Experimenting with the convergence of min-max optimizers

Luca Bracone François Charroin Thomas Rimbot

***Abstract*—We make empirical tests regarding the convergence of different min-max optimizers on a variety of loss functions. We consider the $O(s^r)$-resolution differential equations as seen in [2], and show that for a couple of loss functions, they do indeed follow the discrete time algorithm more closely. We also investigate more in details the performances of the Jacobian Method.**

I. Introduction

There has been an increasing interest in deriving ordinary differential equations (ODEs) that mimic a discrete time algorithm. The gist of the idea lies in supposing that the discrete iterates x_k produced by an optimization algorithm come from a continuous curve $X : \mathbb{R} \to \mathbb{R}^n$ which we sample at equal time intervals $x_k = X(tk)$. Using Taylor expansion can make higher order terms appear, and this method is generally known as *high-resolution ODEs*. Once an ODE is obtained we can use well-known theory such as Lyapunov functions to obtain theoretical results and translate them back to the discrete case, this idea first appears in [5].

In this work, we wish to extend this idea to algorithms that find equilibrium points. Indeed, in some machine learning applications, one is interested in computing an equilibrium point of the loss function. That is, given a differentiable function $l : \mathbb{R}^{n_1+n_2} \to \mathbb{R}$ we are looking for points $x^* \in \mathbb{R}^{n_1}, y^* \in \mathbb{R}^{n_2}$ such that an equilibrium is reached.

$$(x^*, y^*) = \text{argmin}_x \text{argmax}_y l(x, y)$$

We study different algorithms taken from [2] and apply them to various losses. The theory of *Gradient Flow* motivates the use of high-resolution differential equations on which we will experiment in section V We look at their convergence behaviours and give some insights on them, as well as possible solutions for optimizing their use.

II. The algorithms

The analogous of Gradient Descent in this setting is *Gradient Descent Ascent* (GDA), in which the relevant variables take a step towards the gradient and the others step away from the gradient. However, as will be seen in the next section, GDA fails to converge in simple settings such as $l(x, y) = xy$. In light of this, new approaches have been proposed. Some of the currently used algorithms are:

- Extra Gradient Method (EGM) [1]:
  $z_{n+1} = z_n - \gamma \nabla l(z_n - \gamma \nabla l(z_n))$
- Proximal Point Method (PPM) [3]:
  $z_{n+1} = z_n - \gamma \nabla l(z_{n+1})$
- Optimistic Gradient Descent Ascent (OGDA) [4]:
  $z_{n+1} = z_n - 2\gamma \nabla l(z_n) + \gamma \nabla l(z_{n-1})$

Moreover, [2] introduces a new algorithm derived from a high-resolution ODE:

- Jacobian Method (JM):
  $z_{n+1} = z_n + \gamma \Delta l(z_n) \nabla l(z_n)$

In this work, we focus on these algorithms, compare them on the basic case $l(x, y) = xy$, develop a further analysis by looking at bilinear forms $l(x, y) = x^T Ay$, and move on to more atypical losses.

III. The two scenarios in [2]

Starting from the aforementioned paper, the first goal of this project was to implement the described algorithms: GDA, EGM, OGDA and JM. The aim was to reproduce the results in the paper, and conduct an experimental analysis of their convergence properties when applied to special losses.

A. The basic case

We first focus on what is referred to as *Scenario (i)* in the paper, namely losses $l(x, y)$ which are strongly convex in x and strongly concave in y. The most basic example of such a loss is the first order function $l(x, y) = xy$. We report in Figure 1 the trajectories of the four algorithms for this problem.

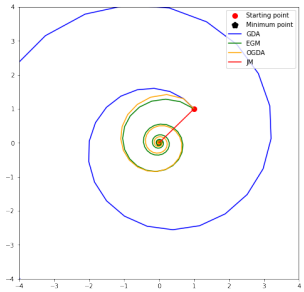

Figure 1: Trajectories of the four algorithms with $l(x, y) = xy$, step-size $s = 0.3$ and initial guess $z_0 = (1, 1)$

This is in fact the exact same picture as the one described in [2] (Figure 1 b) page 5). Most notably, it showcases the different behaviours of the four algorithms.

- GDA spirals outwards without converging.
- EGD and OGDA spiral inwards, converging to the fixed point.
- JM instantaneously goes to the fixed point.

B. Bilinear losses

We now focus on the investigation of losses l which can be represented as a bilinear form (*Scenario (ii)*). More precisely:

$$l(x, y) = x^T Ay, \text{ where } x \in \mathbb{R}^n, y \in \mathbb{R}^m, A \in \mathbb{R}^{n \times m} \quad (1)$$

In the following, we consider the case $n = m = 2$.
Starting from what we did in subsection III-A, we implemented the four algorithms for this type of loss. Contrary to the case above, we now have the freedom to choose the expression of l through the definition of A. We therefore expect to see different convergence behaviours depending on this choice. We report here a few of them.

1) Good convergence: Let us set

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \tag{2}$$

This matrix is positive-definite, with eigenvalues $\lambda_{\pm} = 1 \pm i$. In that case, we get exactly the same convergence behaviours as the ones described in subsection III-A, for both dimensions 0 and 1. The trajectories follow a similar trend as in Figure 1.

It is worth noting that changing the values of the matrix while keeping its structure does not impact the result. For instance, setting

$$A = \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix} \tag{3}$$

with $\lambda_{\pm} = 1 \pm 2i$ yields the same behaviours.

2) Faster convergence in one coordinate: Let us set

$$A = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix} \tag{4}$$

This matrix is once again positive-definite, with eigenvalues $\lambda_{\pm} \approx 1 \pm 1.4i$. In that case, the convergence behaviours follow the same general trend, but we can see (Figure 2) that the evolution is faster in one coordinate with respect to the other.

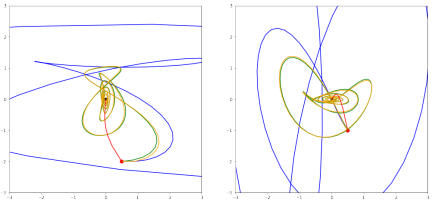

Figure 2: Trajectories of the four algorithms with $l(x, y) = x^T Ay$, with A as in 4, step-size $s = 0.1$, 200 iterations. Dimensions 0 (left) and 1 (right).

This comes from the asymmetry in the absolute values of A, namely the fact that $|A_{12}| = 2 > 1 = |A_{21}|$. Note that convergence is attained nonetheless for the best algorithms.

3) Slow convergence: Let us set

$$A = \begin{bmatrix} 0.1 & 0.2 \\ -0.2 & 0.2 \end{bmatrix} \tag{5}$$

This matrix is also positive-definite, with eigenvalues $\lambda_{\pm} \approx 0.15 \pm 0.2i$. Convergence behaviours are similar, but Figure 3 illustrates the extremely slow evolution of the solution. Note

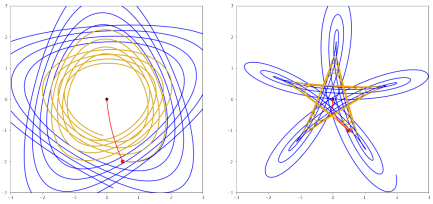

Figure 3: Trajectories of the four algorithms with $l(x, y) = x^T Ay$, with A as in 5, step-size $s = 0.1$, 2500 iterations. Dimensions 0 (left) and 1 (right).

that this is the result of **2500** iterations with the same time step $s = 0.1$.

This comes from the general scale of the matrix. In particular, the modulus of the eigenvalues is way smaller than before, inducing a much slower convergence. As another example, scaling the matrix in Equation 3 by 0.1 gives similar conclusions.

4) Convergence to the other fixed point: Let us set

$$A = \begin{bmatrix} 1 & -1 \\ -0.5 & 0.5 \end{bmatrix} \tag{6}$$

Contrary to the other cases, this matrix is only semi-positive-definite, with eigenvalues $\lambda \in \{1.5; 0\}$. As we can see on Figure 4, the algorithms tend to push the solution to the other fixed point of l, implying that the origin is not a stable fixed point.

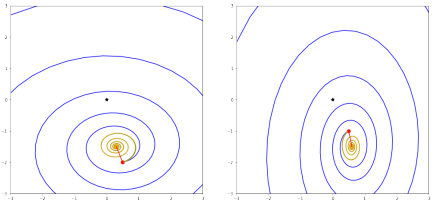

Figure 4: Trajectories of the four algorithms with $l(x, y) = x^T Ay$, with A as in 6, step-size $s = 0.1$, 200 iterations. Dimensions 0 (left) and 1 (right).

This comes from the fact that the matrix A is non-invertible, with one eigenvalue being 0. The fixed point is therefore not unique anymore, which impedes the algorithms from efficiently converging to it.

5) Discussion of the results: As we saw in the previous sections, the choice of A in the case of a bilinear loss $x^T Ay$ is fundamental to describe the convergence behaviours of the algorithms for the min-max problem. Most notably, the eigenvalues of A play a big role. If their real part is positive, convergence seems guaranteed, but the scale of the matrix influences its speed. This suggests a tuning of the step size to account for the scale of the modulus of the eigenvalues. Moreover, the general structure of A has its importance as well. If it is too "unbalanced", convergence is faster in one coordinate with respect to the other. An adaptive step-size, different for each coordinate could solve this problem and

optimize convergence in a smoother way. Finally, if the matrix is such that there exists another fixed point of l, convergence to the actual minimum is not guaranteed, and we might end up in a different solution to the expected one.

IV. Convex-concave losses

In [2], the authors show an example of a convex-concave function with which JM converges quickly. The goal of this section is to evaluate if this algorithm converges for a wider variety of convex-concave functions. First, we replicate the function presented on the paper to assess if the method used is working. The function is the following one:

$$f(x, y) = 0.5x + 2xy - 0.5y \tag{7}$$

When comparing the four algorithms for two different step sizes, we obtain the trajectories in Figure 5.

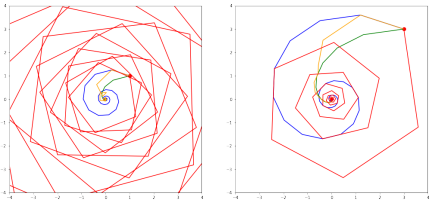

Figure 5: Trajectories of the four algorithms with $l(x, y) = 0.5x^2 + 2xy - 0.5y^2$, with step-sizes $s = 0.25$ (left) and $s = 0.2$ (right) 100 iterations.

The choice of the step-size is really important for the Jacobian Method as it converges with a step size of 0.2 but not for 0.25. Now let us look at variations of the first function:

$$f_1(x, y) = 0.5x^2 + xy - 0.5y^2 \tag{8}$$

$$f_2(x, y) = x^2 - y^2 \tag{9}$$

With these two functions, JM does not converge even when changing the step size, while all the other methods do.

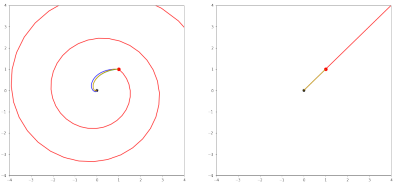

Figure 6: Trajectories of the four algorithms with $l(x, y) = f_1(x, y)$ (left) and $l(x, y) = f_2(x, y)$ (right).

To summarize, while slightly changing the function does not affect the other three algorithms, JM does not converge in a lot of cases. Now, let us try with more atypical functions:

$$f_3(x, y) = exp(x) + exp(-x) - exp(y) - exp(-y) \tag{10}$$

$$f_4(x, y) = cos(x) - cos(y) \tag{11}$$

With these two functions, the results are about the same. In the case of the periodic function, JM converges, but it might not be to the same saddle point that the other methods converge to, depending on the initial coordinates.

To conclude this part, contrary to the apparent claim in [2], JM is not really efficient at optimizing min-max problems and attaining the saddle point when compared to GDA, EGM and OGDA. Furthermore, this method seems really sensitive to the step size and the initial coordinates. A more in-depth tuning of these parameters should be conducted in order to optimize convergence.

V. High-resolution ODE

In this section we try out the differential equations given in [2], in particular the ones for EGM. We illustrate the results in Figure 8. The idea is to create differential equations that follow the discrete time algorithm (DTA) with increasing precision. We used a numerical ODE solver to give a trajectory (see Figure 7) which we sampled at regular intervals, then we measured the l_2 distance between the sampled points and the corresponding discrete iterates.

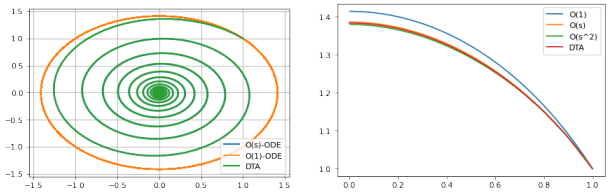

Figure 7: Trajectory of different ODEs whose flows follow the discrete time algorithm with increased precision. We see that they are hard to tell apart.

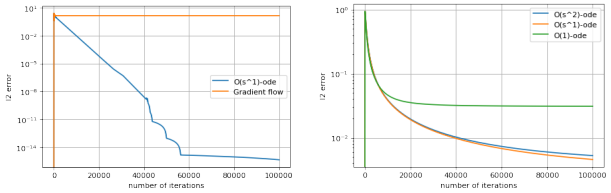

Figure 8: Error between the $O(s^r)$-ODE and the DTA. On the left $l(x, y) = xy$, on the right $l(x, y) = x^3y^3$.

From this little experiment we see that there is a great improvement in precision between the naïve $O(1)$-ODE and the high-order differential equations. Surprisingly, the $O(s^2)$-ODE does not improve precision.

VI. Conclusions

In this work, we have tested the convergence properties of various optimization algorithms with different loss functions for the min-max problem. We then verified that the high-resolution ODE described in [2] does indeed follow the DTA with increased precision. In a future work, we would like to test a greater variety of loss functions, hoping that this would help find a previously unknown pattern. Finally, applying these algorithms to concrete min-max problems such as GANs could be the object of a future work.

REFERENCES

[1] G. Korpelevich. "The extragradient method for finding saddle points and other problems". In: *Ehkon. Mat. Metody* (1976).

[2] Haihao Lu. "An $O(Sr)$-Resolution ODE Framework for Understanding Discrete-Time Algorithms and Applications to the Linear Convergence of Minimax Problems". In: *arXiv:2001.08826 [cs, math]* (July 2021). arXiv: 2001.08826 [cs, math].

[3] Kaplan et al. "Proximal Point Methods and Nonconvex Optimization". In: *Journal of Global Optimization* ().

[4] Mokhtari et al. "A Unified Analysis of Extragradient and Optimistic Gradient Methods for Saddle Point Problems: Proximal Point Approach". In: *https://arxiv.org/abs/1901.08511* ().

[5] Weijie Su, Stephen Boyd, and Emmanuel Candes. "A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights". In: (), p. 9.