The advantage of symplectic Euler in optimization and its application

Shibo Zhao

Computational Mathematics Department, University of Science and Technology of China, Hefei, Anhui, 230026, China

Zhao2001@mail.ustc.edu.cn

Abstract. Many professions today place a high value on optimization, and many problems can eventually be transformed into optimization issues. There are many iterative methods available today to handle optimization issues, however many algorithms' design principles are unclear. Weijie Su solved this problem by discretizing the iterative equation using an ordinary differential equation, but different discretization techniques will provide different outcomes. So choosing an appropriate method is important. Three discretization techniques—explicit Euler, implicit Euler, and symplectic Euler—are compared in this work. It is found that while both symplectic and implicit Euler can accelerate the process, only symplectic Euler can be put to use in practice. This further demonstrates symplectic Euler's supremacy in iteration. The use of symplectic Euler in other fields is also introduced in this study, particularly in the Lotka-Volterra equation where promising results might be attained. Symplectic Euler is critical to optimization and is likely to be applied in more areas in the future.

Keywords: symplectic Euler, NAG-SC, Lotka-Volterra equation.

1. Introduction

With the development of science and technology, people find that more and more crucial problems in every field finally transform into optimization problems. Nowadays, the optimization method has become a necessary research tool for engineers and technicians.

The most basic optimization problem is shown below:

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

With the development of optimization, many algorithms have been created. For the first-order method, the most basic algorithm is gradient descent (GD). Then Polyak created the heavy-ball method by using historical accumulated momentum instead of the current gradient to reduce the oscillation. The next critical development is Nesterov's accelerated method (NAG). It means that when calculating the current gradient, follow the historical accumulated momentum one step ahead and then see what to do next. NAG can also be written as NAG-SC for a strongly convex function with L-Lipschitz gradients. However, the design principle of some methods is not clear, for example, the NAG method. People have tried to solve this problem in many ways, and in recent years, Weijie Su proved that when s→0, NAG-C approaches the ODE [1]. Then he used this consequence to prove that the convergence rate of NAG-C is correct by calculating the Lyapunov function [1]. This means that ODE can be used to explain the algorithm and calculate the convergence rate. However, the translation between ODEs and discrete algorithms is not clear. A notable algorithm is geometric numerical integration, but it still can't prove the existence of accelerated rates in discrete time [2]. Researchers find that different discrete forms can lead to different acceleration rates, so choosing an appropriate discrete form is important in this method [3]. To achieve discretization and acceleration, Euler's method is basic and common. Euler's method, which is invented by Leonhard Euler is one of the oldest and simplest numerical solution methods in the world. The main idea of Euler's method is to use tangents instead of the solution function. There is explicit Euler (forward Euler scheme), implicit Euler (backward Euler method), and symplectic Euler, respectively.

In this paper, NAG-SC is used as an example to prove that both the symplectic Euler method and the implicit Euler method can accelerate high-resolution differential equations. While generally, implicit Euler is difficult to use in practice, symplectic Euler is the best Euler scheme among those three on this issue. Moreover, symplectic Euler has plenty of applications in practice, so it is critical in optimization.

2. Symplectic Euler for NAG-SC

2.1. Euler method

NAG-SC is a useful algorithm in optimization. In this section, like what Weijie Su did, the iterative equation is converted to ODE at first, and then the ODE is discretized in three ways to see what happens.

The Euler method is one of the most basic iteration methods. Suppose s is the step size and v_k is the velocity variable.

2.1.1. Explicit Euler scheme. The simplest single-step method is the explicit Euler scheme. It is a first-order scheme with poor precision, but its formula is simple.

$$x_{k+1}-x_k=\sqrt{s}v_k, \qquad \sqrt{s}\nabla^2 f(x_k)v_k \approx \nabla f(x_{k+1})-\nabla f(x_k) \tag{2}$$

The geometric significance of explicit Euler is quite obvious: it uses a broken line passing through a known point to approximately replace the original curve passing through this point. Therefore, this method is also called the broken line method [4].

2.1.2. Implicit Euler scheme. Implicit Euler is quite similar to Explicit Euler. It just replaces the v_k with v_{k+1} in the equation (2).

$$x_{k+1}-x_k=\sqrt{s}v_{k+1}, \qquad \sqrt{s}\nabla^2 f(x_{k+1})v_{k+1} \approx \nabla f(x_{k+1})-\nabla f(x_k) \tag{3}$$

There is a big difference from explicit Euler that needs to be noticed. In the explicit Euler method, x_{k+1} can be clearly expressed in terms of x_k and v_k, but in the implicit Euler method, when computing x_{k+1}, it has to solve the implicit equation which might be difficult to calculate [4].

2.1.3 Symplectic Euler scheme. Because it combines the traits of earlier schemes, symplectic Euler is also known as semi-implicit Euler.

$$x_{k+1}-x_k=\sqrt{s}v_k, \qquad \sqrt{s}\nabla^2 f(x_{k+1})v_k \approx \nabla f(x_{k+1})-\nabla f(x_k) \tag{4}$$

It is easy to find that the first equation of this scheme is similar to explicit Euler while the second one is similar to implicit Euler. Like the explicit Euler method, x_{k+1} can be also clearly expressed in terms of x_k and v_k, so it is easy to column x_{k+1} in this iteration.

2.2. The ODE for NAG-SC

The following two iteration equations are NAG-SC:

$$y_{k+1}=x_k-s\nabla f(x_k)\ , \qquad x_{k+1}=y_{k+1}+\frac{1-\sqrt{\mu s}}{1+\sqrt{\mu s}}(y_{k+1}-y_k) \tag{5}$$

NAG-SC can also be expressed in the univariate form by plugging and multiplying $\frac{1+\sqrt{\mu s}}{1-\sqrt{\mu s}}$ on both sides:

$$\frac{x_{k+1}+x_{k-1}-2x_k}{s}+\frac{2\sqrt{\mu s}}{1-\sqrt{\mu s}}*\frac{x_{k+1}-x_k}{s}+\nabla f(x_k)-\nabla f(x_{k-1})+\frac{1+\sqrt{\mu s}}{1-\sqrt{\mu s}}\nabla f(x_k)=0 \tag{6}$$

The definition of the velocity variable $v_k=\frac{x_{k+1}-x_k}{\sqrt{s}}$. Plug v_k in equation (2.5), so NAG-SC can be rewritten as:

$$\begin{cases} x_k-x_{k-1}=\sqrt{s}v_{k-1} \\ v_k-v_{k-1}=-\frac{2\sqrt{\mu s}}{1-\sqrt{\mu s}}*v_k-\sqrt{s}(\nabla f(x_k)-\nabla f(x_{k-1}))-\frac{1+\sqrt{\mu s}}{1-\sqrt{\mu s}}*\sqrt{s}\nabla f\ \ (x_k) \end{cases} \tag{7}$$

This is the other form of iteration equation.

Then, assume $x_k=X(t_k)$ for an equation X(t) and then suppose $t_k=k\sqrt{s}$. Using the Taylor expansion of $\sqrt{s}$ and gradient on equation (2.5), the following equation can be proved:

$$\frac{\ddot{X}(t_k)}{1-\sqrt{\mu s}}+\frac{2\sqrt{\mu}}{1-\sqrt{\mu s}}\dot{X}(t_k)+\nabla^2 f\big(X(t_k)\big)\dot{X}(t_k)\sqrt{s}+\frac{1+\sqrt{\mu s}}{1-\sqrt{\mu s}}*\nabla f\big(X(t_k)\big)+o(s)=0 \tag{8}$$

Eliminate o(s) terms but maintain the $o(\sqrt{s})$ terms of (2.7), so $\left(1-\sqrt{\mu s}\right)*\sqrt{s}=\sqrt{s}$. Therefore (2.7) can be transformed into the equation below:

$$\ddot{X}+2\sqrt{\mu}\dot{X}+\sqrt{s}*\nabla^2 f(X)\dot{X}+\left(1+\sqrt{\mu s}\right)*\nabla f(X)=0 \tag{9}$$

And this is the ODE of NAG-SC [5].

2.3. *Different ways to discretize the ODE*

Suppose $V=\dot{X}$, then it is easy to find that the third term of (2.8) and the right side of the second equation of (2.1)-(2.3) are the same. So the discretization forms of different Euler scheme is:

$$\text{(Symplectic)}\begin{cases} x_k-x_{k-1}=\sqrt{s}v_{k-1} \\ v_k-v_{k-1}=-2\sqrt{\mu s}v_k-\sqrt{s}(\nabla f(x_k)-\nabla f(x_{k-1}))-\sqrt{s}(1+\sqrt{\mu s})*\nabla f(x_k) \end{cases} \tag{10}$$

$$\text{(Explicit)}\begin{cases} x_k-x_{k-1}=\sqrt{s}v_{k-1} \\ v_k-v_{k-1}=-2\sqrt{\mu s}v_{k-1}-\sqrt{s}(\nabla f(x_k)-\nabla f(x_{k-1}))-\sqrt{s}(1+\sqrt{\mu s})*\nabla f(x_{k-1}) \end{cases} \tag{11}$$

$$\text{(Implicit)}\begin{cases} x_k-x_{k-1}=\sqrt{s}v_k \\ v_k-v_{k-1}=-2\sqrt{\mu s}v_k-\sqrt{s}(\nabla f(x_k)-\nabla f(x_{k-1}))-\sqrt{s}(1+\sqrt{\mu s})\nabla f(x_k) \end{cases} \tag{12}$$

Comparing those three Euler schemes (2.9)- (2.11) with the iteration form of NAG-SC (2.6), The closest scheme to (2.6) is symplectic Euler. To be more specific, NAG-SC only deviates from the symplectic scheme in the second line of the (2.6) by a factor of $\frac{1}{1-\sqrt{\mu s}}$. When the step size s is close to 0, $\frac{1}{1-\sqrt{\mu s}}\approx 1$. So NAG-SC is roughly a symplectic method [6]. But for explicit Euler and implicit Euler, some subscripts are different from (2.6), so there's a big difference in iteration. All in all, symplectic Euler is the best one for NAG-SC, and this discovery also highlights the superiority of symplectic Euler in optimization.

2.4. *Some example*

For any f(x) that can use NAG-SC to iterate, the following assessment is correct [6]:

The following symplectic Euler scheme of (2.8) is satisfied with step size $s=\frac{4}{9L}$:

$$f(x_k)-f(x^*)\leq 5L\|x_0-x^*\|^2 * \left(1+\frac{1}{9}\sqrt{\frac{\mu}{L}}\right)^{-k} \tag{13}$$

The following explicit Euler scheme of (2.8) is satisfied with step size $s=\frac{\mu}{100L^2}$:

$$f(x_k)-f\big(x^*\big)\leq 3L\|x_0-x^*\|^2*\left(1-\frac{\mu}{80L}\right)^{k} \tag{14}$$

The following implicit Euler scheme of (2.8) is satisfied with $s=\frac{1}{L}$:

$$f(x_k)-f(x^*)\leq 13\|x_0-x^*\|^2*4\left(1+\frac{1}{4}\sqrt{\frac{\mu}{L}}\right)^{-k} \tag{15}$$

When $s=\frac{1}{L}$, the convergence rate of NAG-SC has been proved by Nesterov to be $O((1-\sqrt{\mu L})^k)$ [7]. So the application demonstrates that both the implicit and symplectic schemes can accelerate. The implicit approach, however, is difficult to put into practice. Because when the objective is not quadratic, it necessitates the solution of a nonlinear problem. As a result, symplectic Euler is the best of the three Euler schemes.

Moreover, researchers have demonstrated that the optimization approach which created by using symplectic Euler on a high-resolution ODE minimizes convex functions at an accelerated rate [6]. So symplectic Euler is important in optimization.

3. Application of symplectic Euler

Symplectic Euler is commonly used in the Lotka-Volterra equation. This interspecific competition equation, which is also called the predator-prey equation, had a great influence on the development of modern ecological theory. Suppose there are some sheep and wolves on the grassland, and there are three basic principles: sheep give birth to sheep, wolves give birth to wolves and wolves eat sheep. The population of wolves x(t) and the population of sheep y(t) over time can be given by the following Lotka-Volterra equation:

$$\begin{cases}\frac{dx}{dt}=-\alpha x+\beta xy\\ \frac{dy}{dt}=\gamma y-\delta xy\end{cases} \tag{16}$$

The first component of the equation's right side means the natural growth rate, and the second one represents the predatory relationship between predator and prey.

To simplify the problem, suppose $\beta=\delta=\gamma=1$, $\alpha=2$. So it is easy to find out that when x=1 and y=2, the equation is satisfied. Let's solve this problem by using three Euler schemes, so there are:

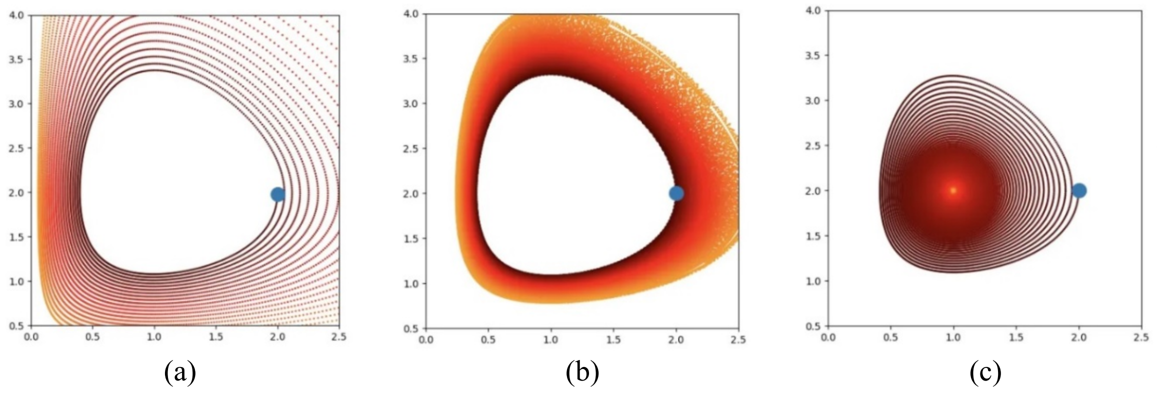

Figure 1. Iteration results.

Each figure has an x-axis for the predator population, a y-axis for the prey population, and a point for the initial value.

In Figure 1, (a) shows the iteration results of explicit Euler. Over time, the populations of sheep and wolves will become more and more extreme, and finally, they run beyond the limits of the model. So, explicit Euler can't be applied to this problem.

(b) shows the iteration results of implicit Euler. At first, the result was balanced, but this was a sham. Keep iterating until it has enough iterations, and the model eventually breaks down. So, implicit Euler is also not the best choice.

(c) shows the iteration results of symplectic Euler. It is easy to find out that the result gradually converges to a point (1, 2). This result is the same as the previously predicted result, the model is balanced [8]. So symplectic Euler is the best one among the three Euler schemes.

Moreover, symplectic Euler is also useful in many other fields. It can be used, for instance, to solve the biological cell migration equations based on the second Newton's law [9]. It can also be used to develop a mathematical model for ground penetrating radar (GPR) because by doing this GPR can be high computing efficiency and ease of programming [10].

4. Conclusion

This paper mainly analyses the superiority of symplectic Euler. By comparing three common Euler schemes, the result is that implicit Euler and symplectic Euler can accelerate NAG-SC, but only symplectic Euler can be used in practice. This paper also discusses its application in other fields. It can be used to solve a famous equation: the Lotka-Volterra equation, while others cannot. Symplectic Euler also can be used to deal with GPR and cell migration. So, symplectic Euler plays a crucial role in optimization.

This conclusion also inspires researchers that they can try to use symplectic Euler to discrete in optimization. Only three Euler methods are compared in this paper, and more methods can be selected to be compared in the future to get the best results.

References

[1] Weijie S, Stephen B, and Emmanuel C 2016 A differential equation for modeling Nesterov's accelerated gradient method: theory and insights Journal of Machine Learning Research 17.

[2] Ernst H, Christian L, and Gerhard W 2006 Geometric numerical integration: structure-preserving algorithms for ordinary differential equations 31 Springer Science & Business Media.

[3] Wibisono A, Wilson C, and Jordan I 2016 A variational perspective on accelerated methods in optimization Proceedings of the National Academy of Sciences of the United States of America 47 doi:10.1073/PNAS.1614734113.

[4] Xiaohong L 2008 Numerical solutions of ordinary differential equations and their applications.

[5] Bin S, Simon D, Michael J, and Weijie S 2021 Understanding the acceleration phenomenon via high-resolution differential equations Mathematical Programming (pre-publish) doi:10.1007/S10107-021-01681-8.

[6] Bin S, Simon D, Weijie S, and Michael J 2019 Acceleration via symplectic discretization of high-resolution differential equations CoRR.

[7] Yurii N 1983 A method of solving a convex programming problem with convergence rate "O(1/" "k" ^"2" ")" Soviet Mathematics Doklady 27.

[8] T E and B K 2005 Poisson integrators for Volterra lattice equations Applied Numerical Mathematics 6 doi:10.1016/j.apnum.2005.06.009.

[9] Dmitry G and Gennady B 2019 Spatially resolved modeling of immune responses following a multiscale approach: from computational implementation to quantitative predictions Russian Journal of Numerical Analysis and Mathematical Modelling 5 doi:10.1515/rnam-2019-0021.

[10] Jianwei L, Hongyuan F, Yinping L, Juan Z, and Man Y 2020A Parallel conformal symplectic Euler algorithm for GPR numerical simulation on dispersive media Chinese Journal of Geophysics 08 3192-3204.