Chapter 8
A Conservation Law Method Based on Optimization

This chapter is organized as follows: In Sect. 8.1, we warm up with an analytical solution for simple 1-D quadratic function. In Sect. 8.2, we propose the artificially dissipating energy algorithm, energy conservation algorithm, and the combined algorithm based on the symplectic Euler scheme, and remark a second-order scheme—the Störmer–Verlet scheme. In Sect. 8.3, we propose the locally theoretical analysis for high-speed convergence. Section 8.4 proposes the experimental demonstration. In Sect. 8.4, we propose the experimental result for the proposed algorithms on strongly convex, non-strongly convex, and non-convex functions in high dimension. Finally, we propose some perspective view for the proposed algorithms and two adventurous ideas based on the evolution of Newton's second law—fluid and quantum.

8.1 Warm-up: An Analytical Demonstration for Intuition

For a simple 1-D function with ill-conditioned Hessian, $f(x) = \frac{1}{200}x^2$ with the initial position at $x_0 = 1000$. The solution and the function value along the solution for (3.9) are given by

$$\begin{cases} x(t) = x_0 e^{-\frac{1}{100}t} & (8.1) \\ f(x(t)) = \dfrac{1}{200} x_0^2 e^{-\frac{1}{50}t}. & (8.2) \end{cases}$$

Parts of this chapter is in the paper titled "A Conservation Law Method in Optimization" by Bin Shi et al. (2017) published by 10th NIPS Workshop on Optimization for Machine Learning.

B. Shi, S. S. Iyengar, *Mathematical Theories of Machine Learning - Theory and Applications*, https://doi.org/10.1007/978-3-030-17076-9_8

The solution and the function value along the solution for (3.10) with the optimal friction parameter $\gamma_t = \frac{1}{5}$ are

$$\begin{cases} x(t) = x_0 \left(1 + \dfrac{1}{10}t\right) e^{-\frac{1}{10}t} & (8.3) \\ f(x(t)) = \dfrac{1}{200} x_0^2 \left(1 + \dfrac{1}{10}t\right)^2 e^{-\frac{1}{5}t}. & (8.4) \end{cases}$$

The solution and the function value along the solution for (3.12) are

$$\begin{cases} x(t) = x_0 \cos\left(\dfrac{1}{10}t\right) \quad \text{and} \quad v(t) = x_0 \sin\left(\dfrac{1}{10}t\right) & (8.5) \\ f(x(t)) = \dfrac{1}{200} x_0^2 \cos^2\left(\dfrac{1}{10}t\right), & (8.6) \end{cases}$$

stop at the point that $|v|$ arrives maximum. Combined with (8.2), (8.4), and (8.6) with stop at the point that $|v|$ arrives maximum, the function value approximating $f(x^\star)$ is shown as below.

From the analytical solution for local convex quadratic function with maximum eigenvalue L and minimum eigenvalue μ, in general, the step size by $\frac{1}{\sqrt{L}}$ for momentum method and Nesterov's accelerated gradient method, hence the simple estimate for iterative times is approximately

$$n \sim \frac{\pi}{2}\sqrt{\frac{L}{\mu}}.$$

Hence, the iterative times n is proportional to the reciprocal of the square root of minimal eigenvalue $\sqrt{\mu}$, which is essentially different from the convergence rate of the gradient method and momentum method (Fig. 8.1).

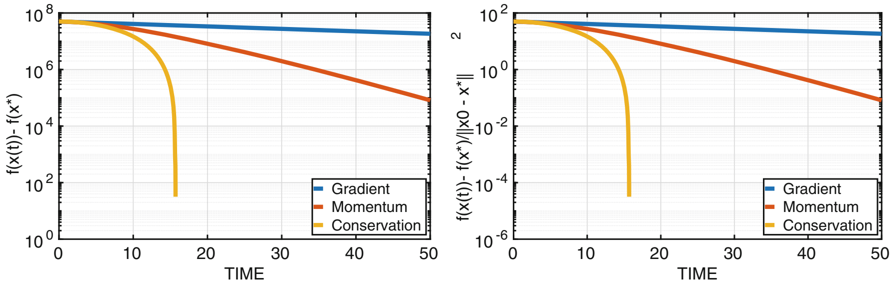


Fig. 8.1 Minimizing $f(x) = \frac{1}{200}x^2$ by the analytical solution for (8.2), (8.4), and (8.6) with stop at the point that $|v|$ arrives maximum, starting from $x_0 = 1000$ and the numerical step size $\Delta t = 0.01$

8.2 Symplectic Scheme and Algorithms

In this chapter, we utilize the first-order symplectic Euler scheme from numerically solving Hamiltonian system as below to propose the corresponding artificially dissipating energy algorithm to find the global minima for convex function, or local minima in non-convex function

$$\begin{cases} x_{k+1} = x_k + hv_{k+1} \\ v_{k+1} = v_k - h\nabla f(x_k). \end{cases} \tag{8.7}$$

Then by the observability of the velocity, we propose the energy conservation algorithm for detecting local minima along the trajectory. Finally, we propose a combined algorithm to find better local minima between some local minima.

Remark 8.1 In all the algorithms below, the symplectic Euler scheme can be replaced by the Störmer–Verlet scheme

$$\begin{cases} v_{k+1/2} = v_k - \frac{h}{2}\nabla f(x_k) \\ x_{k+1} = x_k + hv_{k+1/2} \\ v_{k+1} = v_{k+1/2} - \frac{h}{2}\nabla f(x_{k+1}). \end{cases} \tag{8.8}$$

This works better than the symplectic scheme even if doubling step size and keeping the left–right symmetry of the Hamiltonian system. The Störmer–Verlet scheme is the natural discretization for 2nd-order ODE which is named as leap-frog scheme in PDEs

$$x_{k+1} - 2x_k + x_{k-1} = -h^2\nabla f(x_k). \tag{8.9}$$

We remark that the discrete scheme (8.9) is different from the finite difference approximation by the forward Euler method to analyze the stability of 2nd ODE in [SBC14], since the momentum term is biased.

8.2.1 Artificially Dissipating Energy Algorithm

Firstly, the artificially dissipating energy algorithm based on (8.7) is proposed as below.

Algorithm 1 Artificially dissipating energy algorithm

```
1: Given a starting point $x_0 \in \mathbf{dom}(f)$
2: Initialize the step length $h$, maxiter, and the velocity variable $v_0 = 0$
3: Initialize the iterative variable $v_{iter} = v_0$
4: while $\|\nabla f(x)\| > \epsilon$ and $k <$ maxiter do
5:     Compute $v_{iter}$ from the below equation in (8.7)
6:     if $\|v_{iter}\| \leq \|v\|$ then
7:         $v = 0$
8:     else
9:         $v = v_{iter}$
10:    end if
11:    Compute $x$ from the above equation in (8.7)
12:    $x_k = x$;
13:    $f(x_k) = f(x)$;
14:    $k = k + 1$;
15: end while
```

Remark 8.2 In the actual Algorithm 1, the codes in line 15 and 16 are not needed in the while loop in order to speed up the computation.

A Simple Example for Illustration

Here, we use a simple convex quadratic function with ill-conditioned eigenvalue for illustration as below:

$$f(x_1, x_2) = \frac{1}{2}\left(x_1^2 + \alpha x_2^2\right), \tag{8.10}$$

of which the maximum eigenvalue is $L = 1$ and the minimum eigenvalue is $\mu = \alpha$. Hence the scale of the step size for (8.10) is

$$\frac{1}{L} = \sqrt{\frac{1}{L}} = 1.$$

In Fig. 8.2, we demonstrate the convergence rate of gradient method, momentum method, Nesterov's accelerated gradient method, and artificially dissipating energy method with the common step size $h = 0.1$ and $h = 0.5$, where the optimal friction parameter for momentum method $\gamma = \frac{1-\sqrt{\alpha}}{1+\sqrt{\alpha}}$ with $\alpha = 10^{-5}$. A further result for comparison with the optimal step size in gradient method $h = \frac{2}{1+\alpha}$, the momentum method $h = \frac{4}{(1+\sqrt{\alpha})^2}$, and Nesterov's accelerated gradient method with $h = 1$ and the artificially dissipating energy method with $h = 0.5$ is shown in Fig. 8.3.

With the illustrative convergence rate, we need to learn the trajectory. Since the trajectories of all the four methods are so narrow in ill-condition function in (8.10), we use a relatively good-conditioned function to show it as $\alpha = \frac{1}{10}$ in Fig. 8.4.

The fact highlighted in Fig. 8.4 demonstrates the gradient correction decreases the oscillation when compared with the momentum method. A clearer observation

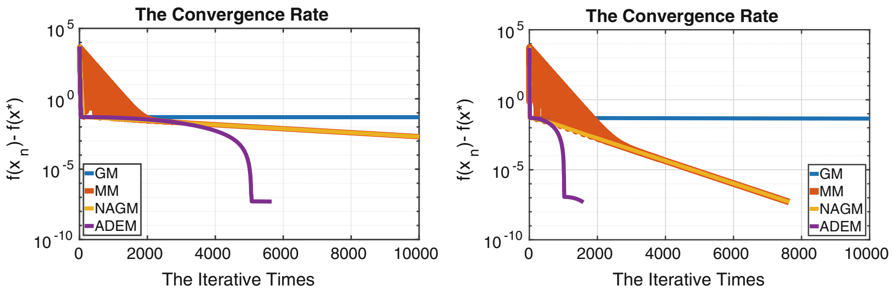


Fig. 8.2 Minimizes the function in (8.10) for artificially dissipating energy algorithm comparing with gradient method, momentum method, and Nesterov's accelerated gradient method with stop criteria $\epsilon = 1e - 6$. The step size: Left: $h = 0.1$; Right: $h = 0.5$

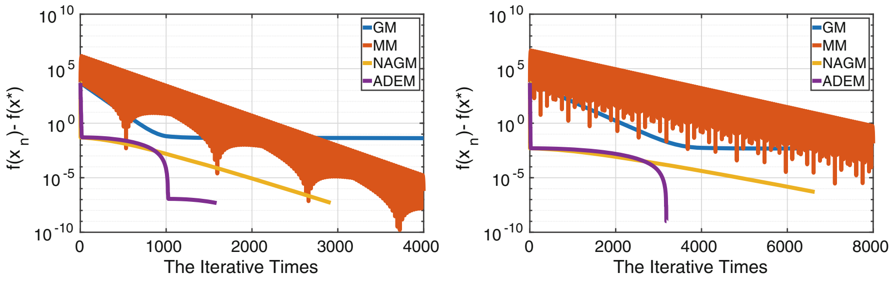
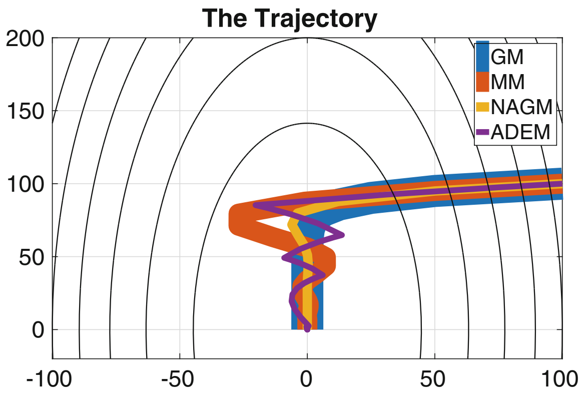


Fig. 8.3 Minimizes the function in (8.10) for artificially dissipating energy algorithm comparing with gradient method, momentum method, and Nesterov's accelerated gradient method with stop criteria $\epsilon = 1e - 6$. The Coefficient α: Left: $\alpha = 10^{-5}$; Right: $\alpha = 10^{-6}$

Fig. 8.4 The trajectory for gradient method, momentum method, Nesterov's accelerated method, and artificially dissipating energy method for the function (8.10) with $\alpha = 0.1$



is the artificially dissipating method shares the same property with the other three methods by the law of nature, that is, if the trajectory comes into the local minima in one dimension it will not leave it very far. However, from Figs. 8.2 and 8.3, we see the more rapid convergence rate from using the artificially dissipating energy method.

8.2.2 Detecting Local Minima Using Energy Conservation Algorithm

Here, the energy conservation algorithm based on (8.7) is proposed as below.

Algorithm 2 Energy conservation algorithm

```
 1: Given a starting point $x_0 \in \mathbf{dom}(f)$
 2: Initialize the step size $h$ and the maxiter
 3: Initialize the velocity $v_0 > 0$ and compute $f(x_0)$
 4: Compute the velocity $x_1$ and $v_1$ from Eq. (8.7), and compute $f(x_1)$
 5: for $k = 1 : n$ do
 6:     Compute $x_{k+1}$ and $v_{k+1}$ from (8.7)
 7:     Compute $f(x_{k+1})$
 8:     if $\|v_k\| \geq \|v_{k+1}\|$ and $\|v_k\| \geq \|v_{k-1}\|$ then
 9:         Record the position $x_k$
10:     end if
11: end for
```

Remark 8.3 In Algorithm 2, we can set $v_0 > 0$ so that the total energy is large enough to climb up some high peak. Similar to Algorithm 1 defined earlier, the function value $f(x)$ is not need in the while loop in order to speed up the computation.

The Simple Example for Illustration

Here, we use the non-convex function for illustration as below:

$$f(x) = \begin{cases} 2\cos(x), & x \in [0, 2\pi] \\ \cos(x) + 1, & x \in [2\pi, 4\pi] \\ 3\cos(x) - 1, & x \in [4\pi, 6\pi], \end{cases} \tag{8.11}$$

which is the 2nd-order smooth function but not 3rd-order smooth. The maximum eigenvalue can be calculated as below:

$$\max_{x \in [0, 6\pi]} |f''(x)| = 3.$$

The step length is set $h \sim \sqrt{\frac{1}{L}}$. We illustrate that Algorithm 2 simulates the trajectory and find the local minima in Fig. 8.5.

Another 2D potential function is shown as below:

$$f(x_1, x_2) = \frac{1}{2}\left[(x_1 - 4)^2 + (x_2 - 4)^2 + 8\sin(x_1 + 2x_2)\right], \tag{8.12}$$

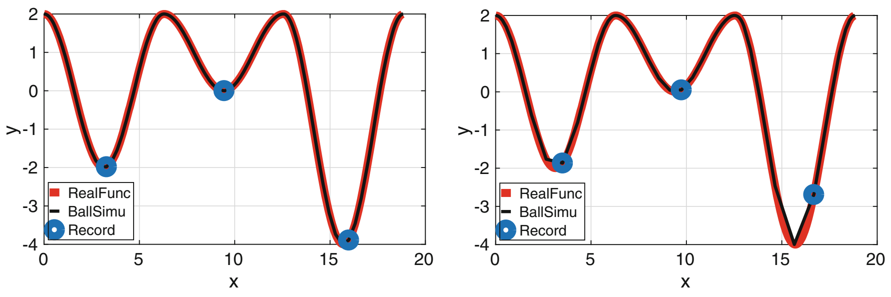


Fig. 8.5 Left: the step size $h = 0.1$ with 180 iterative times. Right: the step size $h = 0.3$ with 61 iterative times

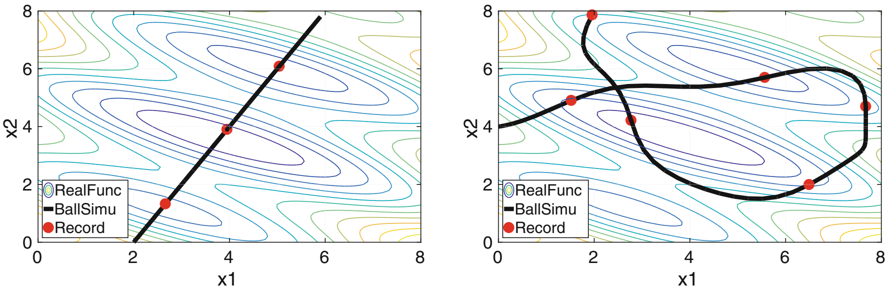


Fig. 8.6 The common step size is set $h = 0.1$. Left: the position at $(2, 0)$ with 23 iterative times. Right: the position at $(0, 4)$ with 62 iterative times

which is the smooth function with domain in $(x_1, x_2) \in [0, 8] \times [0, 8]$. The maximum eigenvalue can be calculated as below:

$$\max_{x\in[0,6\pi]} |\lambda(f''(x))| \geq 16.$$

The step length is set $h \sim \sqrt{\frac{1}{L}}$. We illustrate that Algorithm 2 simulates the trajectory and find the local minima as in Fig. 8.6.

Remark 8.4 We point out that the energy conservation algorithm for detecting local minima along the trajectory cannot detect the saddle point in the sense of almost every, since the saddle point in the original function $f(x)$ is also a saddle point for the energy function $H(x, v) = \frac{1}{2}\|v\|^2 + f(x)$. The proof process is fully the same in [LSJR16].

8.2.3 *Combined Algorithm*

Finally, we propose the comprehensive algorithm combining the artificially dissipating energy algorithm (Algorithm 1) and the energy conservation algorithm (2) to find global minima.

Algorithm 3 Combined algorithm

```
1: Given some starting points $x_{0,i} \in \mathbf{dom}(f)$ with $i = 1, \ldots, n$
2: Implement algorithm 2 detecting the position there exists local minima, noted as $x_j$ with $j = 1, \ldots, m$
3: Implement algorithm 1 from the result on line 2 finding the local minima, noted as $x_k$ with $k = 1, \ldots, l$
4: Comparison of $f(x_k)$ with $k = 1, \ldots, l$ to find global minima.
```

Remark 8.5 We remark that the combined algorithm (Algorithm 3) cannot guarantee to find global minima if the initial position is not ergodic. The tracking local minima is dependent on the trajectory. However, the time of computation and precision based on the proposed algorithm is far better than the large sampled gradient method. Our proposed algorithm first makes the identified global minima to become possible.

8.3 An Asymptotic Analysis for the Phenomena of Local High-Speed Convergence

In this section, we analyze the phenomena of high-speed convergence shown in Figs. 8.1, 8.2, and 8.3. Without loss of generality, we use the translate transformation $y_k = x_k - x^\star$ ($x^\star$ is the point of local minima) and $v_k = v_k$ into (8.7), shown as below:

$$\begin{cases} y_{k+1} = y_k + hv_{k+1} \\ v_{k+1} = v_k - h\nabla f(x^\star + y_k), \end{cases} \tag{8.13}$$

the locally linearized scheme of which is given as below:

$$\begin{cases} y_{k+1} = y_k + hv_{k+1} \\ v_{k+1} = v_k - h\nabla^2 f(x^\star)y_k. \end{cases} \tag{8.14}$$

Remark 8.6 The local linearized analysis is based on the stability theorem in finite dimension, the invariant stable manifold theorem, and Hartman–Grobman linearized map theorem [Har82]. The thought is firstly used in [Pol64] to estimate the local convergence of momentum method. And in the paper [LSJR16], the thought is used to exclude the possibility of convergence to saddle point. However, the two theorems above belong to the qualitative theorem of ODE. Hence, the linearized scheme (8.14) is only an approximate estimate for the original scheme (8.13) locally.

8.3.1 Some Lemmas for the Linearized Scheme

Let A be the positive-semidefinite and symmetric matrix to represent $\nabla^2 f(x^\star)$ in (8.14).

Lemma 8.1 *The numerical scheme shown as below*

$$\begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} I - h^2 A & hI \\ -hA & I \end{pmatrix} \begin{pmatrix} x_k \\ v_k \end{pmatrix} \tag{8.15}$$

is equivalent to the linearized symplectic Euler scheme (8.14), where we note that the linear transformation is

$$M = \begin{pmatrix} I - h^2 A & hI \\ -hA & I \end{pmatrix}. \tag{8.16}$$

Proof

$$\begin{pmatrix} I & -hI \\ 0 & I \end{pmatrix} \begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} I & 0 \\ -hA & I \end{pmatrix} \begin{pmatrix} x_k \\ v_k \end{pmatrix} \Leftrightarrow \begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} I - h^2 A & hI \\ -hA & I \end{pmatrix} \begin{pmatrix} x_k \\ v_k \end{pmatrix}$$

□

Lemma 8.2 *For every $2n \times 2n$ matrix M in (8.16), there exists the orthogonal transformation $U_{2n\times 2n}$ such that the matrix M is similar as below:*

$$U^T M U = \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_n, \end{pmatrix} \tag{8.17}$$

where T_i $(i = 1, \ldots, n)$ is a 2×2 matrix with the form

$$T_i = \begin{pmatrix} 1 - \omega_i^2 h^2 & h \\ -\omega_i^2 h & 1, \end{pmatrix} \tag{8.18}$$

where ω_i^2 is the eigenvalue of the matrix A.

Proof Let Λ be the diagonal matrix with the eigenvalues of the matrix A as below:

$$\Lambda = \begin{pmatrix} \omega_1^2 & & & \\ & \omega_2^2 & & \\ & & \ddots & \\ & & & \omega_n^2 \end{pmatrix}.$$

Since A is positive define and symmetric, there exists orthogonal matrix U_1 such that

$$U_1^T A U_1 = \Lambda.$$

Let Π be the permutation matrix satisfying

$$\Pi_{i,j} = \begin{cases} 1, & j \text{ odd}, \ i = \dfrac{j+1}{2} \\ 1, & j \text{ even}, \ i = n + \dfrac{j}{2} \\ 0, & \text{otherwise}, \end{cases}$$

where i is the row index and j is the column index. Then, let $U = \mathbf{diag}(U_1, U_1)\Pi$, we have by conjugation

$$\begin{aligned} U^T M U &= \Pi^T \begin{pmatrix} U_1^T & \\ & U_1^T \end{pmatrix} \begin{pmatrix} I - h^2 A & hI \\ -hA & I \end{pmatrix} \begin{pmatrix} U_1 & \\ & U_1 \end{pmatrix} \Pi \\ &= \Pi^T \begin{pmatrix} I - h^2 \Lambda & hI \\ -h\Lambda & I \end{pmatrix} \Pi \\ &= \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_n \end{pmatrix}. \end{aligned}$$

□

From Lemma 8.2, we know that Eq. (8.15) can be written as the equivalent form

$$\begin{pmatrix} (U_1^T x)_{k+1,i} \\ (U_1^T v)_{k+1,i} \end{pmatrix} = T_i \begin{pmatrix} (U_1^T x)_{k,i} \\ (U_1^T v)_{k,i} \end{pmatrix} = \begin{pmatrix} 1 - \omega_i^2 h^2 & h \\ -\omega_i^2 h & 1 \end{pmatrix} \begin{pmatrix} (U_1^T x)_{k,i} \\ (U_1^T v)_{k,i} \end{pmatrix}, \tag{8.19}$$

where $i = 1, \ldots, n$.

Lemma 8.3 *For any step size h satisfying $0 < h\omega_i < 2$, the eigenvalues of the matrix T_i are complex with absolute value 1.*

Proof For $i = 1, \ldots, n$, we have

$$|\lambda I - T_i| = 0 \Leftrightarrow \lambda_{1,2} = 1 - \frac{h^2\omega_i^2}{2} \pm h\omega_i\sqrt{1 - \frac{h^2\omega_i^2}{4}}.$$

□

Let θ_i and ϕ_i for $i = 1, \ldots, n$ for the new coordinate variables be as follows:

$$\begin{cases} \cos\theta_i = 1 - \dfrac{h^2\omega_i^2}{2} \\ \sin\theta_i = h\omega_i\sqrt{1 - \dfrac{h^2\omega_i^2}{4}} \end{cases}, \qquad \begin{cases} \cos\phi_i = \dfrac{h\omega_i}{2} \\ \sin\phi_i = \sqrt{1 - \dfrac{h^2\omega_i^2}{4}}. \end{cases} \tag{8.20}$$

In order to make θ_i and ϕ_i located in $\left(0, \frac{\pi}{2}\right)$, we need to shrink to $0 < h\omega_i < \sqrt{2}$.

Lemma 8.4 *With the new coordinate in (8.20) for $0 < h\omega_i < \sqrt{2}$, we have*

$$2\phi_i + \theta_i = \pi \tag{8.21}$$

and

$$\begin{cases} \sin\theta_i = \sin(2\phi_i) = h\omega_i \sin\phi_i \\ \sin(3\phi_i) = -\left(1 - h^2\omega_i^2\right)\sin\phi_i. \end{cases} \tag{8.22}$$

Proof With sum–product identities of trigonometric function, we have

$$\begin{aligned} \sin(\theta_i + \phi_i) &= \sin\theta_i\cos\phi_i + \cos\theta_i\sin\phi_i \\ &= h\omega_i\sqrt{1 - \frac{h^2\omega_i^2}{4}} \cdot \frac{h\omega_i}{2} + \left(1 - \frac{h^2\omega_i^2}{2}\right)\sqrt{1 - \frac{h^2\omega_i^2}{4}} \\ &= \sqrt{1 - \frac{h^2\omega_i^2}{4}} \\ &= \sin\phi_i. \end{aligned}$$

Since $0 < h\omega_i < 2$, we have $\theta_i, \phi_i \in \left(0, \frac{\pi}{2}\right)$, we can obtain that

$$\theta_i + \phi_i = \pi - \phi_i \Leftrightarrow \theta_i = \pi - 2\phi_i$$

and with the coordinate transformation in (8.20), we have

$$\sin\theta_i = h\omega_i \sin\phi_i \Leftrightarrow \sin(2\phi_i) = h\omega_i \sin\phi_i.$$

Next, we use sum–product identities of trigonometric function furthermore

$$\begin{aligned}\sin(\theta_i - \phi_i) &= \sin\theta_i \cos\phi_i - \cos\theta_i \sin\phi_i \\ &= h\omega_i\sqrt{1-\frac{h^2\omega_i^2}{4}}\cdot\frac{h\omega_i}{2} - \left(1-\frac{h^2\omega_i^2}{2}\right)\sqrt{1-\frac{h^2\omega_i^2}{4}} \\ &= \left(h^2\omega_i^2 - 1\right)\sqrt{1-\frac{h^2\omega_i^2}{4}} \\ &= -\left(1-h^2\omega_i^2\right)\sin\phi_i\end{aligned}$$

and with $\theta_i = \pi - 2\phi_i$, we have

$$\sin(3\phi_i) = -\left(1-h^2\omega_i^2\right)\sin\phi_i.$$

□

Lemma 8.5 *With the new coordinate in (8.20), the matrix T_i $(i = 1,\ldots,n)$ in (8.18) can expressed as below:*

$$T_i = \frac{1}{\omega_i\left(e^{-i\phi_i} - e^{i\phi_i}\right)}\begin{pmatrix}1 & 1\\ \omega_i e^{i\phi_i} & \omega_i e^{-i\phi_i}\end{pmatrix}\begin{pmatrix}e^{i\theta_i} & 0\\ 0 & e^{-i\theta_i}\end{pmatrix}\begin{pmatrix}\omega_i e^{-i\phi_i} & -1\\ -\omega_i e^{i\phi_i} & 1\end{pmatrix}. \tag{8.23}$$

Proof For the coordinate transformation in (8.20), we have

$$T_i\begin{pmatrix}1\\ \omega_i e^{i\phi_i}\end{pmatrix} = \begin{pmatrix}1\\ \omega_i e^{i\phi_i}\end{pmatrix}e^{i\theta_i} \quad\text{and}\quad T_i\begin{pmatrix}1\\ \omega_i e^{-i\phi_i}\end{pmatrix} = \begin{pmatrix}1\\ \omega_i e^{-i\phi_i}\end{pmatrix}e^{-i\theta_i}.$$

Hence, (8.23) is proved. □

8.3.2 The Asymptotic Analysis

Theorem 8.1 *Let the initial value x_0 and v_0, after the first k steps without resetting the velocity, the iterative solution (8.14) with the equivalent form (8.19) has the form as below:*

$$\begin{pmatrix}(U_1^T x)_{k,i}\\(U_1^T v)_{k,i}\end{pmatrix} = T_i^k \begin{pmatrix}(U_1^T x)_{0,i}\\(U_1^T v)_{0,i}\end{pmatrix} = \begin{pmatrix}-\frac{\sin(k\theta_i-\phi_i)}{\sin\phi_i} & \frac{\sin(k\theta_i)}{\omega_i\sin\phi_i}\\ -\frac{\omega_i\sin(k\theta_i)}{\sin\phi_i} & \frac{\sin(k\theta_i+\phi_i)}{\sin\phi_i}\end{pmatrix}\begin{pmatrix}(U_1^T x)_{0,i}\\(U_1^T v)_{0,i}\end{pmatrix}. \tag{8.24}$$

Proof With Lemma 8.5 and the coordinate transformation (8.20), we have

$$\begin{aligned}
T_i^k &= \frac{1}{\omega_i\left(e^{-i\phi_i}-e^{i\phi_i}\right)}\begin{pmatrix}1 & 1\\ \omega_i e^{i\phi_i} & \omega_i e^{-i\phi_i}\end{pmatrix}\begin{pmatrix}e^{i\theta_i} & 0\\ 0 & e^{-i\theta_i}\end{pmatrix}^k\begin{pmatrix}\omega_i e^{-i\phi_i} & -1\\ -\omega_i e^{i\phi_i} & 1\end{pmatrix}\\
&= \frac{1}{\omega_i\left(e^{-i\phi_i}-e^{i\phi_i}\right)}\begin{pmatrix}1 & 1\\ \omega_i e^{i\phi_i} & \omega_i e^{-i\phi_i}\end{pmatrix}\begin{pmatrix}\omega e^{i(k\theta_i-\phi_i)} & -e^{ik\theta_i}\\ -\omega e^{-i(k\theta_i-\phi_i)} & e^{-ik\theta_i}\end{pmatrix}\\
&= \begin{pmatrix}-\frac{\sin(k\theta_i-\phi_i)}{\sin\phi_i} & \frac{\sin(k\theta_i)}{\omega_i\sin\phi_i}\\ -\frac{\omega_i\sin(k\theta_i)}{\sin\phi_i} & \frac{\sin(k\theta_i+\phi_i)}{\sin\phi_i}\end{pmatrix}.
\end{aligned}$$

The proof is complete. □

Comparing (8.24) and (8.19), we can obtain that

$$\frac{\sin(k\theta_i-\phi_i)}{\sin\phi_i} = 1 - h^2\omega_i^2.$$

With the initial value $(x_0, 0)^T$, then the initial value for (8.19) is $(U_1^T x_0, 0)$. In order to make sure the numerical solution or the iterative solution owns the same behavior as the analytical solution, we need to set $0 < h\omega_i < 1$.

Remark 8.7 Here, the behavior is similar as the thought in [LSJR16]. The step size $0 < hL < 2$ makes sure the global convergence of gradient method. And the step size $0 < hL < 1$ makes the uniqueness of the trajectory along the gradient method, the thought of which is equivalent of the existence and uniqueness of the solution for ODE. Actually, the step size $0 < hL < 1$ owns the property with the solution of ODE, the continuous-limit version. A global existence of the solution for gradient system is proved in [Per13].

For the good-conditioned eigenvalue of the Hessian $\nabla^2 f(x^\star)$, every method such as gradient method, momentum method, Nesterov's accelerated gradient method, and artificially dissipating energy method has the good convergence rate shown by the experiment. However, for our artificially dissipating energy method, since there are trigonometric functions from (8.24), we cannot propose the rigorous mathematic proof for the convergence rate. If everybody can propose a theoretical proof, it is very beautiful. Here, we propose a theoretical approximation for ill-conditioned case, that is, the direction with small eigenvalue $\lambda(\nabla^2 f(x^\star)) \ll L$.

Assumption 8.1 If the step size $h = \frac{1}{\sqrt{L}}$ for (8.14), for the ill-conditioned eigenvalue $\omega_i \ll \sqrt{L}$, the coordinate variable can be approximated by the analytical solution as

$$\theta_i = h\omega_i, \qquad \text{and} \qquad \phi_i = \frac{\pi}{2}. \tag{8.25}$$

With Assumption 8.1, the iterative solution (8.24) can be rewritten as

$$\begin{pmatrix} (U_1^T x)_{k,i} \\ (U_1^T v)_{k,i} \end{pmatrix} = \begin{pmatrix} \cos(kh\omega_i) & \frac{\sin(kh\omega_i)}{\omega_i} \\ -\omega_i \sin(kh\omega_i) & -\cos(kh\omega_i) \end{pmatrix} \begin{pmatrix} (U_1^T x)_{0,i} \\ (U_1^T v)_{0,i} \end{pmatrix}. \tag{8.26}$$

Theorem 8.2 *For every ill-conditioned eigen-direction, with every initial condition $(x_0, 0)^T$, if Algorithm 1 is implemented at $\|v_{iter}\| \le \|v\|$, then there exists an eigenvalue ω_i^2 such that*

$$k\omega_i h \ge \frac{\pi}{2}.$$

Proof When $\|v_{iter}\| \le \|v\|$, then $\left\|U_1^T v_{iter}\right\| \le \left\|U_1^T v\right\|$. While for the $\left\|U_1^T v\right\|$, we can write in the analytical form

$$\left\|U_1^T v\right\| = \sqrt{\sum_{i=1}^{n} \omega_i^2 (U_1 x_0)_i^2 \sin^2(kh\omega_i)},$$

if there is no $k\omega_i h < \frac{\pi}{2}$, $\left\|U_1^T v\right\|$ increases with k increasing. □

For some i such that $k\omega_i h$ approximating $\frac{\pi}{2}$, we have

$$\begin{aligned} \frac{\left|(U_1^T x)_{k+1,i}\right|}{\left|(U_1^T x)_{k,i}\right|} &= \frac{\cos\left((k+1)h\omega_i\right)}{\cos\left(kh\omega_i\right)} \\ &= e^{\ln\cos((k+1)h\omega_i) - \ln\cos(kh\omega_i)} \\ &= e^{-\tan(\xi)h\omega_i} \end{aligned} \tag{8.27}$$

where $\xi \in (kh\omega_i, (k+1)h\omega_i)$. Hence, with ξ approximating $\frac{\pi}{2}$, $\left|(U_1^T x)_{k,i}\right|$ approximates 0 with the linear convergence, but the coefficient will also decay with the rate $e^{-\tan(\xi)h\omega_i}$ with $\xi \to \frac{\pi}{2}$. With the Laurent expansion for $\tan\xi$ at $\frac{\pi}{2}$, i.e.,

$$\tan\xi = -\frac{1}{\xi - \frac{\pi}{2}} + \frac{1}{3}\left(\xi - \frac{\pi}{2}\right) + \frac{1}{45}\left(\xi - \frac{\pi}{2}\right)^3 + \mathcal{O}\left(\left(\xi - \frac{\pi}{2}\right)^5\right),$$

the coefficient has the approximating formula

$$e^{-\tan(\xi)h\omega_i} \approx e^{\frac{h\omega_i}{\xi - \frac{\pi}{2}}} \le \left(\frac{\pi}{2} - \xi\right)^n,$$

where n is an arbitrary large real number in $\mathbb{R}^+$ for $\xi \to \frac{\pi}{2}$.

8.4 Experimental Demonstration

In this section, we implement the artificially dissipating energy algorithm (Algorithm 1), energy conservation algorithm (Algorithm 2), and the combined algorithm (Algorithm 3) into high-dimensional data for comparison with gradient method, momentum method, and Nesterov's accelerated gradient method (Fig. 8.7).

8.4.1 Strongly Convex Function

Here, we investigate the artificially dissipating energy algorithm (Algorithm 1) for the strongly convex function for comparison with gradient method, momentum method, and Nesterov's accelerated gradient method (strongly convex case) by the quadratic function as below:

$$f(x) = \frac{1}{2}x^T Ax + b^T x, \tag{8.28}$$

where A is symmetric and positive-definite matrix. The two cases are shown as below:

(a) The generate matrix A is 500×500 random positive define matrix with eigenvalue from $1e-6$ to 1 with one defined eigenvalue $1e-6$. The generate vector b follows i.i.d. Gaussian distribution with mean 0 and variance 1.
(b) The generate matrix A is the notorious example in Nesterov's book [Nes13], i.e.,

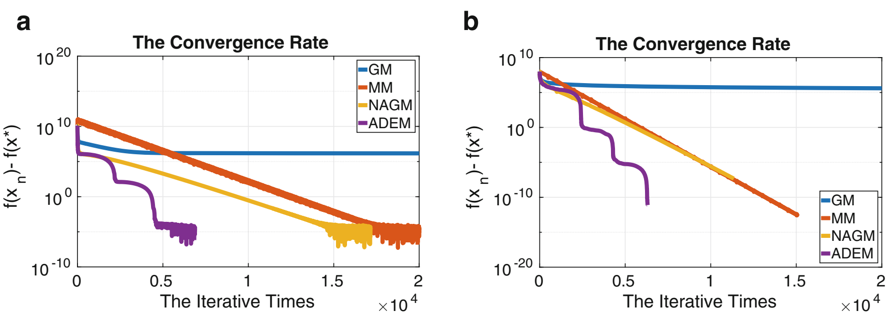


Fig. 8.7 Left: the case (**a**) with the initial point $x_0 = 0$. Right: the case (**b**) with the initial point $x_0 = 1000$

$$A = \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & -1 \\ & & & & -1 & 2 \end{pmatrix},$$

the eigenvalues of the matrix are

$$\lambda_k = 2 - 2\cos\left(\frac{k\pi}{n+1}\right) = 4\sin^2\left(\frac{k\pi}{2(n+1)}\right),$$

and n is the dimension of the matrix A. The eigenvector can be solved by the second Chebyshev's polynomial. We implement $\dim(A) = 1000$ and b is zero vector. Hence, the smallest eigenvalue is approximating

$$\lambda_1 = 4\sin^2\left(\frac{\pi}{2(n+1)}\right) \approx \frac{\pi^2}{1001^2} \approx 10^{-5}.$$

8.4.2 Non-Strongly Convex Function

Here, we investigate the artificially dissipating energy algorithm (Algorithm 1) for the non-strongly convex function for comparison with gradient method, Nesterov's accelerated gradient method (non-strongly convex case) by the log-sum-exp function as below:

$$f(x) = \rho \log\left[\sum_{i=1}^{n} \exp\left(\frac{\langle a_i, x\rangle - b_i}{\rho}\right)\right], \tag{8.29}$$

where A is the $m \times n$ matrix with a_i, $(i = 1, \ldots, m)$ and the column vector of A and b is the $n \times 1$ vector with component b_i. ρ is the parameter. We show the experiment in (8.29): the matrix $A = \left(a_{ij}\right)_{m\times n}$ and the vector $b = (b_i)_{n\times 1}$ are set by the entry following i.i.d Gaussian distribution for the parameter $\rho = 5$ and $\rho = 10$ (Fig. 8.8).

8.4.3 Non-Convex Function

For the non-convex function, we exploit classical test function, known as artificial landscape, to evaluate characteristics of optimization algorithms from general performance and precision. In this paper, we show our algorithms implementing on

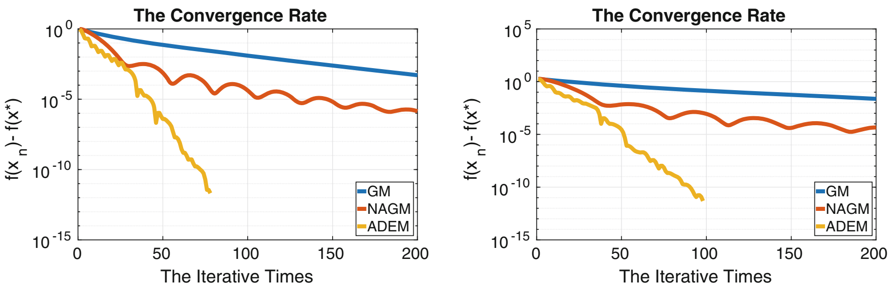


Fig. 8.8 The convergence rate is shown from the initial point $x_0 = 0$. Left: $\rho = 5$; Right: $\rho = 10$

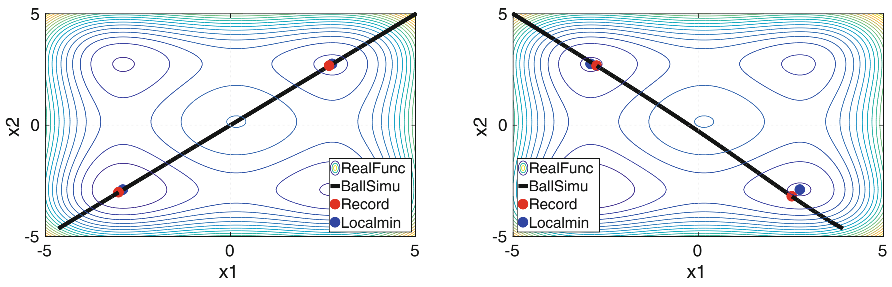


Fig. 8.9 Detecting the number of the local minima of 2-D Styblinski–Tang function by Algorithm 3 with step length $h = 0.01$. The red points are recorded by Algorithm 2 and the blue points are the local minima by Algorithm 1. Left: The initial position $(5, 5)$; Right: The initial position $(-5, 5)$

the Styblinski–Tang function and Shekel function, which is recorded in the virtual library of simulation experiments.[1] Firstly, we investigate Styblinski–Tang function, i.e.,

$$f(x) = \frac{1}{2}\sum_{i=1}^{d}\left(x_i^4 - 16x_i^2 + 5x_i\right), \tag{8.30}$$

to demonstrate the general performance of Algorithm 2 to track the number of local minima and then find the local minima by Algorithm 3 (Fig. 8.9).

To the essential 1-D non-convex Styblinski–Tang function of high dimension, we implement Algorithm 3 to obtain the precision of the global minima as below.

The global minima calculated at the position $(-2.9035, -2.9035, \ldots)$ is -391.6617 as shown in Table 8.1. And the real global minima at $(-2.903534, -2.903534, \ldots)$ is $-39.16599 \times 10 = -391.6599$.

[1] https://www.sfu.ca/~ssurjano/index.html.

Table 8.1 The example for ten-dimensional Styblinski–Tang function from two initial positions

	Local_min1	Local_min2	Local_min3	Local_min4
Initial position	(5, 5, ...)	(5, 5, ...)	(5, −5, ...)	(5, −5, ...)
Position	(2.7486, 2.7486, ...)	(−2.9035, −2.9035, ...)	(2.7486, −2.9035, ...)	(−2.9035, 2.7486, ...)
Function value	−250.2945	−391.6617	−320.9781	−320.9781

Furthermore, we demonstrate the numerical experiment from Styblinski–Tang function to more complex Shekel function

$$f(x) = -\sum_{i=1}^{m}\left(\sum_{j=1}^{4}\left(x_j - C_{ji}\right)^2 + \beta_i\right)^{-1}, \tag{8.31}$$

where

$$\beta = \frac{1}{10}(1, 2, 2, 4, 4, 6, 3, 7, 5, 5)^T$$

and

$$C = \begin{pmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{pmatrix}.$$

(1) Case $m = 5$, the global minima at $x^\star = (4, 4, 4, 4)$ is $f(x^\star) = -10.1532$.

(a) From the position (10, 10, 10, 10), the experimental result with the step length $h = 0.01$ and the iterative times 3000 is shown as below:
Detect Position (Algorithm 2)

$$\begin{pmatrix} 7.9879 & 6.0136 & 3.8525 & 6.2914 & 2.7818 \\ 7.9958 & 5.9553 & 3.9196 & 6.2432 & 6.7434 \\ 7.9879 & 6.0136 & 3.8525 & 6.2914 & 2.7818 \\ 7.9958 & 5.9553 & 3.9196 & 6.2432 & 6.7434 \end{pmatrix}$$

Detect value

$$\begin{pmatrix} -5.0932 & -2.6551 & -6.5387 & -1.6356 & -1.7262 \end{pmatrix}$$

Final position (Algorithm 1)

$$\begin{pmatrix} 7.9996 & 5.9987 & 4.0000 & 5.9987 & 3.0018 \\ 7.9996 & 6.0003 & 4.0001 & 6.0003 & 6.9983 \\ 7.9996 & 5.9987 & 4.0000 & 5.9987 & 3.0018 \\ 7.9996 & 6.0003 & 4.0001 & 6.0003 & 6.9983 \end{pmatrix}$$

Final value

$$\begin{pmatrix} -5.1008 & -2.6829 & -10.1532 & -2.6829 & -2.6305 \end{pmatrix}$$

(b) From the position (3, 3, 3, 3), the experimental result with the step length $h = 0.01$ and the iterative times 1000 is shown as below:
Detect Position (Algorithm 2)

$$\begin{pmatrix} 3.9957 & 6.0140 \\ 4.0052 & 6.0068 \\ 3.9957 & 6.0140 \\ 4.0052 & 6.0068 \end{pmatrix}$$

Detect value

$$\begin{pmatrix} -10.1443 & -2.6794 \end{pmatrix}$$

Final position (Algorithm 1)

$$\begin{pmatrix} 4.0000 & 5.9987 \\ 4.0001 & 6.0003 \\ 4.0000 & 5.9987 \\ 4.0001 & 6.0003 \end{pmatrix}$$

Final value

$$\begin{pmatrix} -10.1532 & -2.6829 \end{pmatrix}$$

(2) Case $m = 7$, the global minima at $x^\star = (4, 4, 4, 4)$ is $f(x^\star) = -10.4029$.

(a) From the position (10, 10, 10, 10), the experimental result with the step length $h = 0.01$ and the iterative times 3000 is shown as below:
Detect Position (Algorithm 2)

$$\begin{pmatrix} 7.9879 & 6.0372 & 3.1798 & 5.0430 & 6.2216 & 2.6956 \\ 8.0041 & 5.9065 & 3.8330 & 2.8743 & 6.2453 & 6.6837 \\ 7.9879 & 6.0372 & 3.1798 & 5.0430 & 6.2216 & 2.6956 \\ 8.0041 & 5.9065 & 3.8330 & 2.8743 & 6.2453 & 6.6837 \end{pmatrix}$$

Detect value

$$\begin{pmatrix} -5.1211 & -2.6312 & -0.9428 & -3.3093 & -1.8597 & -1.5108 \end{pmatrix}$$

Final position (Algorithm 1)

$$\begin{pmatrix} 7.9995 & 5.9981 & 4.0006 & 4.9945 & 5.9981 & 3.0006 \\ 7.9996 & 5.9993 & 3.9996 & 3.0064 & 5.9993 & 7.0008 \\ 7.9995 & 5.9981 & 4.0006 & 4.9945 & 5.9981 & 3.0006 \\ 7.9996 & 5.9993 & 3.9996 & 3.0064 & 5.9993 & 7.0008 \end{pmatrix}$$

Final value

$$\begin{pmatrix} -5.1288 & -2.7519 & -10.4029 & -3.7031 & -2.7519 & -2.7496 \end{pmatrix}$$

(b) From the position (3, 3, 3, 3), the experimental result with the step length $h = 0.01$ and the iterative times 1000 is shown as below:
Detect Position (Algorithm 2)

$$\begin{pmatrix} 4.0593 & 3.0228 \\ 3.9976 & 7.1782 \\ 4.0593 & 3.0228 \\ 3.9976 & 7.1782 \end{pmatrix}$$

Detect value

$$\begin{pmatrix} -9.7595 & -2.4073 \end{pmatrix}$$

Final position (Algorithm 1)

$$\begin{pmatrix} 4.0006 & 3.0006 \\ 3.9996 & 7.0008 \\ 4.0006 & 3.0006 \\ 3.9996 & 7.0008 \end{pmatrix}$$

Final value

$$\begin{pmatrix} -10.4029 & -2.7496 \end{pmatrix}$$

(3) Case $m = 10$, the global minima at $x^\star = (4, 4, 4, 4)$ is $f(x^\star) = -10.5364$.

(a) From the position (10, 10, 10, 10), the experimental result with the step length $h = 0.01$ and the iterative times 3000 is shown as below:
Detect Position (Algorithm 2)

$$\begin{pmatrix} 7.9977 & 5.9827 & 4.0225 & 2.7268 & 6.1849 & 6.2831 & 6.3929 \\ 7.9942 & 6.0007 & 3.8676 & 7.3588 & 6.0601 & 3.2421 & 1.9394 \\ 7.9977 & 5.9827 & 4.0225 & 2.7268 & 6.1849 & 6.2831 & 6.3929 \\ 7.9942 & 6.0007 & 3.8676 & 7.3588 & 6.0601 & 3.2421 & 1.9394 \end{pmatrix}$$

Detect value

$$\begin{pmatrix} -5.1741 & -2.8676 & -7.9230 & -1.5442 & -2.4650 & -1.3703 & -1.7895 \end{pmatrix}$$

Final position (Algorithm 1)

$$\begin{pmatrix} 7.9995 & 5.9990 & 4.0007 & 3.0009 & 5.9990 & 6.8999 & 5.9919 \\ 7.9994 & 5.9965 & 3.9995 & 7.0004 & 5.9965 & 3.4916 & 2.0224 \\ 7.9995 & 5.9990 & 4.0007 & 3.0009 & 5.9990 & 6.8999 & 5.9919 \\ 7.9994 & 5.9965 & 3.9995 & 7.0004 & 5.9965 & 3.4916 & 2.0224 \end{pmatrix}$$

Final value

$$\begin{pmatrix} -5.1756 & -2.8712 & -10.5364 & -2.7903 & -2.8712 & -2.3697 & -2.6085 \end{pmatrix}$$

(b) From the position (3, 3, 3, 3), the experimental result with the step length $h = 0.01$ and the iterative times 1000 is shown as below:
Detect Position (Algorithm 2)

$$\begin{pmatrix} 4.0812 & 3.0206 \\ 3.9794 & 7.0173 \\ 4.0812 & 3.0206 \\ 3.9794 & 7.0173 \end{pmatrix}$$

Detect value

$$\begin{pmatrix} -9.3348 & -2.7819 \end{pmatrix}$$

Final position (Algorithm 1)

$$\begin{pmatrix} 4.0007 & 3.0009 \\ 3.9995 & 7.0004 \\ 4.0007 & 3.0009 \\ 3.9995 & 7.0004 \end{pmatrix}$$

Final value

$$\begin{pmatrix} -10.5364 & -2.7903 \end{pmatrix}$$

8.5 Conclusion and Further Works

Based on the view for understanding arithmetical complexity from analytical complexity in the seminal book [Nes13] and the idea for viewing optimization from differential equation in the novel blog,[2] we propose some original algorithms based on Newton's second law with the kinetic energy observable and controllable in the computational process firstly. Although our algorithm cannot fully solve the global optimization problem, or it is dependent on the trajectory path, this work introduces the Hamilton system essential to optimization such that it is possible that the global minima can be obtained. Our algorithms are easy to implement and own a more rapid convergence rate.

For the theoretical view, the Hamilton system is closer to nature and a lot of fundamental work have appeared in the previous century, such as KAM theory, Nekhoroshev estimate, operator spectral theory, and so on. Are these beautiful and essentially original work used to understand and improve the algorithm for optimization and machine learning? Also, in establishing the convergence rate, the matrix containing the trigonometric function can be hard to estimate. Researchers have proposed some methods for estimating the trigonometric matrix based on spectral theory. For the numerical scheme, we only exploit the simple first-order symplectic Euler method. Several more efficient schemes, such as Störmer–Verlet scheme, symplectic Runge–Kutta scheme, order condition method, and so on, are proposed in [Nes13].

These schemes can make the algorithms in this paper more efficient and accurate. For the optimization, the method we proposed is only about unconstrained problem. In the nature, the classical Newton's second law, or the equivalent expression—Lagrange mechanics and Hamilton mechanics, is implemented on the manifold in the almost real physical world. In other words, a natural generalization is from unconstrained problem to constrained problem for our proposed algorithms. A more natural implementation is the geodesic descent in [LY$^+$84]. Similar to the development of the gradient method from smooth condition to nonsmooth condition, our algorithms can be generalized to nonsmooth condition by the subgradient. For application, we will implement our algorithms to non-negative matrix factorization, matrix completion, and deep neural network and speed up the training of the objective function. Meanwhile, we apply the algorithms proposed in this paper to the maximum likelihood estimator and maximum a posteriori estimator in statistics.

Starting from Newton's second law, we implement only a simple particle in classical mechanics, or macroscopic world. A natural generalization is from the macroscopic world to the microscopic world. In the field of fluid dynamics, the Newton's second law is expressed by Euler equation, or more complex Navier–Stokes equation. An important topic from fluid dynamics is geophysical fluid dynamics, containing atmospheric science and oceanography. Especially, a key

[2] http://www.offconvex.org/2015/12/11/mission-statement/.

feature in the oceanography different from atmospheric science is the topography, which influences mainly vector field of the fluid. So many results have been demonstrated based on many numerical modeling, such as the classical POM,[3] HYCOM,[4] ROMS,[5] and FVCOM.[6] A reverse idea is that if we view the potential function in black box is the topography, we observe the changing of the fluid vector field to find the number of local minima in order to obtain the global minima with a suitable initial vector field. A more adventurous idea is to generalize the classical particle to the quantum particle. For quantum particle, the Newton's second law is expressed by the energy form that is from the view of Hamilton mechanics, which is the starting point for the proposed algorithm in this paper. The particle appears in the wave form in a microscopic world. When the wave meets the potential barrier, the tunneling phenomena will appear. The tunneling phenomena will also appear in higher dimensions. It is very easy to observe the tunneling phenomena in the physical world. However, if we attempt to compute this phenomena, the problem becomes NP-hard. Only if quantum computing is used is the phenomena very easy to simulate, as we can find the global minima by binary section search. That is, if there exist tunneling phenomena in the upper level, the algorithm will continue to detect this in the upper level, otherwise go to the lower level. In the quantum world, it needs only $\mathcal{O}(\log n)$ times to find the global minima rather than becoming NP-hard.

[3] http://ofs.dmcr.go.th/thailand/model.html.

[4] https://hycom.org/.

[5] https://www.myroms.org/.

[6] http://fvcom.smast.umassd.edu/.