No-Regret Learning in Games is Turing Complete

Gabriel P. Andrade
University of Colorado Boulder
Boulder, Colorado, USA
gabriel.andrade@colorado.edu

Rafael Frongillo
University of Colorado Boulder
Boulder, Colorado, USA
raf@colorado.edu

Georgios Piliouras
SUTD
Singapore, Singapore
georgios@sutd.edu.sg

ABSTRACT

Games are natural models for multi-agent machine learning settings, such as generative adversarial networks (GANs). The desirable outcomes from algorithmic interactions in these games are encoded as game theoretic equilibrium concepts, e.g. Nash and coarse correlated equilibria. As directly computing an equilibrium is typically impractical, one often aims to design learning algorithms that iteratively converge to equilibria. A growing body of negative results casts doubt on this goal, from non-convergence to chaotic and even arbitrary behaviour. In this paper we add a strong negative result to this list: learning in games is Turing complete. Specifically, we prove Turing completeness of the replicator dynamic on matrix games, one of the simplest possible settings. Our results imply the undecicability of reachability problems for learning algorithms in games, a special case of which is determining equilibrium convergence.

KEYWORDS

Game Theory; Online Learning; Replicator Dynamics; Regret; Dynamical Systems; Turing Completeness; Undecidability

1 INTRODUCTION

Many multi-agent machine learning settings can be modeled as games, from social or economic systems with algorithmic decision-makers to popular learning architectures such as generative adversarial networks (GANs). Desired outcomes in these settings are often encoded as equilibrium concepts, and therefore a primary goal is identifying machine learning algorithms with provable convergence to these equilibria.

While there has been progress in deriving strong time-average convergence guarantees for popular online learning algorithms, the per-iteration behaviour of learning in games remains elusive. Recent results attempt to formalize how elusive these dynamics can be, from non-convergence results to establishing chaotic, or even essentially arbitrary, behaviour [1, 3, 8, 10, 15]. Experiments confirm that chaos can actually be typical behaviour [21].

In this work, we add an even more sobering negative result to this list: learning in games is Turing complete. Specifically, we show that replicator dynamics in matrix games, one of the simplest possible settings, can simulate an arbitrary Turing machine (Theorem 1). Here simulation is defined in terms of reachability, a natural decision problem for dynamical systems that asks whether a given system and initial condition eventually intersects (reaches) a certain set; a dynamical system simulates a Turing machine if the corresponding halting problem reduces to the reachability problem.

Our proof combines two recent results, on the Turing completeness of fluid dynamics [5], and on the approximate universality of learning in games [1].

We believe our results have far-reaching implications for the literature on learning in games. Most immediate is the fact that the reachability problem is undecidable for no-regret learning in general (Corollary 3). This result calls into question the feasibility of equilibration as a goal, since even deciding whether a learning algorithm gets close to an equilibrium is a special case of reachability. More broadly, these results establish the computational power of learning dynamics in games—and accordingly, their inherent complexity as formalized by computabiity theory.

Beyond the continuous-time setting, we borrow tools from numerical analysis to show that the multiplicative weights algorithm can simulate any bounded Turing machine (Theorem 2). Extending this analysis to arbitrary Turing machines, and thus establishing Turing completeness for the discrete-time setting, may not be possible with the techniques we consider. Establishing (or refuting) the Turing completeness of multiplicative weights is therefore left as an important open question, and one that will likely require entirely new techniques.

2 PRELIMINARIES

2.1 Matrix Games

A *finite n-player normal form game* consists of n agents $[n] = \{1, \ldots, n\}$, where each agent $i \in [n]$ can choose actions from a finite action set S_i. Actions are chosen by agent i according to a *mixed strategy*, a distribution $\mathbf{x}_i$ in the probability $|S_i|$-simplex $\Delta^{|S_i|} = \{\mathbf{x}_i \in \mathbb{R}_+^{|S_i|} : \sum_{s \in S_i} x_{is} = 1\}$. In normal form games, agents receive payoffs from pairwise interactions according to payoff matrices $A_{i,j} \in \mathbb{R}^{|S_i| \times |S_j|}$ where $i, j \in [n]$ and $i \neq j$. Given that mixed strategies $\mathbf{x}_i \in \Delta^{|S_i|}$ and $\mathbf{x}_j \in \Delta^{|S_j|}$ are chosen, agent i receives payoff $\mathbf{x}_i^\mathsf{T} A_{i,j} \mathbf{x}_j$. These payoffs yield a natural optimization problem for each agent, where agents act strategically and independently to maximize their expected payoff over the other agents' mixed strategies, i.e.

$$\max_{\mathbf{x}_i \in \Delta^{|S_i|}} \sum_{j \in [n];\ j \neq i} \mathbf{x}_i^\mathsf{T} A_{i,j} \mathbf{x}_j, \qquad i \in [n] . \tag{1}$$

Throughout the paper we'll restrict our attention to the case known as *matrix games*, when $n = 2$.

2.2 Follow-the-Regularized-Leader (FTRL) Learning and Replicator Dynamics

In many game settings, the optimization in eq. (1) is a moving target since the opponent adaptively updates their strategy and the payoff matrix may be unknown. In such settings, arguably the most well known class of algorithms is Follow-the-Regularized-Leader (FTRL).

The continuous-time version of an FTRL algorithm is as follows. Given initial payoff vector $\mathbf{y}_i(0)$, an agent i that plays against agent j in a matrix game $A_{i,j}$ updates their strategy at time t according to

$$\mathbf{y}_i(t) = \mathbf{y}_i(0) + \int_0^t A_{i,j}\mathbf{x}_j(s)ds$$
$$\mathbf{x}_i(t) = \arg\max_{\mathbf{x}_i \in \Delta^{|S_i|}}\{\langle \mathbf{x}_i, \mathbf{y}_i(t) \rangle - h_i(\mathbf{x}_i)\}$$
(2)

where h_i is strongly convex and continuously differentiable. FTRL effectively performs a balancing act between exploration and exploitation. The cumulative payoff vector $\mathbf{y}_i(t)$ indicates the total payouts until time t, i.e. if agent i had played strategy $s_i \in S_i$ continuously from $t = 0$ until time t, agent i would receive a total reward of $\mathbf{y}_{is_i}(t)$. The two most well-known instantiations of FTRL dynamics are the online gradient descent algorithm when $h_i(\mathbf{x}_i) = ||\mathbf{x}_i||_2^2$, and the replicator dynamics (the continuous-time analogue of Multiplicative Weights Update [2]) when $h_i(\mathbf{x}_i) = \sum_{s_i \in S_i} \mathbf{x}_{is_i} \ln \mathbf{x}_{is_i}$. FTRL dynamics in continuous time has bounded regret in arbitrary games [17]. For more information on FTRL dynamics and online optimization, see [24].

In this paper, we will focus on *replicator dynamics* (RD) as the learning process generating game dynamics. In addition to its role in optimization, replicator dynamics is the prototypical dynamic studied in evolutionary game theory [22, 30] and is one of the key mathematical models of evolution and biological competition [23, 28]. In this context, replicator dynamics can be thought of as a normalized form of ecological population models, and is studied given a single payoff matrix A and a single probability distribution $\mathbf{x}$ that can be thought abstractly as capturing the proportions of different species/strategies in the current population. Species/strategies get randomly paired up and the resulting payoff determines which strategies will increase/decrease over time.

Formally, the dynamics are as follows. Let $A \in \mathbb{R}^{m \times m}$ be a matrix game and $\mathbf{x} \in \Delta^m$ be the mixed strategy played. RD on A are given by:

$$\dot{x}_i = \frac{dx_i}{dt} = x_i\left((A\mathbf{x})_i - \mathbf{x}^\top A\mathbf{x}\right), \qquad i \in [m]$$
(3)

Under the symmetry of $A_{i,j} = A_{j,i}$, and of initial conditions (i.e. $\mathbf{x}_i = \mathbf{x}_j$ at $t = 0$), it is immediate to see that under the $\mathbf{x}_i, \mathbf{x}_j$ solutions of eq. (2) are identical to each other and to the solution of eq. (3) with $A = A_{i,j} = A_{j,i}$. For our purposes, it will suffice to focus on exactly this setting of matrix games defined by a single payoff matrix A and a single probability distribution $\mathbf{x}$, which is actually the standard setting within evolutionary game theory.

2.3 Dynamical Systems Theory

A dynamical system is a mathematical model of a time-evolving process. The objects undergoing change in a dynamical system is called its *state* and is often denoted by $\mathbf{x} \in \mathbb{X}$, where $\mathbb{X}$ is a topological space called a *state space*. For most of this paper we will be focusing on *continuous time* systems, but in §5 we will consider *discrete time* systems derived from numerical approximations of their continuous counterpart. To distinguish between continuous and discrete time, we will use $\mathbf{x}(t)$ to describe the state as a function of continuous time $t \in \mathbb{R}$ and $\mathbf{x}^t$ to describe the state as a function of discrete time $t \in \mathbb{Z}$.

Change between states in a continuous time dynamical system is described by a *flow* $\Phi : \mathbb{X} \times \mathbb{R} \to \mathbb{X}$ satisfying two properties:

(i) For each $t \in \mathbb{R}$, $\Phi(\cdot, t) : \mathbb{X} \to \mathbb{X}$ is bijective, continuous, and has a continuous inverse.
(ii) For every $s, t \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{X}$, $\Phi(\mathbf{x}, s + t) = \Phi(\Phi(\mathbf{x}, t), s)$.

Intuitively, flows describe the evolution of states in the dynamical system. Given a time $t \in \mathbb{R}$, the flow gives us the relative movement of every point $\mathbf{x} \in \mathbb{X}$; we will denote this by the map $\Phi^t : \mathbb{X} \to \mathbb{X}$. Similarly, given a point $\mathbf{x} \in \mathbb{X}$, the flow captures the trajectory of $\mathbf{x}$ as a function of time; in an abuse of notation, we will denote this by $\mathbf{x}(t)$ where t is changing.

Continuous time dynamical systems are often given as systems of *ordinary differential equations* (ODEs). Systems of ODEs describe a *vector field* $V : \mathbb{X} \to T\mathbb{X}$ which assigns to each $\mathbf{x} \in \mathbb{X}$ a vector in the tangent space of $\mathbb{X}$ at $\mathbf{x}$. The unit sphere $\mathbb{S}^n = \{\mathbf{x} \in \mathbb{R}^{n+1} : ||x||_2^2 = 1\}$ will play a special role in proving Theorem 1, in which case the tangent space $T\mathbb{S}^n$ at each $\mathbf{x} \in \mathbb{S}^n$ is $\{\mathbf{y} \in \mathbb{R}^n : \mathbf{x} \cdot \mathbf{y} = 0\}$. Intuitively, the tangent space defines bundles of vectors that ensure the system's states remain well defined on the state space as time progresses. A system of ODEs is said to *generate* (or *give*) a flow Φ if Φ describes a solution of the ODEs at each point $\mathbf{x} \in \mathbb{X}$. Throughout this paper we assume that all dynamical systems discussed can be given by a system of ODEs. For this reason, we will use the term *dynamical system* to refer to the system of ODEs, the associated vector field, and a generated flow interchangeably. A well known result in dynamical systems theory states that, for Lipschitz-continuous systems of ODEs, the generated flow is unique (see [16, 19]) and using these terms interchangeably is well defined.

An important notion for proving Theorem 1, and for dynamical systems in general, is that of a *global attracting set* of the dynamical system. Let Φ be a flow generated by some dynamical system on $\mathbb{X}$. We say $\mathbb{Y} \subset \mathbb{X}$ is *forward invariant* for the flow Φ if $\Phi^t(\mathbf{y}) \in \mathbb{Y}$ for every $t \geq 0$, $\mathbf{y} \in \mathbb{Y}$. We say $\mathbb{Y} \subset \mathbb{X}$ is *globally attracting* for the flow Φ if $\mathbb{Y}$ is nonempty, forward invariant, and

$$\mathbb{Y} \supseteq \bigcap_{t>0}\{\Phi^t(\mathbf{x}) : \mathbf{x} \in \mathbb{X}\}.$$
(4)

Stated informally, if $\mathbb{Y}$ is globally attracting it will eventually capture the dynamics of Φ starting from any point in $\mathbb{X}$ after some transitionary period of time.

Now let $\mathbb{X}$ and $\mathbb{Y}$ be two topological spaces. We say that a function $f : \mathbb{X} \to \mathbb{Y}$ is a *homeomorphism* if (i) f is bijective, (ii) f is continuous, and (iii) f has a continuous inverse. Furthermore, two flows $\Phi : \mathbb{X} \times \mathbb{R} \to \mathbb{X}$ and $\Psi : \mathbb{Y} \times \mathbb{R} \to \mathbb{Y}$ are *homeomorphic* if there exists a homeomorphism $g : \mathbb{X} \to \mathbb{Y}$ such that for each $\mathbf{x} \in \mathbb{X}$ and $t \in \mathbb{R}$ we have $g(\Phi(\mathbf{x}, t)) = \Psi(g(\mathbf{x}), t)$. If g is also C^1 and has a C^1 inverse, then we say g is a *diffeomorphism* and that the flows Φ and Ψ are *diffeomorphic*. Observe that every diffeomorphism is also a homeomorphism, and thus every pair of diffeomorphic flows are also homeomorphic. Homeomorphic (resp. diffeomorphic) flows satisfy a strong, and typical, notion of equivalence between dynamical systems. Intuitively, two dynamical systems are homeomorphic if their trajectories can be mapped to one another by stretching and bending space.

2.4 Turing Machines

Throughout this paper we rely crucially on the notion of a Turing complete dynamical systems, i.e. a dynamical system able to simulate any Turing machine. We will briefly recall the Turing machine model and formalize its relationship with dynamical systems.

A *Turing machine* is given by a tuple $T = (Q, \Sigma, \delta, q_0, q_{halt})$ where

- Q is a finite *set of states*, including an *initial state* q_0 and a *halting state* q_{halt};
- Σ is an *alphabet* with cardinality at least two;
- $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$ is a *transition function*.

For a given Turing machine T and an *input tape* $s = (s_i)_{i \in \mathbb{Z}} \in \Sigma^{\mathbb{Z}}$, the Turing machine's computation is carried out according to the following process:

- [0] Initialize the *current state* q to q_0, and the current tape $w = (w_i)_{i \in \mathbb{Z}}$ to be s.
- [1] If $q = q_{halt}$ then halt the algorithm and return w as output. Otherwise compute $\delta(q, w_0) = (q', w_0', \sigma)$, where $\sigma \in \{-1, 0, 1\}$.
- [2] Update the current state and tape by setting $q = q'$ and the 0th position of w to $w_0 = w_0'$.
- [3] Update w with the σ shifted tape $(w_{i+\sigma})$, then return to [1].

Without loss of generality, we will assume that Turing machines adhere to standard simplifying conventions (cf. [26]). Specifically, we assume that the alphabet $\Sigma = \{0, \ldots, 9\}$ and any given tape of the Turing machine only has a finite number of symbols different from 0, where 0 represents the special "blank symbol". Under these assumptions it follows that there exists a finite (possibly large) integer $k_0 > 0$ such that any tape w satisfies

$$w = (w_i)_{i \in \mathbb{Z}} = \ldots 000 w_{-k_0} \ldots w_{k_0} 000 \ldots \tag{5}$$

with each $w_i \in \Sigma$. Equivalently, at any given step in the Turing machine's evolution, these assumptions ensure there can be at most $2k_0 + 1$ non-blank symbols on the tape. In particular, we get that the space of configurations of a Turing machine T is $Q \times A \subset Q \times \Sigma^{\mathbb{Z}}$, where A is the subset of strings taking the form (5).

The construction of dynamical systems that simulate Turing machines is at the heart of our results, and has been studied for various problems in physics [6, 9, 20]. Although equivalent definitions exist, our analyses will adopt the formalisms used by recent work on fluid dynamics [5, 27]. An analogous definition can be given for flows on a manifold.

DEFINITION 1. *A vector field X on a manifold M simulates a Turing machine T if there exists an explicitly constructible open set $U_{w_{-k}, \ldots, w_k} \subset M$ corresponding to each finite string $w_{-k}, \ldots, w_k \in \Sigma$, and an explicitly constructible point $p_s \in M$ corresponding to each $s \in \Sigma^{\mathbb{Z}}$, such that: T with input tape s halts with an output tape having values $w_{-k}, \ldots, w_k$ in positions $-k, \ldots, k$ respectively if and only if the trajectory of X through p_s intersects $U_{w_{-k}, \ldots, w_k}$.*

Intuitively, a dynamical system simulates a Turing machine if there is a correspondence between trajectories reaching certain sets and computations halting with certain configurations. In particular, constructing the point p_s depends only on the Turing machine T and input tape s, while constructing the set $U_{w_{-k}, \ldots, w_k}$ depends only on the specified halting configuration of T. Both here and

throughout the paper, we say a mathematical object (e.g. points, sets, or matrices) is *constructible* if it can be computed in finite time; constructability is not explicitly used in our arguments, but is important for nuanced technical reasons since it disallows pathological scenarios such as having all information about a machine's computations being encoded in an initial condition.

Definition 1 leads to a natural notion of Turing completeness for dynamical systems.

DEFINITION 2. *A dynamical system is* Turing complete *if it can simulate a universal Turing machine T.*

3 TURING COMPLETE DYNAMICS ON MATRIX GAMES

Our goal in this section is to establish the Turing completeness of replicator dynamics; in §3.1 we provide all precursory results required to prove the main result in §3.2.

3.1 Turing Complete Vector Fields and Approximation-Free Game-Embeddings

Our construction of Turing complete game dynamics relies crucially on the notion of *generalized Lotka-Volterra* (GLV) vector fields. In particular, two properties of GLV vector fields will play a key role in the proof: (i) *polynomial* vector fields on $\mathbb{R}^n_{++}$ are a special case of GLV vector fields, and (ii) GLV vector fields can be embedded into RD on a matrix games *without* approximation.

Formally, a GLV vector field is a vector field on $\mathbb{R}^n_{++}$ given by the system of ODEs

$$\dot{x}_i = \frac{dx_i}{dt} = x_i \left(\lambda_i + \sum_{j \in [m]} A_{ij} \prod_{k \in [n]} x_k^{B_{jk}} \right), \qquad i \in [n] \tag{6}$$

where m is some positive integer, $\lambda \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$, and $B \in \mathbb{R}^{m \times n}$. Since exponents given by B can be any real number, the terms in the parentheses are *multivariate generalized polynomials*. In special cases where the ODEs are standard multivariate polynomials, GLV vector fields equate to polynomial vector fields—a fact straightforwardly shown by noting that any polynomial vector field $P = \{P_i\}_{i \in [n]}$ on $\mathbb{R}^n_{++}$ is equivalent to the GLV vector field $\tilde{P} = \{x_i (\frac{1}{x_i} P_i)\}_{i \in [n]}$.

Polynomial and GLV vector fields play an integral role by allowing us to invoke recent results by [5] and [1]. The starting point of our construction can stated as follows:

PROPOSITION 1 (THEOREM 4.1 OF [5]). *There exists a constructible polynomial vector field X of degree 58 on $\mathbb{S}^{17}$ which is Turing complete and bounded.*

In Appendix A we provide a proof sketch of this result; we refer the reader to [5] for the full proof. In §3.2 we will extend the Turing completeness from Proposition 1 to replicator dynamics in matrix games by leveraging recent work by [1]. In essence, [1] showed that GLV vector fields can approximate essentially any dynamical system, and that any GLV vector field can be embedded into the dynamics of RD on some matrix game. In this paper we only rely on the latter result, since polynomial vector fields are already a special case of GLV vector fields and thus do not need to be approximated.

PROPOSITION 2 (THEOREM 3 OF [1]). *Let $\tilde{P}$ be a GLV vector field on $\mathbb{R}_{++}^n$ and Φ be the flow generated by $\tilde{P}$. For $m \geq n$, there exists a flow Θ on $\mathrm{relint}(\Delta^m)$ and a constructible diffeomorphism $f : \mathbb{R}_{++}^n \to \mathbb{P} \subseteq \mathrm{relint}(\Delta^m)$ such that:*

(i) *The flow Θ on $\mathrm{relint}(\Delta^m)$ is given by RD on a matrix game with payoff matrix $A \in \mathbb{R}^{m \times m}$.*

(ii) *The flow $\Theta\big|_{\mathbb{P}} = f(\Phi)$ and $\Phi = f^{-1}(\Theta\big|_{\mathbb{P}})$, where $\Theta\big|_{\mathbb{P}}$ is the flow given by Θ restricted to $\mathbb{P}$.*

(iii) *The integer $m - 1$ is at least the number of unique monomials in $\tilde{P}$.*

At a high level, proving Proposition 2 boils down to composing an embedding trick introduced by [4] with Theorem 7.5.1 by [13]. The relationship highlighted here between $m - 1$ and the number of monomials was not included in the original statement by [1], however it is shown as part of an important step in their proof and is required for Corollary 1.

3.2 Replicator Dynamics on Matrix Games is Turing Complete

To prove the main result of this section, Theorem 1, we will derive a diffeomorphism of the Turing complete vector field constructed in Proposition 1 that enables us to apply Proposition 2.

THEOREM 1. *There exists $m \geq 0$ and a constructible matrix game $A \in \mathbb{R}^{m \times m}$ such that replicator dynamics on A is Turing complete.*

PROOF. Let X be the Turing complete polynomial vector field on $\mathbb{S}^{17}$ given by Proposition 1. We begin by embedding X into a polynomial vector field $\overline{X}$ on $\mathbb{R}^{18}$ where $\mathbb{S}^{17}$ is globally attracting. Since trajectories of $\overline{X}$ are globally attracted to $\mathbb{S}^{17}$, a standard change of coordinates via translation yields a polynomial vector field that is well-defined on $\mathbb{R}_{++}^{18}$. Therefore, as polynomial vector fields on $\mathbb{R}_{++}^{18}$ are a special case of GLV vector fields, we will conclude the proof by applying Proposition 2 from §3.1.

Let $\{\phi_i\}_{i \in [18]}$ be the set of polynomials given by X. Define $\pi(\mathbf{x}) = (1 - \|\mathbf{x}\|_2^2)$ for $\mathbf{x} \in \mathbb{R}^{18}$. Now define $\overline{X}$ as the vector field on $\mathbb{R}^{18}$ given by the system

$$\dot{x}_i = x_i \left(\pi(\mathbf{x}) + \frac{1}{x_i} \phi_i(\mathbf{x}) \right)$$
$$= x_i \pi(\mathbf{x}) + \phi_i(\mathbf{x}) \, ,$$

for each $i \in [18]$. By construction $\mathbb{S}^{17}$ is forward invariant under $\overline{X}$, as $\pi(\mathbf{x}) = 0$ on $\mathbb{S}^{17}$ and X is forward invariant on $\mathbb{S}^{17}$. Furthermore, observe that for $\mathbf{x} = \mathbf{x}(t) \in \mathbb{R}^{18}$ the solutions of $\overline{X}$ satisfy

$$\frac{d}{dt} \|\mathbf{x}\|_2^2 = 2 \sum_{i \in [18]} x_i \dot{x}_i$$

$$= 2 \left(\sum_{i \in [18]} x_i^2 \pi(\mathbf{x}) + \sum_{i \in [18]} x_i \phi_i(\mathbf{x}) \right)$$

$$= 2\pi(\mathbf{x}) \left(\sum_{i \in [18]} x_i^2 \right) + 2 \left(\sum_{i \in [18]} x_i \phi_i(\mathbf{x}) \right)$$

$$= 2\pi(\mathbf{x}) \|\mathbf{x}\|_2^2$$

$$= 2\|\mathbf{x}\|_2^2 \left(1 - \|\mathbf{x}\|_2^2 \right) \, ,$$

since, by definition of $T\mathbb{S}^{17}$, the constraint $\|\mathbf{x}\|_2^2 = 1$ ensures X satisfies

$$2 \sum_{i \in [18]} x_i \phi_i(\mathbf{x}) = 0 \, .$$

The term $2\|\mathbf{x}\|_2^2 \left(1 - \|\mathbf{x}\|_2^2 \right)$ is a logistic equation in $\|\mathbf{x}\|_2^2$. Thus, for every $\mathbf{x} \in \mathbb{R}^{18}$, we know $\|\mathbf{x}\|_2^2 \to 1$ as $t \to \infty$. It follows that $\mathbb{S}^{17}$ is globally attracting for the trajectories generated by $\overline{X}$.

Denote a standard translation of axes by $\sigma \in \mathbb{R}$ as $F_\sigma : \mathbb{R}^{18} \to \mathbb{R}^{18}$, $\mathbf{x} \mapsto \mathbf{x} + \sigma \mathbb{1}$, where $\mathbb{1}$ is the all-ones vector. Since solutions of $\overline{X}$ are attracted to $\mathbb{S}^{17}$ and Proposition 1 ensures $\{\phi_i\}_{i \in [18]}$ is bounded due to the reparametrization done in eq. (4.2) in [5], there exists suitable values of σ such that composing F_σ with $\overline{X}$ yields a polynomial vector field that is forward invariant on $\mathbb{R}_{++}^{18}$. Formally, let $B > 0$ be the bound on $\{\phi_i\}_{i \in [18]}$ given in Proposition 1, i.e. for all $i \in [18]$ and $\mathbf{x} \in \mathbb{R}^{18}$ the vector field X satisfies $|\phi_i(\mathbf{x})| \leq B$. To ensure the translated vector field is forward invariant on $\mathbb{R}_{++}^{18}$, it suffices to find σ such that $Y = F_\sigma \circ \overline{X}$ is strictly positive on the boundary when $\mathbf{y} \in \mathbb{R}_{++}^{18}$ has $y_i = 0$ for some $i \in [18]$. By definition we know that Y at any $\mathbf{y} \in \mathbb{R}_{++}^{18}$ is identical to $\overline{X}$ at $\mathbf{x} = \mathbf{y} - \sigma \mathbb{1}$. The system of equations $\{\dot{y}_i\}_{i \in [18]}$ is given by the system of equations $\{\dot{x}_i\}_{i \in [18]}$ under the substitution $\mathbf{x} = \mathbf{y} - \sigma \mathbb{1}$. Therefore we find that, for $\mathbf{y} \in \mathbb{R}_{++}^{18}$ with $y_i = 0$ for some $i \in [18]$,

$$\dot{y}_i = (y_i - \sigma)\pi(\mathbf{y} - \sigma \mathbb{1}) + \phi_i(\mathbf{y} - \sigma \mathbb{1})$$

$$\geq (-\sigma)(1 - \|\mathbf{y} - \sigma \mathbb{1}\|_2^2) - B$$

$$= \sigma \|\mathbf{y} - \sigma \mathbb{1}\|_2^2 - \sigma - B$$

$$\geq \sigma^3 - \sigma - B \, ,$$

which implies $\dot{y}_i > 0$ whenever $B < -\sigma + \sigma^3$. Thus, for values of σ satisfying $B < -\sigma + \sigma^3$, we have $Y = F_\sigma \circ \overline{X}$ which is well defined on $\mathbb{R}_{++}^{18}$ for all initial conditions in $\mathbb{R}_{++}^{18}$.

By definition of Y, as a translated copy of $\overline{X}$, the set $F_\sigma(\mathbb{S}^{17})$ is globally attracting in Y, and $Y\big|_{F_\sigma(\mathbb{S}^{17})}$ is a Turing complete polynomial vector field. It follows we have constructed a polynomial vector on $\mathbb{R}_{++}^{18}$ that inherits the Turing complete dynamics of X. Since polynomial vector fields on $\mathbb{R}_{++}^{18}$ are a special case of GLV vector fields on $\mathbb{R}_{++}^{18}$, from Proposition 2 there exists a diffeomorphism $f : \mathbb{R}_{++}^{18} \to \mathbb{P} \subseteq \mathrm{relint}(\Delta^m)$ from trajectories of Y onto trajectories of an invariant submanifold of replicator dynamics on a matrix game $A \in \mathbb{R}^{m \times m}$.

We conclude by showing how the Turing completeness of X corresponds to Turing completeness for replicator dynamics on A. Suppose we have a given Turing machine T, an input tape s, and some finite string ω. By Proposition 1 there exists a point p_s and open set U_ω such that trajectories of X through p_s intersect U_ω if and only if T halts with input s and output matching ω about the machine's head. Our analysis above shows that $\overline{X}\big|_{\mathbb{S}^{17}} = X$, so trajectories of $\overline{X}$ through p_s intersect U_ω if and only if T halts with input s and output matching ω. Therefore, after translating $\overline{X}$, we know trajectories of Y through $F_\sigma(p_s)$ intersect $F_\sigma(U_\omega)$ if and only if T halts with input s and output matching ω. Finally, since diffeomorphisms are closed under composition, we conclude that trajectories of replicator dynamics on A through the point $f(F_\sigma(p_s))$ intersect the set $f(F_\sigma(U_\omega))$ if and only if T halts with

input s and output matching ω, where f is the diffeomorphism above. Thus, on an invariant submanifold of $\mathrm{relint}(\Delta^m)$, replicator dynamics on A simulates T. Taking T to be a universal Turing machine completes the proof. $\quad\square$

An interesting corollary of Theorem 1 is that we arrive at a bound on the number of actions needed for defining games where learning dynamics can be Turing complete. The bound is likely loose for several reasons. Firstly, the polynomial vector field from Proposition 1 is not known to have minimal degree nor dimension. Secondly, the combinatorial argument in Appendix B makes no attempt at a nuanced count on the number of unique monomials in the polynomials given by these vector fields. Deriving a tight bound is not only an interesting open question for game dynamics, but also for recent work in fluid dynamics [5, 29] and analog computing [12].

COROLLARY 1. *For some $m \leq \binom{76}{18} + 1$, there exists a matrix game $A \in \mathbb{R}^{m\times m}$ such that replicator dynamics on A is Turing complete.*

4 UNDECIDABLE PHENOMENA IN NO-REGRET LEARNING DYNAMICS

The Turing completeness of replicator dynamics (i.e. Theorem 1) has deep implications for machine learning and, more generally, learning in strategic environments. Specifically, if a dynamical system simulates a Turing machine, Definition 1 gives a reduction from the halting problem for Turing machines to the reachability problem for dynamical systems, which we use alongside the Turing completeness established in Theorem 1 to uncover the existence of undecidable reachability problems. As will be discussed in §4.2, the existence of undecidable problems makes it increasingly important that we understand computability in instances of reachability arising from fundamental solution concepts for game theory and machine learning.

4.1 The Halting and Reachability Problems

The halting problem is a prototypical decision problem for Turing machines and is arguably the most famous undecidable problem in computer science. Given a Turing machine T and an input tape, the *halting problem for T* asks whether or not T will halt. By contrast, the reachability problem is canonical for dynamical systems and has been studied in various control settings; given a dynamical system X and a set of initial conditions, the *reachability problem for X* asks whether or not X's trajectory will intersect a predetermined set. Although the computability of the halting problem is generally well understood in Turing machines, the computability of the reachability problem has not traditionally been studied in the context of game dynamics. However, from the strong equivalence between halting and reachability required by Definition 1, we immediately get a reduction between these classic decision problems.

PROPOSITION 3. *If a dynamical system X simulates a Turing machine T, then the halting problem for T reduces to the reachability problem for X.*

The proof of this proposition follows directly from Definition 1, since checking whether the dynamical system reaches a set becomes equivalent to checking whether the Turing machine halts by definition. From Theorem 1 we know that replicator dynamics

on a matrix game can simulate a universal Turing machine. Therefore, due to the undecidability of the halting problem in general, we deduce that the reachability problem can be undecidable for replicator dynamics on matrix games.

COROLLARY 2. *There exist matrix games where the reachability problem is undecidable for replicator dynamics.*

The corollary follows immediately from Proposition 3 and Theorem 1, since the undecidability of the halting problem for universal Turing machines uncovers the undecidability of the reachability problem for replicator dynamics on matrix games.

4.2 Implications for No-Regret Learning in Games

Games are primarily understood and studied via equilibrium concepts, e.g. Nash equilibria, evolutionary stable strategies, and coarse correlated equilibria. It is therefore unsurprising that a central goal of learning in games is often to converge on some set of equilibria. Yet, beyond certain special cases (e.g. potential games), learning behaviours remain largely enigmatic and there has been limited progress towards resolving non-convergence in general settings. The results in this paper may explain why: determining convergence to a set of equilibria is a special case of reachability, and identifying learning algorithms that provably converge on such a set may be an undecidable problem even in very simple classes of games. The goal of this section is to formalize this intuition.

In Corollary 2 we found that reachability can be undecidable for replicator dynamics on matrix games. Therefore, taken as a negative result, Corollary 2 implies that undecidable trajectories can exist in larger classes of game dynamics where replicator dynamics on matrix games is a special case. Unfortunately, replicator dynamics is special case of FTRL dynamics and no-regret learning dynamics more generally [17], which suggests these popular learning dynamics can inherit the negative result on any class of games containing matrix games. Similarly, matrix games are very restricted and a special case of many popular classes of games, e.g. normal form and smooth games. As an example of how broadly these results generalize, matrix games in the FTRL framework describe quadratic objective functions and thus undecidable trajectories exist for optimization-driven learning over quadratic objectives. Thus, as Corollary 2 holds for replicator dynamics on matrix games, we arrive at the reachability problem being generally undecidable for rich classes of game dynamics studied in the literature and used in practice.

COROLLARY 3. *There exist games where reachability for no-regret learning dynamics is undecidable.*

In light of Corollary 2, the claim follows from our discussion above. As determining convergence to sets of game theoretic solution concepts is a special case of the reachability problem, Corollary 3 reveals that determining whether game dynamics converge to fundamental solution concepts is undecidable in general. It is important to note that the undecidability may not hold for specific games, learning dynamics, or solution concepts; the primary takeaway is that undecidability is possible and has strong implications about how we should approach these important questions.

5 DISCRETE LEARNING DYNAMICS AND TURING MACHINE SIMULATIONS

Thus far we have focused on continuous-time replicator learning dynamics, but in practice discrete-time learning dynamics are typically used. A folk result in the study of game dynamics states that the multiplicative weights update (MWU) algorithm is essentially an Euler discretization of replicator dynamics. It is therefore natural to ask whether MWU, the discrete analogue of replicator dynamics, are also Turing complete. Unfortunately, as will be shown in this section, standard numerical error analyses are likely insufficient for proving Turing completeness in discrete time; intuitively, the reason is because discretizations of a continuous time process will yield error bounds that grow as a function of time. We will formalize these error bounds in §5.1 and use them in §5.2 to begin untangling the computational power of MWU. Discussions of related open questions are left for §6.

5.1 Discretization Error of Multiplicative Weights Updates

The fact that MWU is a discretization of replicator dynamics is well known in the field of game dynamics, but a precise derivation of this relationship is often omitted. For clarity in our analysis of discretization errors, we will highlight one possible discretization that reveals MWU as a discrete-analogue of replicator dynamics in Appendix C. The discretization we arrive at is used to find a bound on the cumulative error of MWU relative to replicator dynamics, which is crucial for the analyses and discussion to follow.

LEMMA 1. *Let Φ be the flow generated by replicator dynamics and $\boldsymbol{x}^t$ be the mixed strategy found on the tth iterate of MWU. The error accrued by a single iteration of MWU with step-size $\eta > 0$ is*

$$\|\boldsymbol{x}^{t+1} - \Phi(\eta, \boldsymbol{x}^t)\|_\infty \le 1 - e^{-2\eta} .$$

The proof of Lemma 1 consists of relatively straightforward calculations, but requires carefully handling nonlinearities introduced by MWU; a full proof is included in Appendix D.

Using Lemma 1 as a basis, we can bound the error accrued over multiple iterations of MWU.[1]

LEMMA 2. *Let Φ be the flow generated by replicator dynamics and $\boldsymbol{x}^t$ be the mixed strategy found on the tth iterate of MWU. The error accrued after $t + 1$ iterations of MWU with step-size $\eta > 0$ is*

$$\|\boldsymbol{x}^{t+1} - \Phi(t\eta, \boldsymbol{x}^0)\| \le O\left(e^t\right) .$$

A full derivation of Lemma 2 is found in Appendix E, and follows from using Lemma 1 alongside standard techniques for bounding error in iterated numerical methods.

5.2 Simulating Bounded Turing Machines with Multiplicative Weights Update

The result in Lemma 2 shows that, relative to replicator dynamics, the error accrued by MWU will grow with the number of iterations. Error growing as a function of time is problematic when simulating a Turing machine by using MWU as discretization of replicator dynamics.

---

[1]In the language of numerical analysis, Lemma 1 gives the local error used to find the global error in Lemma 2.

Recall that in Theorem 1 we showed that replicator dynamics can simulate a universal Turing machine because it can embed a dynamical system that simulates a universal Turing machine, which is done to ensure the Turing machine's halting remains equivalent to the dynamics' trajectories reaching a certain set. However, in general, determining whether such a Turing machine will halt or how many steps are required to halt is undecidable. Therefore, without an a priori bound on the maximum amount of time needed to determine whether the machine halts or not, we cannot choose step sizes for MWU that guarantee the discretization remains sufficiently close to replicator dynamics when intersecting the relevant set.

THEOREM 2. *Let $k > 0$ be a finite integer and $\mathbb{T}_k$ be the set of Turing machines that we can determine to halt or not after k steps of computation. For any k, there exists step sizes $\eta > 0$ such that MWU with step-size η can simulate any Turing machine in $\mathbb{T}_k$.*

The result follows from the construction of the open sets used in Proposition 1 and the fact that we can ensure MWU's discretization error stays sufficiently small over any finite window of time due to Lemma 2. Resolving the limitations of Theorem 2, and uncovering the true computational power of discrete algorithms such as MWU, will likely require new technical approaches for bounding errors or simulating Turing machines.

6 CONCLUSION

We have shown that replicator dynamics in matrix games can simulate universal Turing machines. In continuous time, this observation was extended to provide deeper insight into the complexities of game theoretic learning. In fact, as highlighted in §4, the plurality of negative results on game dynamics can be understood as a natural byproduct of Theorem 3. Given that the present paper uses replicator dynamics specifically and matrix games broadly, complimenting the results given here with analyses based on other learning dynamics and classes of games could be instrumental in guiding future research by finding settings where designing well-behaved game dynamics is a tractable problem. As was done for Turing machines in computational complexity theory and becomes more natural given the techniques used in our analyses, compartmentalizing the complexity of learning in games using traditional complexity classes suggests a promising line of investigation for finding tractable settings for learning in games.

In discrete time, the Turing completeness of replicator dynamics was used to show that MWU can simulate bounded Turing machines. However, our approach does not disallow for the possibility of MWU being Turing complete as well; using MWU's relationship to replicator dynamics seems to have inherent numerical limitations arising from error growing with time. Since discrete-time learning is more applicable in practice, it remains an important open question to determine whether MWU and other discrete learning algorithms are Turing complete. That being said, the smoothness constraints on continuous-time learning often leads to better behaved dynamics than discrete-analogues, and thus the study of continuous dynamics generally serves as restricted special case of what is possible in discrete-time. As evidence of this claim, not only are complex dynamic phenomena prevalent in low dimensional discrete systems where it is impossible in continuous systems (e.g. chaos [7]),
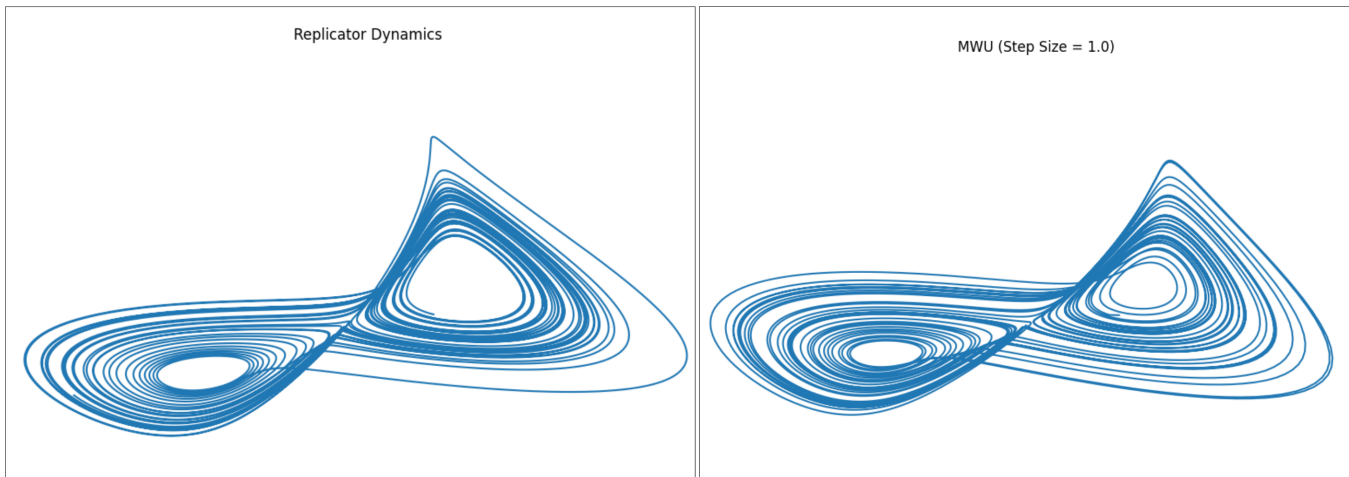
Figure 1: A comparison of replicator dynamics and MWU on a matrix game derived by [1] to simulate a chaotic dynamical system. On the left is replicator dynamics with the dynamics embedded into its behaviours, whereas on the right we have 10000 iterations of MWU with a relatively large step size. Although not identical, it is clear that MWU retains the intricate and complex behaviours of replicator dynamics.

but Figure 1 demonstrates the robustness of MWU by showing it can follow replicator dynamics on a matrix game derived by [1] in order to emulate the iconic Lorenz strange attractor. In future work, instead of using continuous learning dynamics as a proxy, directly simulating Turing machines with discrete dynamics may provide powerful tools for learning in games. Research on Turing machine simulations using physical systems has a rich history and encompasses far more than what is discussed in this paper. Various techniques have been used to directly simulate Turing machines using discrete dynamics [18, 25], and insights from this prior work may hold potent insights for applications to learning in games.

ACKNOWLEDGMENTS

REFERENCES

[1] Gabriel P. Andrade, Rafael Frongillo, and Georgios Piliouras. 2021. Learning in Matrix Games can be Arbitrarily Complex. In *Proceedings of Thirty Fourth Conference on Learning Theory (Proceedings of Machine Learning Research, Vol. 134)*, Mikhail Belkin and Samory Kpotufe (Eds.). PMLR, 159–185. https://proceedings.mlr.press/v134/andrade21a.html

[2] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing* 8, 1 (2012), 121–164.

[3] Michel Benaïm, Josef Hofbauer, and Sylvain Sorin. 2012. Perturbations of Set-Valued Dynamical Systems, with Applications to Game Theory. *Dynamic Games and Applications* 2 (2012), 195–205.

[4] L. Brenig and A. Goriely. 1989. Universal canonical forms for time-continuous dynamical systems. *Phys. Rev. A* 40 (Oct 1989), 4119–4122. Issue 7. https://doi.org/10.1103/PhysRevA.40.4119

[5] Robert Cardona, Eva Miranda, and Daniel Peralta-Salas. 2021. Turing Universality of the Incompressible Euler Equations and a Conjecture of Moore. *International Mathematics Research Notices* (08 2021). https://doi.org/10.1093/imrn/rnab233 arXiv:https://academic.oup.com/imrn/advance-article-pdf/doi/10.1093/imrn/rnab233/39904547/rnab233.pdf rnab233.

[6] Robert Cardona, Eva Miranda, Daniel Peralta-Salas, and Francisco Presas. 2021. Constructing Turing complete Euler flows in dimension 3. *Proceedings of the National Academy of Sciences* 118, 19 (2021). https://doi.org/10.1073/pnas.2026818118 arXiv:https://www.pnas.org/content/118/19/e2026818118.full.pdf

[7] Thiparat Chotibut, Fryderyk Falniowski, Michal Misiurewicz, and Georgios Piliouras. 2020. Family of chaotic maps from game theory. *Dynamical Systems* (2020). https://doi.org/10.1080/14689367.2020.1795624 https://doi.org/10.1080/14689367.2020.1795624.

[8] Lampros Flokas, Emmanouil-Vasileios Vlatakis-Gkaragkounis, Thanasis Lianeas, Panayotis Mertikopoulos, and Georgios Piliouras. 2020. No-regret learning and mixed Nash equilibria: They do not mix. In *NeurIPS*.

[9] Michael H. Freedman. 1998. P/NP, and the quantum field computer. *Proceedings of the National Academy of Sciences* 95, 1 (1998), 98–101. https://doi.org/10.1073/pnas.95.1.98 arXiv:https://www.pnas.org/content/95/1/98.full.pdf

[10] Angeliki Giannou, Emmanouil Vasileios Vlatakis-Gkaragkounis, and Panayotis Mertikopoulos. 2021. Survival of the strictest: Stable and unstable equilibria under regularized learning with partial information. In *Proceedings of Thirty Fourth Conference on Learning Theory (Proceedings of Machine Learning Research, Vol. 134)*, Mikhail Belkin and Samory Kpotufe (Eds.). PMLR, 2147–2148. https://proceedings.mlr.press/v134/giannou21a.html

[11] Daniel S. Graça, Manuel L. Campagnolo, and Jorge Buescu. 2008. Computability with polynomial differential equations. *Advances in Applied Mathematics* 40, 3 (2008), 330–349. https://doi.org/10.1016/j.aam.2007.02.003

[12] Emmanuel Hainry. 2009. Decidability and undecidability in dynamical systems. (2009).

[13] Josef Hofbauer and Karl Sigmund. 1998. *Evolutionary Games and Population Dynamics*. Cambridge University Press. https://doi.org/10.1017/CBO9781139173179

[14] Josef Hofbauer, Sylvain Sorin, and Yannick Viossat. 2009. Time Average Replicator and Best-Reply Dynamics. *Mathematics of Operations Research* 34, 2 (2009), 263–269. http://www.jstor.org/stable/40538381

[15] Alistair Letcher. 2021. On the Impossibility of Global Convergence in Multi-Loss Optimization. arXiv:2005.12649 [math.OC]

[16] James D. Meiss. 2007. *Differential Dynamical Systems (Monographs on Mathematical Modeling and Computation)*. Society for Industrial and Applied Mathematics, USA.

[17] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. 2018. Cycles in Adversarial Regularized Learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (New Orleans, Louisiana)

(SODA '18). Society for Industrial and Applied Mathematics, USA, 2703–2717.

[18] Cristopher Moore. 1990. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.* 64 (May 1990), 2354–2357. Issue 20. https://doi.org/10.1103/PhysRevLett.64.2354

[19] Lawrence Perko. 1991. *Differential Equations and Dynamical Systems.* Springer-Verlag, Berlin, Heidelberg.

[20] John H. Reif, J. Doug Tygar, and A. Yoshida. 1994. Computability and complexity of ray tracing. *Discrete & Computational Geometry* 11 (1994), 265–288.

[21] James B. T. Sanders, J. Doyne Farmer, and Tobias Galla. 2018. The prevalence of chaotic dynamics in games with many players. *Scientific reports* 8, 1 (2018), 1–13.

[22] William H. Sandholm. 2010. *Population Games and Evolutionary Dynamics.* MIT Press.

[23] Peter Schuster and Karl Sigmund. 1983. Replicator dynamics. *Journal of Theoretical Biology* 100, 3 (1983), 533 – 538. https://doi.org/10.1016/0022-5193(83)90445-9

[24] Shai Shalev-Shwartz. 2012. Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning* 4, 2 (2012), 107–194. https://doi.org/10.1561/2200000018

[25] Hava T. Siegelmann and Eduardo D. Sontag. 1992. On the Computational Power of Neural Nets. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (Pittsburgh, Pennsylvania, USA) *(COLT '92)*. Association for Computing Machinery, New York, NY, USA, 440–449. https://doi.org/10.1145/130385.130432

[26] Michael Sipser. 1996. *Introduction to the Theory of Computation* (1st ed.). International Thomson Publishing.

[27] Terence Tao. 2017. On the universality of potential well dynamics. *Dynamics of Partial Differential Equations* 14, 3 (2017), 219–238.

[28] Peter D. Taylor and Leo B. Jonker. 1978. Evolutionary stable strategies and game dynamics. *Mathematical Biosciences* 40, 1 (1978), 145 – 156. https://doi.org/10.1016/0025-5564(78)90077-9

[29] Francisco Torres de Lizaur. 2022. Chaos in the incompressible Euler equation on manifolds of high dimension. *Inventiones mathematicae* (01 2022), 1–29. https://doi.org/10.1007/s00222-021-01089-3

[30] Jörgen W. Weibull. 1995. *Evolutionary Game Theory.* MIT Press; Cambridge, MA: Cambridge University Press.

A   TURING COMPLETE POLYNOMIAL FLOWS ON $\mathbb{S}^{17}$

We will briefly sketch the construction by [5] of the Turing complete polynomial vector field in Proposition 1; for a complete treatment we refer the reader to [5]. To simplify notation, throughout this section we will represent a step in a Turing machine T's evolution (i.e. an iteration of Steps 1–3 in §2.4) by the global transition function

$$G_T : Q \times A \to Q \times A \,,$$

where we set $G_T(q_{\text{halt}}, w) := (q_{\text{halt}}, w)$ for any tape w.

Let $T = (Q, \Sigma, \delta, q_0, q_{\text{halt}})$ be a Turing machine. We begin by encoding each configuration of T as a constructible point in $\mathbb{R}^3$. Let $r = |Q|$ be the cardinality of the set of states Q, then we will represent the elements of Q by $[r] = \{1, \ldots, r\} \in \mathbb{N}$. Since we know tapes satisfy eq. (5), we can encode any such $w = (w_i)_{i \in \mathbb{Z}}$ as the pair of points in $\mathbb{N}^2$ given by

$$y_1 = w_0 + w_1 10 + \cdots + w_{k_0} 10^{k_0}$$
$$y_2 = w_{-1} + w_{-2} 10 + \cdots + w_{-k_0} 10^{k_0-1} \,.$$

Taken together, we have an encoding of every $(q, w) \in Q \times A$ as $(y_1, y_2, q) \in \mathbb{N}^3 \subset \mathbb{R}^3$. Define $\zeta : Q \times A \to \mathbb{N}^3$ as the map assigning each configuration in $Q \times A$ its associated point in $\mathbb{N}^3$ that we constructed. The global transition function G_T can now be reinterpreted as a map from $\zeta(Q \times A) \subset \mathbb{N}^3$ to itself. By extending said map to be the identity map on points in $\mathbb{N}^3 \setminus \zeta(Q \times A)$, we arrive at a map on the whole of $\mathbb{N}^3$ to itself—for simplicity, we will denote this extended map by $G_{\zeta(T)} : \mathbb{N}^3 \to \mathbb{N}^3$.

Using this encoding, the next step in the construction is to simulate T using a polynomial vector field P on $\mathbb{R}^{n+3}$. To this end, a

modification of a construction by [11] is given. Specifically, [11] construct a non-autonomous polynomial vector field that simulates T, and this vector field is made autonomous via a standard trick of introducing a proxy variable in place of the explicit dependence on time. Let P on $\mathbb{R}^{n+3}$ be the *autonomous* polynomial vector field derived via this modification. The construction by [11] also shows how, given an input tape $s \in A$, a point $p_s = (\zeta(q_0, s), \tilde{y}_0) \in \mathbb{R}^{n+3}$ is constructed so that the trajectory of P starting from p_s will simulate $G_{\zeta(T)}$. The term $\zeta(q_0, s) \in \mathbb{R}^3$ is defined above and the term $\tilde{y}_0 \in \mathbb{R}^n$ is from a composition of polynomials depending only on T and s—neither of these points are affected by the modification and can be taken as is. The group property of flows ensures that any trajectory passing through p_s is equivalent to a trajectory ending at and then "restarting" from p_s, so we can assume p_s is an initial condition in Definition 2 without loss of generality. Suppose we have a finite string $w^* = (w^*_{-k}, \ldots, w^*_k)$ of symbols in Σ, we will now construct the set U_{w^*} in Definition 2.[2] Let $\omega = \{w \in \Sigma^{\mathbb{Z}} \mid w_i = w^*_i \, \forall i \in [-k, k]\}$, $\epsilon > 0$ be a small positive constant, and $\mathbb{R}^3 \big|_{\zeta(q_{\text{halt}}, \omega)}$ be the set of points in $\mathbb{R}^3$ corresponding to configurations of T of the form $(q_{\text{halt}}, w \in \omega)$. Defining $U^\epsilon_{w^*} \subset \mathbb{R}^3$ as an ϵ-neighborhood of $\mathbb{R}^3 \big|_{\zeta(q_{\text{halt}}, \omega)}$ gives the open set

$$U_{w^*} := U^\epsilon_{w^*} \times \mathbb{R}^n \,.$$

Showing P satisfies Definition 2 with this choice of p_s and U_{w^*} follows from a relatively straightforward argument using properties inherited from the construction by [11]. Finally, the polynomial vector field X in Proposition 1 is constructed by using the pullback of inverse stereographic projection on a suitable reparametrization of P and taking T to be a universal Turing machine. The pullback of inverse stereographic projection ensures that X is a polynomial vector field tangent to the sphere and the reparametrization ensures the vector field is bounded.[3] The fact that X is well-defined on $\mathbb{S}^{17}$ and has degree 58 follows from an analysis by [12] of the construction by [11].

B   PROOF OF COROLLARY 1

COROLLARY 1. *For some $m \leq \binom{76}{18} + 1$, there exists a matrix game $A \in \mathbb{R}^{m \times m}$ such that replicator dynamics on A is Turing complete.*

PROOF. Let $X, \overline{X}$, and Y be the vector fields defined in the proof of Theorem 1. Similarly, let $A \in \mathbb{R}^{m \times m}$ be the matrix game we arrived at by applying Proposition 2 to Y. From Proposition 2 we know that $m - 1$ is at least the number of unique monomials in the generalized polynomials in Y, so the proof follows by bounding the number of unique monomials from above.

From Proposition 1 we know that X is a polynomial vector field of degree 58. As mentioned in Appendix A, the specific degree of

---

[2]For brevity we will brush over the construction of this set on the component corresponding to the proxy variable for time. Technically this component should be a union of small open intervals for each $i \in \mathbb{N}$, which intuitively associates a rough length of time in the dynamical system with a step in the Turing machine. However, formally introducing this portion of the construction is not particularly insightful since the relevance to the proof is rather tautological due to the proxy variable monotonically increasing at the same constant rate as time.

[3]Technically $X = \overline{X} \big|_{\mathbb{S}^{n+4}}$, where $\overline{X}$ is a polynomial vector field on $\mathbb{R}^{n+5}$ and tangent to $\mathbb{S}^{n+4}$. Similarly, as discussed in the proof of Theorem 1.3 by [5], the reparametrization ensures the vector field is global because it is bounded.

58 was derived from follow-up work by [12] analyzing the construction by [11]. However, although the vector field is technically constructible, actually constructing X to simulate a universal Turing machine is non-trivial in practice. With this complication in mind, a crude upper bound on the number of unique monomials in X is simply the number of unique monomials of degree 58 in 18 variables. Therefore, a standard combinatorial argument tells us that the number of unique monomials in the polynomials of X is at most $\binom{58+18}{18} = \binom{76}{18}$. The construction of $\overline{X}$ cannot increase the number of monomials counted by this combinatorial argument since it can only introduce unique monomials via the term $1 - \|\mathbf{x}\|_2^2$, which is already counted in the bound $\binom{76}{18}$. Similarly, we construct Y by translating $\overline{X}$ by a constant and therefore can only introduce the constant monomial (i.e. terms with all variables having zero exponents) which is already being counted. Thus we have found that $m - 1 \leq \binom{76}{18}$, which implies $m \leq \binom{76}{18} + 1$. □

C DERIVING MWU AS DISCRETE-ANALOGUE OF REPLICATOR DYNAMICS

Let $\delta : \mathbb{R}^n \to \Delta^n$ be the *logit map* defined as

$$\delta_i(\mathbf{y}) = \frac{\exp(\mathbf{y}_i)}{\sum_{j \in [n]} \exp(\mathbf{y}_j)}, \qquad \mathbf{y} \in \mathbb{R}^n, i \in [n].$$

[14] showed that the flow generated by replicator dynamics can be written as

$$\mathbf{x}_i(t) = \delta_i(\mathbf{y}(t)) = \frac{\exp(\mathbf{y}_i(t))}{\sum_{j \in [n]} \exp(\mathbf{y}_j(t))}, \qquad \mathbf{y}(0) \in \mathbb{R}^n, i \in [n], t \in \mathbb{R},$$
(7)

where $\mathbf{x}$ and $\mathbf{y}$ are the mixed strategy and cumulative payoff vectors given in eq. (2). Rewriting eq. (7) in the form of eq. (2) gives an explicit representation of replicator dynamics' trajectories as a functions of cumulative payoffs,

$$\mathbf{y}_i(t) = \mathbf{y}_i(0) + \int_0^t \sum_{j \in [n]} A_{i,j} \delta_j(\mathbf{y}(s)) ds$$
(8)
$$\mathbf{x}_i(t) = \delta_i(\mathbf{y}(t)).$$

By applying a standard Euler discretization with step size η to the payoffs $\mathbf{y}$ in eq. (8), we find

$$\mathbf{y}_i(t + \eta) \approx \mathbf{y}_i(t) + \eta \, \dot{\mathbf{y}}_i(t) = \mathbf{y}_i(t) + \eta \sum_{j \in [n]} A_{i,j} \delta_j(\mathbf{y}(t)).$$

Finally, iteratively applying this Euler discretization of the cumulative payoffs and using the logit map will give us the well-known MWU algorithm. Formally, denoting the discretization's t^{th} iterate by $\mathbf{y}^t$, we write MWU as

$$\mathbf{y}_i^{t+1} = \mathbf{y}_i^t + \eta \sum_{j \in [n]} A_{i,j} \delta_j(\mathbf{y}^t) = \mathbf{y}_i^0 + \eta \sum_{\tau=1}^t A_{i,j} \delta_j(\mathbf{y}^t)$$
$$\mathbf{x}_i^{t+1} = \delta_i(\mathbf{y}^{t+1}) = \delta_i(\mathbf{y}^0 + \eta \sum_{\tau=1}^t A_{i,j} \delta_j(\mathbf{y}^t)).$$
(9)

As the form of MWU in eq. (9) was found via an Euler discretization, a standard result in numerical analysis tells us that the error accrued by a single iteration of MWU starting from the same initial

conditions as replicator dynamics will satisfy

$$\|\mathbf{y}_i^1 - \mathbf{y}_i(\eta)\| \leq O(\eta^2).$$

However, since we are simulating Turing machines in the space of mixed strategies, we need error bounds on the probability simplex itself and not in the space of cumulative payoffs.

D PROOF OF LEMMA 1

LEMMA 1. *Let Φ be the flow generated by replicator dynamics and $\mathbf{x}^t$ be the mixed strategy found on the tth iterate of MWU. The error accrued by a single iteration of MWU with step-size $\eta > 0$ is*

$$\|\mathbf{x}^{t+1} - \Phi(\eta, \mathbf{x}^t)\|_\infty \leq 1 - e^{-2\eta}.$$

PROOF. Suppose without loss of generality that for any action i the expected payoff is bounded to $[-1, 1]$, i.e. $\sum_{j \in [n]} A_{i,j} \delta_j(\mathbf{y}) \in [-1, 1]$.[4] Let $W(t) = \sum_{j \in [n]} \exp(\mathbf{y}_j(t))$ and $W_i(t) = \exp(\mathbf{y}_i(t)) = x_i(t)W(t)$. Then continuous time RD becomes

$$\mathbf{x}_i(t) = \frac{W_i(t)}{W(t)}.$$

Similarly, define $\hat{W}^t = \sum_{j \in [n]} \exp(\mathbf{y}_j^{t-1})$ and $\hat{W}_i^t = \exp(\mathbf{y}_i^{t-1}) = x_i^t \hat{W}^t$. Then MWU becomes

$$\mathbf{x}_i^t = \frac{\hat{W}_i^t}{\hat{W}^t}.$$

We are interested in bounding the local error of MWU as a discretization of RD, i.e. the error introduced by a single step of MWU relative to RD after a single step starting from the same point. Thus without loss of generality we will focus on the first iterate of MWU and RD after $t = \eta$ amount of time. Since expected payoffs are bounded to $[-1, 1]$ we deduce from the analysis in Appendix C that

$$\hat{W}_i^1 \exp(-\eta) \leq W_i(\eta) \leq \hat{W}_i^1 \exp(\eta),$$

which implies

$$\hat{W}^1 \exp(-\eta) \leq W(\eta) \leq \hat{W}^1 \exp(\eta).$$

Hence

$$\mathbf{x}_i^1 \exp(-2\eta) \leq \mathbf{x}_i(\eta) \leq \mathbf{x}_i^1 \exp(2\eta)$$

whenever RD and MWU start from the same initial condition.

We have thus found that the local error introduced by a single time step is

$$\|\mathbf{x}^1 - \mathbf{x}(\eta)\| \leq \|\mathbf{x}^1 - \mathbf{x}^1 \exp(-2\eta)\| \leq |1 - \exp(-2\eta)|.$$

Observing that $\eta > 0$ gives the result. □

E PROOF OF LEMMA 2

LEMMA 2. *Let Φ be the flow generated by replicator dynamics and $\mathbf{x}^t$ be the mixed strategy found on the tth iterate of MWU. The error accrued after $t + 1$ iterations of MWU with step-size $\eta > 0$ is*

$$\|\mathbf{x}^{t+1} - \Phi(t\eta, \mathbf{x}^0)\| \leq O\left(e^t\right).$$

---

[4] The assumption that expected payoffs are bounded to $[-1, 1]$ does not affect learning dynamics since we can always normalize the payoff matrix by its largest element.

PROOF. The flow Φ is C^1 and Δ^n is compact, so we know that Φ is Lipschitz continuous. Let L denote the Lipschitz constant for Φ with respect to $\| \cdot \|_\infty$. It follows that for every initial condition $\mathbf{x}^0 \in \Delta^n$,

$$
\begin{aligned}
E^{t+1} &= \|\mathbf{x}^{t+1} - \Phi((t+1)\eta, \mathbf{x}^0)\| \\
&= \|\mathbf{x}^{t+1} - \Phi(\eta, \Phi(t\eta, \mathbf{x}^0))\| \\
&= \|\mathbf{x}^{t+1} - \Phi(\eta, \mathbf{x}^t) + \Phi(\eta, \mathbf{x}^t) - \Phi(\eta, \Phi(t\eta, \mathbf{x}^0))\| \\
&\leq \|\mathbf{x}^{t+1} - \Phi(\eta, \mathbf{x}^t)\| + \|\Phi(\eta, \mathbf{x}^t) - \Phi(\eta, \Phi(t\eta, \mathbf{x}^0))\| \\
&\leq |1 - \exp(-2\eta)| + e^{\eta L}\|\mathbf{x}^t - \Phi(t\eta, \mathbf{x}^0)\| \\
&= |1 - \exp(-2\eta)| + e^{\eta L} E^t
\end{aligned}
$$

To conclude our proof, we require a special case of the discrete Gronwall lemma. This powerful tool for numerical error analysis tells us that if, for some constants a and b with $a > 0$, a positive sequence $\{z^\tau\}_{\tau=0}^t$ satisfies

$$
z^{\tau+1} \leq b + az^\tau , \qquad \forall \tau \in [t-1] ,
$$

then for $a \neq 1$

$$
z^\tau \leq a^\tau z^0 + b\frac{a^\tau - 1}{a - 1} , \qquad \forall \tau \in [t] ,
$$

and for $a = 1$

$$
z^\tau \leq z^0 + \tau b , \qquad \forall \tau \in [t] .
$$

Recall that both $\eta > 0$ and $L > 0$ by definition. Let $z^t = E^t$, $a = e^{\eta L}$, and $b = |1 - \exp(-2\eta)|$. Applying the discrete Gronwall lemma yields

$$
E^{t+1} \leq e^{(t+1)\eta L} E^0 + |1 - \exp(-2\eta)|\frac{e^{(t+1)\eta L} - 1}{e^{\eta L} - 1} .
$$

Clearly $E^0 = 0$ since MWU and replicator dynamics have the same initial conditions. Thus we have shown

$$
E^{t+1} \leq |1 - \exp(-2\eta)|\frac{e^{(t+1)\eta L} - 1}{e^{\eta L} - 1} ,
$$

which concludes the proof. □