1. Introduction

In these lecture notes we will introduce Graph Dynamical Systems (GDS). For comparison, some basic examples of classical dynamical systems have been included. This is meant to illustrate some key differences in the techniques used to analyze the respective systems. Some examples of applications and mathematical models built on graph dynamical systems have been included as well. This set of notes is meant as an illustration of finite dynamical systems and some of their distinguishing features. It complements and adds to [1, Chap. 1] and also “lightens” up some of the notation used therein.

2. Finite Dynamical Systems – Background

Historically, the class of dynamical systems referred to as finite, discrete dynamical systems has not received anywhere near the research attention that continuous systems have. Apart from the age of the respective areas, there are of course many reasons for this, one of which is the very successful use of for example ordinary differential equations (ODEs) and partial differential equations (PDEs) as descriptive and analytical tools in science and engineering.

With the introduction of digital computers a powerful tool became available for the analysis of finite discrete dynamical systems. This led to an increased interest in their properties, and cellular automata (CA) received a lot of attention [2, 3]. This is a class of graph dynamical systems we will see a lot more of later in the course. As will become clear, the mathematical theory and tools used to analyze finite dynamical systems are generally quite different than the ones used to analyze classical dynamical systems. Rather than using, e.g. calculus and differential geometry, one will typically resort to number theory, algebra, combinatorics, graph theory and probability theory.

It is interesting to note that many mathematical models are now based on discrete dynamical systems rather than systems of ODEs or PDEs. An examples of such a system is the TRANSIMS urban traffic model that we will briefly return to in the introductory part of this course. This is a CA based model that has been used to study highly detailed traffic at the level of US cities (e.g. Portland, Dallas/Forth Worth, Washington DC). Although traffic can be modeled using systems of partial differential equations (PDEs), it is completely unrealistic to do that on the scale of a city. Even if it was possible, the PDE based models
would not be able to answer many of the questions that arise in practice, such as the tracking of individual travelers or cars. The use of these discrete models is becoming more and more common. You can read more about applications in [1, Chapter 1].

3. Classical Dynamical Systems

Classical dynamical systems are usually classified as discrete or continuous systems. Here discrete and continuous refers to the time evolution of the respective systems. Below are basic examples of both classes. More details on this material can be found in [1, Chapter 2].

3.1. Continuous Dynamical Systems. Continuous dynamical systems typically arise from a system of ordinary differential equations (ODEs). In two dimensions it could look like

\[
\begin{align*}
x' &= f(x, y) \\
y' &= g(x, y)
\end{align*}
\]

where \((x, y) \in E \subset \mathbb{R}^2\) where \(E\) is an open subset of \(\mathbb{R}^2\) and \(f\) and \(g\) are smooth functions. Here we may want to derive properties of solution curves and of the phase space in general.

**Example 1.** On the left in Figure 1 we have shown some of the solution curves for the two-dimensional system given by

\[
\begin{align*}
x' &= x^2 + xy, \\
y' &= \frac{1}{2}y^2 + xy.
\end{align*}
\]

On the right we have shown some of the solution curves for the Hamiltonian system (see e.g. [4–6])

\[
\begin{align*}
x' &= y, \\
y' &= x + x^2.
\end{align*}
\]

To illustrate how such systems can be analyzed, let us find the fixed points (also called equilibrium points or stationary points) of the second system and study their stability. In this case, the function \(f\) and \(g\) are given by \(f(x, y) = y\) and \(g(x, y) = x + x^2\). The equilibrium points are those points where \(f\) and \(g\) are simultaneously zero, and we see that this happens at \((0, 0)\) and \((-1, 0)\). The Jacobian matrix of the system is

\[
(1) \quad J(x, y) = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 + 2x & 0 \end{pmatrix},
\]

so that \(J(0, 0) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\). Since \(J(0, 0)\) has eigenvalues \(\lambda = -1\) and \(\lambda = 1\) we can (through the Hartman-Grobman Theorem [6]) determine the behavior near \((0, 0)\) from the linear system

\[
(2) \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = J(0, 0) \begin{pmatrix} x \\ y \end{pmatrix}
\]

near \((0, 0)\). Since the eigenvalue/eigenvector pairs of \(J(0, 0)\) are \((-1, \begin{pmatrix} 1 \\ -1 \end{pmatrix})\) and \((1, \begin{pmatrix} 1 \\ 1 \end{pmatrix})\), we conclude that \((0, 0)\) is an unstable saddle point, which Figure 1 also suggests. \(\square\)

Problem 1. What can you say about the stability of the fixed point \((-1, 0)\)?
3.2. (Classical) Discrete Dynamical Systems. The dynamics of discrete dynamical systems result through iterated application of some map $F$. Starting at the system state $x_0$ at time $t = 0$, we obtain the system state at time $t = 1$ as $x_1 = F(x_0)$. The system state at time $t = 2$ is $x_2 = F(x_1) = F^2(x_0)$, and so on. In general we have
\[ x_n = F^n(x_0) \quad \text{with} \quad n \geq 0. \]

We obtain the (forward) orbit of $F$ through $x$ as
\[ \mathcal{O}(x) = \{ x, F(x), F^2(x), F^3(x), \ldots \} . \]

When $F$ is invertible, we can also consider negative values of $n$, in which case we may define the full orbit
\[ \mathcal{O}(x) = \{ \ldots, F^{-2}(x), F^{-1}(x), x, F(x), F^2(x), \ldots \} . \]

The setting of discrete dynamical systems should be a familiar for those of you who have worked with fractals and for example the Mandelbrot set. In the case of the Mandelbrot set the map often investigated is $F_c: \mathbb{C} \rightarrow \mathbb{C}$ defined by $F(z) = z^2 + c$ where $c \in \mathbb{C}$ is a parameter.

Example 2. Consider the map $F: \mathbb{R} \rightarrow \mathbb{R}$ given by $F(x) = 4x(1 - x)$. It is a special case of the logistic map studied in for example [7]. We see that $F$ has fixed points [a fixed point is a point $x$ for which $F(x) = x$] $x = 0$ and $x = 3/4$. Since the linearized system at both these points have eigenvalues whose magnitude is different than 1 we can use the Hartman-Grobman Theorem to conclude that both fixed points are unstable. For the record: $F'(0) = 4$ and $F'(3/4) = -2$.

Some typical topics studied for classical dynamical systems include:

- Fixed points and periodic orbits.
- Stability of fixed points and orbits.
- Dependence on parameters and bifurcations.
- Dependence on initial conditions.
Structural stability.

The purpose of this section was to give a quick illustration of some of the mathematics used when studying classical dynamical systems. Calculus and differential geometry are central, and one usually assumes that the map $F$ is continuous. In the case of systems of ODEs it is often required that the vector field is continuously differentiable. Although one may define a derivative for finite dynamical system, it is not (at least for now) nearly as powerful as its continuous counterpart. As you will see, the techniques used to analyze finite dynamical systems tend to rely much more on algebra, combinatorics, graph theory, and number theory.

4. Graph Dynamical Systems

Since the text book [1] was written, there has been some changes in notation. The old notation is a little “heavy” and this is an attempt to remedy that. We will use Chapter 4 in [1] as the introduction to discrete dynamical systems, but the changes to notation will be given here along with additional comments and some new examples. We start out with the general notion of a graph dynamical system and then specialize to the cases of sequential dynamical systems (SDS) and generalized cellular automata (GCA).

4.1. Graph Dynamical Systems. Let $X$ be a finite, undirected graph with vertex set

$$v[X] = \{1, 2, \ldots, n\},$$

and edge set denoted by $e[X]$. Will will refer to the graph $X$ as the dependency graph.

**Example 3** (The Circle Graph on $n$ vertices, $\text{Circle}_n$). The Circle graph on $n$ vertices (sometimes called the cycle graph or just $n$-cycle), is the graph $\text{Circle}_n$ given by

$$v[\text{Circle}_n] = \{1, 2, \ldots, n\}$$

$$e[\text{Circle}_n] = \{\{i, i+1\} \mid 1 \leq i \leq n\},$$

where we identify indices modulo $n$. The graph $\text{Circle}_4$ is displayed in Figure 2.

![Figure 2. The circle graph on 4 vertices.](image)

To each vertex $v$ of $X$ we associate a state $x_v \in K$ where $K$ is some finite set. We refer to $x_v$ as the vertex state and

$$x = (x_1, x_2, \ldots, x_n)$$

as the system state.
Example 4 (Example 3 continued). One common choice for vertex state set is \( K = \{0, 1\} \). This particular case is sometimes referred to as the Boolean case, and we speak of binary states. Occasionally, we may view \( K = \{0, 1\} \) as the field with two elements with addition and multiplication modulo 2. In this course we will focus on the case where \( K \) is finite, but several of the results hold for \( K = \mathbb{R} \) or \( K = \mathbb{C} \).

The 1-neighborhood of \( v \) in \( X \) is denoted by \( N(v; X) \) and is the set containing the vertex \( v \) and all those vertices in \( X \) connected to \( v \). We write \( n[v] \) for the sequence of elements from \( N(v; X) \) ordered in increasing order and call it the sorted neighborhood of \( v \). We also introduce the notation \( x[v] \), the restriction of \( x \) to \( n[v] \), that is,

\[
x[v] = (x_{n[v](1)}, x_{n[v](2)}, \ldots, x_{n[v](d(v)+1)})
\]

where \( d(v) \) is the degree of \( v \).

Example 5 (Example 3 continued). For the graph \( X = \text{Circle}_4 \) every vertex has degree 2. Here \( n[1] = (1, 2, 4) \) and \( x[4] = (x_1, x_3, x_4) \). What is \( x[3] \)?

Each vertex \( v \) is assigned a vertex function \( f_v \) of the form \( f_v: K^{d(v)+1} \rightarrow K \). This function takes as argument the restricted state \( x[v] \) at time \( t \) and returns the state of \( v \) at time \( t+1 \).

It is convenient to introduce the \( X \)-local function \( F_v: K^n \rightarrow K^n \) defined by

\[
F_v(x_1, x_2, \ldots, x_n) = (x_1, \ldots, f_v(x[v]), \ldots, x_n).
\]

Whether we use the vertex function \( f_v \) or the \( X \)-local function \( F_v \) depends on the context.

Example 6 (Example 3 continued). A particular Boolean function that we will use throughout this course is the nor-function (that is, “not or”). Here \( K = \mathbb{F}_2 = \{0, 1\} \) is the field of two elements, and the nor-function on \( m \) variables \( \text{nor}_m: K^m \rightarrow K \) is given by

\[
(3) \quad \text{nor}_m(x_1, x_2, \ldots, x_m) = \prod_{i=1}^{m} (1 + x_i).
\]

Here we see that the only case in which \( \text{nor}_m \) evaluates to 1 is when all its arguments are 0. The nor-function is an example of a symmetric function.

Taking \( X = \text{Circle}_4 \) once more, we use the nor_3-function as vertex function for each vertex. The corresponding \( X \)-local functions, which we denote by \( \text{Nor}_i \), are given as \( \text{Nor}_i: K^4 \rightarrow K^4 \) for \( 1 \leq i \leq 4 \) where (as you should verify)

\[
\begin{align*}
\text{Nor}_1(x_1, x_2, x_3, x_4) &= (\text{nor}_3(x_4, x_1, x_2), x_2, x_3, x_4), \\
\text{Nor}_2(x_2, x_2, x_3, x_4) &= (x_1, \text{nor}_3(x_1, x_2, x_3), x_3, x_4), \\
\text{Nor}_3(x_3, x_2, x_3, x_4) &= (x_1, x_2, \text{nor}_3(x_2, x_3, x_4), x_4), \\
\text{Nor}_4(x_4, x_2, x_3, x_4) &= (x_1, x_2, x_3, \text{nor}_3(x_1, x_3, x_4)).
\end{align*}
\]

Again, note that each \( X \)-local function \( F_i \) may only change the state of its own vertex \( i \), and it does so based solely on the state of vertex \( i \) and the states of the neighbors of \( i \) in the graph \( X \).

The vertex function \( f_v \) (or alternatively, the \( X \)-local function \( F_i \)) allows us to transition the state of vertex \( v \) from time \( t \) to time \( t+1 \) in isolation. An update mechanism governs how the collection of vertex functions, or alternatively, the \( X \)-local functions, assemble to the global
dynamical system map. Two choices for update mechanisms that will be used extensively in this course are parallel and sequential application of vertex functions. This leads to the two sub-classes of GDS called generalized cellular automata (GCA) and sequential dynamical systems (SDS), respectively. These cases are described in the next two sections.

Most of the concepts above are illustrated in Figure 3.

![Figure 3. The basic constituents in a GDS.](image)

4.2. **Sequential Dynamical Systems.** Let \( \pi = (\pi_1, \pi_2, \ldots, \pi_n) \) be a permutation of the vertex set of \( X \). We write \( S_X \) for the collection of all such permutations. In the context of SDS, we refer to \( \pi \) as the update sequence since it is used to specify the order in which we apply the \( X \)-local functions to construct the global GDS map. Specifically, with \( X \)-local functions as above and update sequence \( \pi \), the GDS map is given by

\[
F_{\pi} = F_{\pi(n)} \circ \cdots \circ F_{\pi(2)} \circ F_{\pi(1)}.
\]

**Example 7.** Returning to the case with \( X = \text{Circle}_4 \), binary states, and where each vertex function is given by the nor-function, this is how the dynamics arise in the sequential case. We use \( \pi = (1, 2, 3, 4) \) here. By applying the four maps \( \text{Nor}_i \) to for example the state \( x = (x_1, x_2, x_3, x_4) = (1, 1, 0, 0) \) in the order given by \( \pi \) we get (as you should verify)

\[
(1, 1, 0, 0) \xrightarrow{\text{Nor}_1} (0, 1, 0, 0) \xrightarrow{\text{Nor}_2} (0, 0, 0, 0) \xrightarrow{\text{Nor}_3} (0, 0, 1, 0) \xrightarrow{\text{Nor}_4} (0, 0, 1, 0).
\]

This shows that the system state \((1, 1, 0, 0)\) is mapped to the system state \((0, 0, 1, 0)\) using \( \text{Nor}_\pi \), which we write as \( \text{Nor}_\pi(1, 1, 0, 0) = (0, 0, 1, 0) \). What is \( \text{Nor}_\pi(1, 0, 1, 0) \)? The **phase space** (formally defined a little later) is shown in Figure 4.

In general one may use a word over the vertex set of the graph \( X \) to specify the update sequence. We will use Greek letters like \( \pi \) and \( \sigma \) to denote permutation update sequences whereas words are denoted \( w, w' \) and so on. The set of words over \( v[X] \) is denoted by \( W_X \).

**Problem 2.** Looking at the phase space in Figure 4, you will notice that the cycle has length \( > 1 \). The states on cycles are called periodic points. A state on a cycle of length one is called a fixed point: these are the states that are mapped to themselves. Can a Nor-SDS over the graph \( \text{Circle}_n \) have fixed points? Are there graphs where a Nor-SDS may have fixed points?
4.3. **Generalized Cellular Automata.** Generalized cellular automata differ from SDS in the update mechanism. For GCA (or maybe parallel dynamical systems (PDS) would be a better name), the vertex functions are applied in parallel. In this case, the GDS map $F : K^n \rightarrow K^n$ is given by

$$F(x_1, x_2, \ldots, x_n) = (f_1(x[1]), f_2(x[2]), \ldots, f_n(x[n])).$$

**Example 8.** Returning to the previous example with $X = \text{Circle}_4$, binary states, and vertex function given by the nor-function, you may verify that

$$F(1, 1, 0, 0) = (0, 0, 0, 0).$$

The phase space of this system is shown in Figure 5.

![Figure 4. The phase space of the SDS-map Nor\(_\pi\).](image1)

![Figure 5. The phase space of the GCA-map Nor over X = Circle\(_n\).](image2)

**Problem 3.** Looking at the phase space in Figure 5, describe the (system) states that are *not* mapped to $(0, 0, 0, 0)$. Can you generalize this to Circle\(_n\)?

4.4. **Comments Regarding the Definitions.** From the point of view of applications, the graph is meant to capture a system of interacting entities by matching entities to vertices and interactions between entities with corresponding edges. Vertex functions encode the
behavioral properties of the matching entities, and the update mechanism reflects the way in which the system evolves.

**Directed versus undirected dependency graph.** In the definitions above we have taken the graph \( X \) to be undirected. Since the dependency graph is meant to reflect dependencies (!) among vertices and their vertex functions, one may want to be more accurate and use a directed dependency graph. If vertex function \( f_i \) depends on the state of vertex \( j \) in a non-trivial way, one would insert the directed edge \((i, j)\) into the dependency graph \( j \). Many of the combinatorial results we will encounter are not affected by this choice. However, in other contexts this distinction may be beneficial or necessary for the analysis.

**Self-loops in the dependency graph.** By the reasoning in the previous paragraph, if vertex function \( f_i \) depends on state \( x_i \) in a non-trivial way, then there should be an edge from \( i \) to \( i \) in the dependency graph. An edge of the form \( \{i, i\} \) is called a **self-loop** or just a **loop**. While this would certainly be more consistent it would lead to complications that will become evident later in the course when studying acyclic orientations of the dependency graph. We therefore make the implicit assumption that \( f_v \) may always depend on vertex state \( x_v \).

**The role of the dependency graph.** The dependency graph \( X \) may be regarded as redundant in the following sense. If we are given a list of vertex functions \( f = (f_1, \ldots, f_n) \) one may deduce the edges in the dependency graph by investigating which vertex states affect each such function. While this offer a certain level mathematical elegance and minimality, it is nevertheless a fact in e.g. applications and modeling it is the dependency graph that is the typical starting point. It may also be useful to include \( X \) explicitly when working with for example function stochastic system. It will not make a difference for us. Note that the dependency graph is sometimes referred to as the **wiring diagram** of the system.

**Other graph types: multi-graphs, hyper-graph.** In general, one may also operate with more general graphs. In a multi-graph, each pair of vertices may be joined by multiple edges. In a hyper-graph, an edge may join more than two vertices. One may also consider assigning states to edges that also evolve with time depending on **edge functions**. As always, one will have to strike a balance between system “richness” and the possibility of obtaining useful and interesting results. Ultimately, it is desirable to have a theory for such general system, but as you probably will soon realize, the basic versions introduced are already quite complex.

**About vertex state spaces.** In general, one may assign each vertex its own vertex state space \( K_i \). It is obvious how to modify the definitions above, so in the interest of easier book keeping (or laziness) we skip this.

4.5. **Some More Definitions and Terminology.** Let \( \phi : K^n \rightarrow K^n \) be a GDS. We call the application of the map \( F_v \) to \( x \) a **vertex update** of the state \( x_v \), and the application of \( \phi \) to \( x \) a **system update**.

The **phase space** of \( \phi \) is the directed graph \( \Gamma(\phi) \) given by

\[
\begin{align*}
\mathcal{V}[\Gamma(\phi)] &= \{x \in K^n\} \\
\mathcal{E}[\Gamma(\phi)] &= \{(x, \phi(x)) \mid x \in K^n\}.
\end{align*}
\]
Since the dependency graph is finite and the vertex state space is finite, this graph is also finite. This graph describes all of the dynamics of $\phi$.

**Question:** Describe the general structure of a GDS phase space. How many vertices does the graph have? How many edges? What is the component structure?

Since the order of the phase space is exponential in the order of the dependency graph, it is usually impossible explicitly compute this. For a given state $x \in K^n$ we define the *forward orbit* through $x$ under $\phi$ as the sequence

$$O^+(x) = \{x, \phi(x), \phi^2(x), \ldots\}$$

where $\phi^k(x)$ with $k > 0$ is defined recursively by $\phi^k(x) = \phi(\phi^{k-1}(x))$ and $\phi^0(x) = x$. Loosely speaking, $\phi^k(x)$ means $\phi$ applied $k$ times to $x$. When $\phi$ is invertible, one may also consider the iterates of $\phi^{-1}$ and define the *full orbit* of $x$ as

$$O(x) = \{\ldots, \phi^{-2}(x), \phi^{-1}(x), x, \phi(x), \phi^2(x), \ldots\}.$$

An orbit may be visualized in a space time diagram, see [1] for definitions and illustrations.

A state $x \in K^n$ is a *fixed point* for $\phi$ if $\phi(x) = x$. This is represented by a loop in phase space. Similarly, $x$ is a *periodic point* of $\phi$ if there exists an integer $k > 0$ such that $\phi^k(x) = x$. The smallest positive integer for which this holds is called the (prime) *period* of $x$. In the phase space, periodic points are precisely those points contained in cycles. Note that a fixed point is also a periodic point. A point that is not a periodic point is a *transient point*.

### 4.6. A Note on Questions We Will Address.

Here is a list of some of the questions we will address in this course for the analysis of GDS and their properties:

- How many fixed points does the GDS $\phi$ have?
- How many periodic points does $\phi$ have?
- Can the periodic points be characterized?
- When is $\phi$ invertible and its dynamics “reversible”?
- Are periodic points stable?
- How robust is an orbit with respect to perturbation of the initial state?
- If we have two permutations $\pi$ and $\sigma$ of $v[X]$, when is $F_\sigma = F_\pi$?
- When do $F_\sigma$ and $F_\pi$ have equivalent periodic orbits?
- How many periodic orbit structures can we generate by changing the update sequence?
- What is meant by a stochastic GDS, and how can we analyze such systems?

Realistically, one will not be able to compute phase space exhaustively. To answer these questions, one will have to use properties of the defining elements to make these deductions (graph properties, function properties and so on). Since the global dynamics is generated by composition of local dynamics the resulting analysis often have a local-to-global character. We will address all of these questions and many others in the course.

We close this section by summarizing some of the differences between the graph dynamical systems we will consider in this course and the classical systems; the table below list distinguishing features.
<table>
<thead>
<tr>
<th>System class</th>
<th>Time evolution</th>
<th>State space</th>
</tr>
</thead>
<tbody>
<tr>
<td>System of ODEs</td>
<td>Continuous</td>
<td>Continuous</td>
</tr>
<tr>
<td>(Classical) Discrete dynamical systems</td>
<td>Discrete</td>
<td>Continuous</td>
</tr>
<tr>
<td>SDS &amp; GCA</td>
<td>Discrete</td>
<td>Discrete</td>
</tr>
</tbody>
</table>

**REFERENCES**


