$\lambda_k = LineSearch(f, \mathbf{x}^{(k)}, \mathbf{S}_k)$;
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda_k\mathbf{S}_k$;
$\mathbf{d}_k = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$;
$\mathbf{g}_k = \nabla f_{k+1} - \nabla f_k$;
Cập nhật $[\mathbf{B}_{k+1}]$ theo (13) hoặc (14);
$k = k + 1$;
end;
$\mathbf{x} = \mathbf{x}^{(k+1)}$;

Theo thuật toán này, xuất phát từ điểm $x^{(1)}$, dãy lặp (14) hội tụ về điểm tối ưu cục bộ x^* và thoả mãn

$$f(x^{(1)}) \geq ... \geq f(x^{(k)}) \geq ... \geq f(x^*)$$

Trong trường hợp hàm mục tiêu là hàm lồi thì điểm tối ưu thu được chính là nghiệm tối ưu toàn cục của Bài toán (1).

Minh hoạ cho kết quả lý thuyết, sau đây là một số kết quả tính toán thử nghiệm của thuật toán.

3. Một số kết quả tính toán

Trong mục này, chúng tôi thực hiện một số tính toán thử nghiệm để kiểm tra lý thuyết về sự hội tụ của thuật toán được trình bày trong mục 2. Sau đây là một số dữ kiện cho trước để thực hiện thuật toán:

Hàm mục tiêu

$$f(\mathbf{x}) = \sum_{i=1}^{10}(x_i - 1)^4 \tag{15}$$

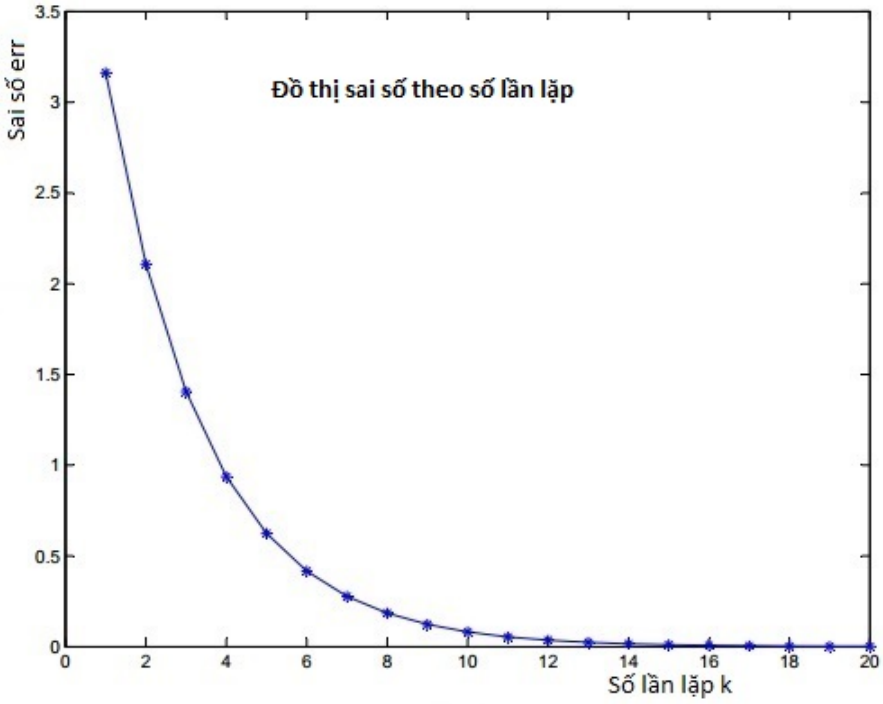
Xấp xỉ đầu: $\mathbf{x}^{(1)} = (0, 0, ..., 0)$.

Dễ thấy nghiệm đúng của Bài toán (1) là $\mathbf{x}^* = (1, 1, ..., 1), f(\mathbf{x}^*) = 0$. Hàm mục tiêu (15) khả vi mọi cấp nên tồn tại ma trận Hessen, vậy ta hoàn toàn có thể áp dụng thuật toán Newton để giải Bài toán (1). Đặt $err = \left\|\mathbf{x}^{(k)} - \mathbf{x}^*\right\|_2$, ta có kết quả tính toán minh hoạ cho sự hội tụ của thuật toán Newton được cho trong Bảng 1.

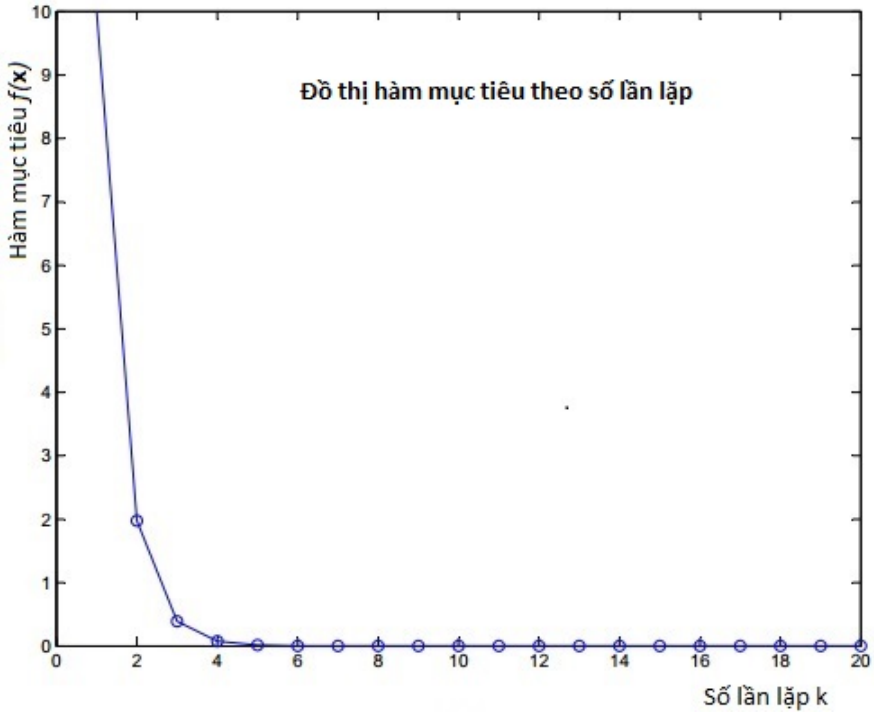
Kết quả tính toán ở Bảng 1 cho thấy nghiệm xấp xỉ tìm được hội tụ về nghiệm đúng của bài toán theo số lần lặp. Đồ thị Hình 1, Hình 2 minh hoạ cho sự hội tụ của thuật toán. Từ Hình 1 và Hình 2, cho thấy hàm sai số và hàm mục tiêu là các hàm đơn điệu giảm theo số lần lặp, đã cho thấy nghiệm xấp xỉ hội tụ về nghiệm đúng của Bài toán (1).

Bảng 1. *Nghiệm xấp xỉ của Bài toán (1) thu được từ Thuật toán 2*

$\mathbf{x}^{(k)}$	k=**5**	k=**10**	k=**15**	k=**20**
$x_1^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_2^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_3^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_4^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_5^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_6^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_7^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_8^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_9^{(k)}$	0,8025	0,9740	0,9966	0,9995
$x_{10}^{(k)}$	0,8025	0,9740	0,9966	0,9995
err	0,6246	0,0823	0,0108	0,0014



Hình 1. *Đồ thị sai số theo số lần lặp với số lần lặp k=1,2,...,20 thu được từ Thuật toán 2*



Hình 2. *Đồ thị hàm mục tiêu theo số lần lặp (số lần lặp k=1,2,. . . ,20) thu được từ Thuật toán 2*

Bây giờ, ta xét hàm mục tiêu sau:

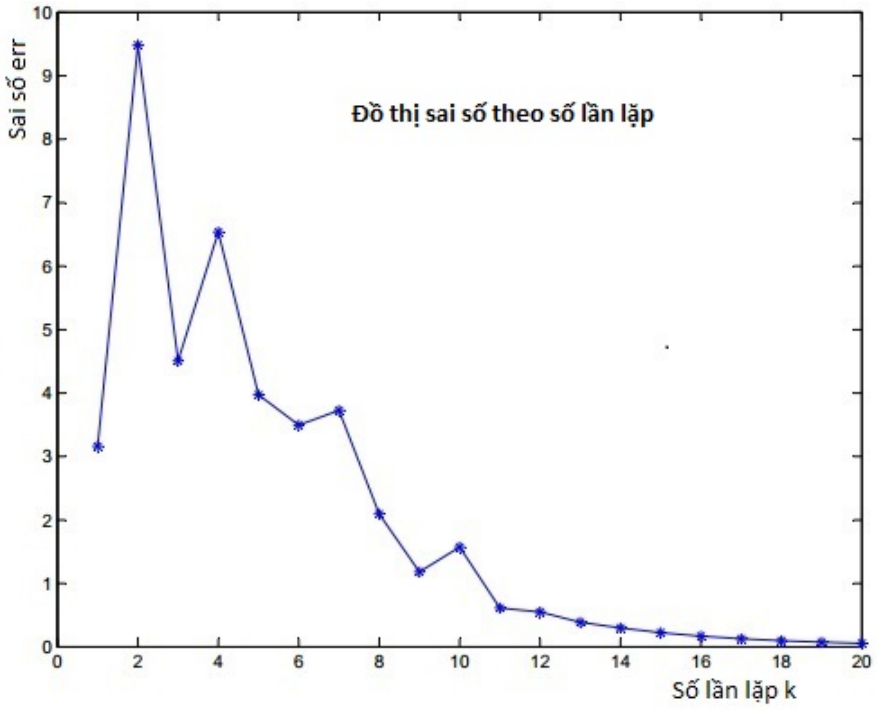
$$f(x) = \begin{cases} \sum_{i=1}^{10} (x_i - 1)^4 & \exists x_i < 1 \\ \sum_{i=1}^{10} (x_i - 1)\sqrt{x_i - 1} & \forall x_i \geq 1 \end{cases} \tag{16}$$

Hàm mục tiêu không tồn tại đạo hàm cấp 2 tại $x^* = (1, 1, ..., 1)$. Vậy để tìm nghiệm cho Bài toán (1) với hàm mục tiêu (16), ta thực hiện Thuật toán 4 với ma trận xấp xỉ ma trận Hessen được cập nhật theo công thức (14). Các kết quả tính toán được cho trong Bảng 2.

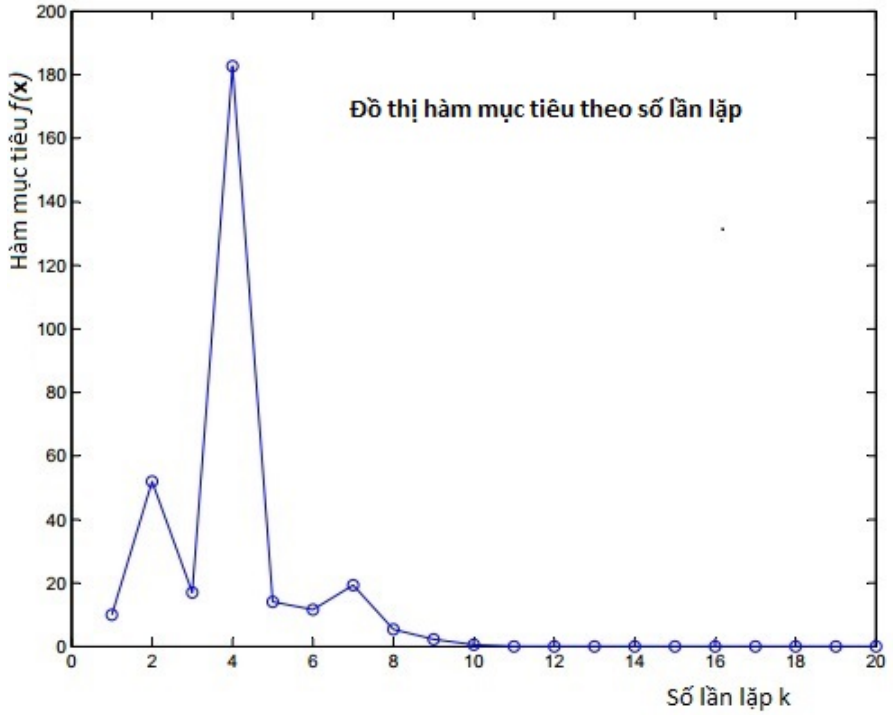
Bảng 2. *Nghiệm xấp xỉ của Bài toán (1) thu được từ Thuật toán 4*

$\mathbf{x}^{(k)}$	k=**5**	k=**10**	k=**15**	k=**20**
$x_1^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_2^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_3^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_4^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_5^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_6^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_7^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_8^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_9^{(k)}$	2,2566	0,5028	0.9292	0,9826
$x_{10}^{(k)}$	2,2566	0,5028	0.9292	0,9826
err	3,9737	1,5722	0.2238	0,0550

Số liệu ở Bảng 2 cho thấy nghiệm xấp xỉ tìm được hội tụ về nghiệm đúng của bài toán theo số lần lặp. Đồ thị Hình 3, Hình 4 minh hoạ cho sự hội tụ của thuật toán.



Hình 3. *Đồ thị sai số theo số lần lặp với số lần lặp k=1,2,. . . ,20 thu được từ Thuật toán 4*



Hình 4. *Đồ thị hàm mục tiêu theo số lần lặp (số lần lặp k=1,2,. . . ,20) thu được từ Thuật toán 4*

Từ đồ thị sai số và hàm mục tiêu, ta có thể thấy phương pháp tựa Newton có ưu điểm không yêu cầu hàm mục tiêu khả vi bậc 2 nhưng tốc độ hội tụ khá chậm so với phương pháp Newton. Hàm sai số và hàm mục tiêu không phải là các hàm đơn điệu giảm theo số lần lặp, nhưng có xu hướng giảm dần, điều này cũng khẳng định nghiệm xấp xỉ hội tụ về nghiệm đúng của bài toán.

4. Kết luận

Trong bài báo này, chúng tôi đề xuất một thuật toán lặp giải bài toán tối ưu lồi không ràng buộc dựa trên phương pháp lặp Newton và tựa Newton, trong đó có sự kế thừa thông tin nghiệm thành phần đã tính được ở lần lặp hiện tại thay vì sử dụng nghiệm đã tính được ở lần lặp trước đó. Các kết quả tính toán theo thuật toán được thực hiện trên môi trường Matlab 2014, kết quả số đã khẳng định được sự hội tụ của phương pháp và phù hợp với lý thuyết đưa ra trong bài báo.

Lời cảm ơn

Các kết quả nghiên cứu trong bài báo này là một phần của Đề tài cấp cơ sở "*Giải pháp ứng dụng công nghệ thông tin trong bảo mật lợi nhuận, nâng cao hiệu quả hoạt động kinh doanh đối với doanh nghiệp vừa và nhỏ*" mang mã số T2022-07-11. Chúng tôi xin trân trọng cám ơn quỹ phát triển khoa học và công nghệ Trường Đại học Công nghệ thông tin và Truyền thông đã tài trợ cho nghiên cứu này.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] H. B. Curry, "The Method of Steepest Descent for Non-linear Minimization Problems," *Quart. Appl. Math*, vol. 2, no. 3, pp.258–261, 1944.

[2] K. Dmitry, M. K. Mishchenko, and R. Peter, "Stochastic Newton and cubic Newton methods with simple local linear-quadratic rates," *arXiv*, vol. 03, no.8, pp.62-79, 2019.

[3] R.Weerakoon and J. Simon, "A New Derivation of the Classical Newton's Method using Rectangular Rule to Compute Integrals," *Applied Mathematics Lettres*, vol.13, no.9, pp.87-93, 2000.

[4] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior Methods for Non Linear Optimization," *SIAM*, vol 44, no.4, pp. 525-597, 2003.

[5] R. Fletcher, *Practical Methods of Optimization,* Second edition, John Wiley and Sons, Chichester, 2000, pp. 40-46.

[6] K. Geleta and R.K. Vasudeva, "Quasi-Newton Line Search Algorithm for solving unconstrained non-linear least square Optimization Problem," *International Journal of Basic and Applied Sciences*, vol. 13, no.4, pp. 11-21, 2021.

[7] Z. Povale, "Quasi-Newton Method for Multi-Objective Optimization," *Journal of Computational and Applied Mathematics*, vol. 10, no.1, pp. 765-777, 2014.

[8] K. Geleta and R.K. Vasudeva, "Quasi-Newton Line Search Algorithm for solving unconstrained non-linear least square Optimization Problem," *International Journal of Basic and Applied Sciences*, vol.13, no.1, pp. 9–18, 2021.

[9] Y.Gonglin, L. Sha, and W. Zengxin, "A Line Search Algorithm for Unconstrained Optimization," *Journal of Software Engineering and Applications*, vol.3, no.5, pp. 503–509, 2021.