BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION [Secție]

DIPLOMA THESIS

[Titlu lucrare]

Supervisor
[Grad, Lorand Gabriel Parajdi]

Author
Stan Ioan-Victor

2025

ABSTRACT

Abstract: un abstract in engleza one in romanian un rezumat în limba engleză

cu prezentarea, pe scurt, a conținutului pe capitole, punând accent pe contribuțiile proprii și originalitate

Contents

Chapter 1

Introduction

Introducere: obiectivele lucrării și descrierea succintă a capitolelor, prezentarea temei, prezentarea contribuției proprii, respectiv a rezultatelor originale și menționarea (dacă este cazul) a sesiunii de comunicări unde a fost prezentată sau a revistei unde a fost publicată.

Chapter 2

Introduction to Chemical Reaction Networks

Chapter 3

Dynamical Systems

3.1 Defining Ordinary Differential Equations

Definition 1. *An **ordinary differential equation** is an equation of an unknown function of **one** variable. This can be expressed as a function **of** this unknown function, its corresponding variable and its various derivatives.*

Its general form looks something like:

$$F(t, y(t), y'(t), ..., y^{(n)}(t)) = 0, \tag{3.1}$$

Where $y(t)$ is the unknown function of independent variable t and $F : \Omega \to \mathbb{R},\ \Omega \subseteq \mathbb{R}^{n+1}$.
This would be what's called the implicit form of the differential equation.

Definition 2. *The **order** of an ODE is the highest order of the derivative present in the equation.*

In our case, the order is n. If, however F satisfies the regularity condition **of the implicit function theorem** then the equation can be written in a much more digestible form.

The **implicit function theorem** allows one to convert a relation (implicit form) to functions of some variables. These functions may not be unique, but togather, their graph may locally satisfy the relation.
As an example: The relation for a cardiod cannot be expressed as a sole function, we'd instead need the union of the graph of two separate functions for expressing it. (si aici gen faci tu niste graphuri cu maple sau alte d-astea vezi cum faci virtual machineu ala sau mergi la faculta la un pc cu maple si iti faci lista cu ce vrei sa graphuiesti)

Reminder: A function $f(x)$ is continuously differentiable if $\exists f'(x)$ and $f'(x) \in C^n$ where $n >= 0$

Theorem 1. Let $F : D \subseteq \mathbb{R}^{n+1} \to \mathbb{R}$ be a continuously differentiale function with the relation that $F(t, y(t), y'(t), ..., y^{(n)}(t)) = 0$. Let us express a point in the set $\mathbb{R}^{n+1} = \mathbb{R} \times \mathbb{R}^n$ as (t, **Y**) = $(t, y, y', \ldots, y^{(n)})$ and fix one such point s.t. $F(t, \mathbf{Y}) = 0$. So $\exists \square_{(t,\mathbf{Y})} \ni U \times V \subseteq D$ s.t. $F \in C^1(U \times V)$ and $\frac{\partial F}{\partial y}(t, \mathbf{Y}) \neq 0$. Then this means $\exists \square_t \ni U_0 \subseteq U, \square_{\mathbf{Y}} \ni V_0 \subseteq V$ and a function $f : U_0 \to V_0$ s.t. $f(t) = \mathbf{Y}$ and $F(t, f(t)) = 0, \forall t \in U_0; f \in C^1(U_0)$ and $\frac{\partial f}{\partial t_i} = -\frac{\frac{\partial F}{\partial t_i}(t, f(t))}{\frac{\partial F}{\partial y}(t, f(t))}, \forall i = \overline{1, n}, \forall t \in U_0$ and $F \in C^1(U \times V), K \in N^* \Rightarrow f \in C^K(U_0)$.

So restricting the domain Ω to one that allows the implicit form to be represented as a function of

the form

$$y^{(n)}(t) = f(t, y(t), y'(t), ..., y^{(n-1)}(t)) \tag{3.2}$$

would yield what's called the **explicit** form of the ODE.

A **solution** is an expression of the unknown function $y(t)$ which satisfies the relation $y^{(n)}(t) = f(t, y(t), y'(t), \ldots, y^{(n-1)}(t))$. A **general solution** includes all of these functions and usually has some constants of integration in the expression, while a **particular solution** doesn't have them. A particular solution is usually found if we also have some initial conditions given for the unknown function, like $y(a) = b, y'(a) = c$ for some $a, b, c \in \mathbb{R}$. Or, more formally:

Definition 3. A function $y : I \to \mathbb{R}$ is a solution of the equation **??** if the following conditions are met:

1. $I \subseteq \mathbb{R}$ is nondegeneate interval. ($|I| > 1$)
2. $y \in C^n(I)$ and $(t, y(t), y'(t), \ldots, y^{(n)}(t)) \in D_f, \forall t \in I$.
3. $y^{(n)}(t) = f(t, y(t), y'(t), \ldots, y^{(n-1)}(t)), \forall t \in I$. (**??** is satisfied)

3.2 Forming Ordinary Differential Equation systems

We may take a sequence of such first order ordinary differential equations to form a system:

$$\{ y'_1(t) = f_1(t, y_1(t), y_2(t), \ldots, y_n(t)) y'_2(t) = f_2(t, y_1(t), y_2(t), \ldots, y_n(t)) \vdots y'_n(t) = f_n(t, y_1(t), y_2(t), \ldots, y_n(t)) \tag{3.3}$$

Constructing $y : D \subseteq \mathbb{R} \to \mathbb{R}^n, f : D_f \subseteq \mathbb{R}^{n+1} \to \mathbb{R}^n$ and denoting $y(t) = (y_1(t), \ldots, y_n(t))$ and $f(t, y_1(t), \ldots, y_n(t)) = (f_1, \ldots, f_n)$ we get what is called the **vector form** of the system **??**.

$$y'(t) = f(t, y(t)) \tag{3.4}$$

Definition 4.

For a function $y : D \subseteq \mathbb{R} \to \mathbb{R}^n$, $|D| > 1$, if $y \in C^1(D, \mathbb{R}^n)$, and $y'(t) = f(t, y(t)), \forall t \in D \Rightarrow y$ is a **solution of the system ??**.

Definition 5. An **autonomous equation** is a differential equation that does not explicitly depend on time, but is instead only defined by the relation between the function itself and its derivatives, so an equation of the form

$$f(y^{(n)}(t), y^{(n-1)}(t), \ldots, y(t)) = 0 \tag{3.5}$$

That is not to say, the function itself does not depend on time, but rather the independent variable t does not explicitly appear in the equation.

One more famous such example is the equation for the damped oscillator

$$\ddot{x} + 2\gamma\dot{x} + \omega^2 x = 0. \tag{3.6}$$

Where

- γ is the damping factor, meaning how quickly the oscillations of the system decay due to loss of momentum (caused by air resistance or friction)

- ω represents the angular (natural) frequency the oscillator would have, were it not to be affected by damping, defined by $\omega = \sqrt{\frac{k}{m}}$; where k is the **elastic stiffness coefficient** of the spring and m the mass.

In addition, a frequently used notation in physics is $\dot{x} = \frac{d}{dt}x$, $\ddot{x} = \frac{d^2}{dt^2}x$

That equation is a **linear** homogenous equation with constant coefficients (hence autonomous), which is pretty nice to work with. Their general form would be:

$$x + a_1\dot{x} + \cdots + a_n x^{(n)} = 0 \tag{3.7}$$

Finding exact solutions, though, can be done by using some nice properties of Euler's exponential function e^{rt}.

Introducing the notion of **operators** would help in writing this next section.

Definition 6. $D = \frac{d}{dt}$ is the **derivation operator**: $D^n = \frac{d^n}{dt^n}$, so $D^n u = D^n(u) = \frac{d^n u}{dt^n}$ where $u = u(t)$ and (related to **??**):
$L = 1 + a_1 D + \cdots + a_n D^n$
is the **linear differential operator of order n**. Meaning **??** $\iff Lx = 0$.

We now can; just by pure Ansatz; assume the solutions take the form e^{rt} for some (yet unknown) r's. Because the exponential is so easy to work with when it comes to taking derivatives - a.k.a. : $D^n(e^{rt}) = r^n e^{rt}$, so its derivatives are proportional to itself, meaning no new t's are introduced - substituting back into **??** yields Le$^{rt} = e^{rt} + a_1 De^{rt} + \cdots + a_n D^n e^{rt} = 0$
$\Updownarrow$
$e^{rt} + a_1 re^{rt} + \cdots + a_n r^n e^{rt} = 0$
$\Updownarrow$
$1 + a_1 r + \cdots + a_n r^n = 0$ The left-hand-side is the **characteristic polynomial** of the linear equation.

$$p(r) = 1 + a_1 r + \cdots + a_n r^n$$

Notice that $L = p(D)$.
So now it all comes down to finding its roots r.

Case I: A **unique, real** root found:
now e^{rt} is a **fundamental solution** for $Le^{rt} = 0$

Case II: A **unique, complex** root found:
say it has the form $r = \lambda \pm i\mu, \mu \neq 0$.
By Euler's formula:

$$e^{rt} = e^{(\lambda+i\mu)t} = e^{\lambda t} \cdot e^{i\mu t} = e^{\lambda t} \cdot [\cos(\mu t) + i\sin(\mu t)] \tag{3.8}$$

So $e^{\lambda t}\cos(\mu t)$ and $e^{\lambda t}\sin(\mu t)$ are both fundamental solutions for $Lx = 0$.

Proof:
Since $a_i \in \mathbb{R}, \forall i = \overline{1,n}$ in $p \implies p(\overline{r}) = \overline{p(r)}, \forall r \in \mathbb{C}$, hence, just like 2AM coffee and the exam session, complex roots of p always come in pairs. They both satisfy:

$$Le^{(\lambda \pm i\mu)} = 0 \tag{3.9}$$

But since our $x : \mathbb{R} \to \mathbb{R}$ we're looking for **real** valued solutions. Because $\mu \in \mathbb{R}$ the following properties hold equivalently for $\lambda + i\mu$ and for $\overline{\lambda + i\mu} = \lambda - i\mu$ so I'll only write λ+$i\mu$ for simplicity. By **??**:

$$\text{Re}(e^{rt}) = e^{\lambda t} \cdot \cos(\mu t), \qquad \text{Im}(e^{rt}) = e^{\lambda t} \cdot \sin(\mu t). \tag{3.10}$$

But (**reminder**) Re(c)= c+$\overline{c}$ $\overline{2, \qquad \text{Im}(c) = \frac{c-\overline{c}}{2i}}$ $\forall c \in \mathbb{C}$

Using these three conclusions we get: L(e$^{\lambda t}$·cos(μt)) $\overset{\textbf{??, ??}}{=\!=\!=} L\left(\frac{e^{rt}+e^{\overline{r}t}}{2}\right) \overset{\text{dist.}}{=\!=} \frac{1}{2}\left(Le^{rt} + Le^{\overline{r}t}\right) \overset{\textbf{??}}{=\!=}$ 0,
$L(e^{\lambda t} \cdot \sin(\mu t)) \overset{\textbf{??, ??}}{=\!=\!=} L\left(\frac{e^{rt}-e^{\overline{r}t}}{2i}\right) \overset{\text{dist.}}{=\!=} \frac{1}{2i}\left(Le^{rt} - Le^{\overline{r}t}\right) \overset{\textbf{??}}{=\!=} 0,$

In conclusion,if $Le^c = 0$ for some $c \in \mathbb{C} \implies L\text{Re}(e^c) = L\text{Im}(e^c) = 0$, for $\text{Re}(e^c), \text{Im}(e^c) \in \mathbb{R}$. Moreover, the same can be said for $e^{\overline{c}}$. □

Now something let's say ... **fundamental** (hehe) about these solutions, is they have to be linearly independent to eachother. Meaning the only solution for

$$c_1x_1 + c_2x_2 + \cdots + c^nx^n = 0 \tag{3.11}$$

is the trivial solution $c_1 = c_2 = \cdots = c_n = 0$ where x_i are the fundamental solutions of the system.

So the reson I've emphasized "**unique**" in the first 2 cases was because If we're to apply those methods for **repeating** roots, meaning the ones with multiplicity $k > 1$, the resulting solutions wouldn't hold **??** because they would in fact be the same!

To counteract this issue, the clever people over at the 1748 Berlin Academy (just Euler) came up with yet another (easily provable) Ansatz: just multiplying by powers of x.

A root with multiplicity k would introduce yet another k -for real- or $2k$ - for complex - fundamental solutions to this equation, so to finish what we've started:

Case III: **Repeated, real** roots r with multiplicity $k > 1$ yield the fundamental solutions
$e^{rt}, xe^{rt}, \ldots, x^{k-1}e^{rt}$.
These can be verified that they do, in fact satisfy **??**. So can:

Case IV: **Repeated, complex** roots r with multiplicity $k > 1$. These introduce the pairs: e^{at}cos(bt), e^{at}sin(bt)
xe^{at}cos(bt), xe^{at}sin(bt)
⋮

$x^{k-1}e^{at}\cos(bt),\ \ x^{k-1}e^{at}\sin(bt)$

So $2k$ new solutions.

The superposition principle

Bernoulli (to the disbelief of Euler and Langrange) told us that the complex behaviour of vibrating strings can be modeled by sepparating their motion into well-defined and easy to compute simple waves with known frequences and amplitudes, which can later be **superposed** with one-another to form the complete system.

This principle can be applied to *our* easier to compute **fundamental solutions** $(x_1, \ldots, x_n)$ in relation to the system formed by our linear operator $Lx = 0$.

Since it's a linear operator, additivity and homogeneity apply: L($x_1 + x_2$) $= Lx_1 + Lx_2$
$aLx = L(ax), \forall a \in \mathbb{C}$ where x, x_1, x_2 are fundamental solutions of the system.
Hence the complete, final, **general** solution for **??** are linear combinations of all our previously found fundamental solutions:

$$x(t) = c_1x_1 + \cdots + c_nx_n$$

for some constants c_i.

3.3 Formally defining and characterizing dynamical systems

The reason why autonomous equations are so useful to us is because of the ease with which we can visualize the **evolution** of a system of such equations. Take, for example:

Definition 7. A **system of first-order autonomous equations** is a system of equations of the form

$$\{\dot{} y_1 = f_1(y_1, \ldots, y_n)\dot{y}_2 = f_2(y_1, \ldots, y_n)\vdots\dot{y}_n = f_n(y_1, \ldots, y_n) \tag{3.12}$$

Definition 8. We can construct the **phase space** of the **??** dynamical system by:

1. Representing all possible values of y_i, $i = \overline{1, n}$ as points in Euclidean space $S \subseteq \mathbb{R}^n$
2. Constructing a function $f : S \to S, f(y_1, y_2, \ldots, y_n) = (\dot{y}_1, \dot{y}_2, \ldots, \dot{y}_n)$ that attaches to each point in S a vector representing the direction in which to go such that the relation **??** is satisfied.

Thus, the **phase space** of the system is a vector field where f is a continuous vector valued function.

So keeping this in mind, we may form a formal definition:

Definition 9. A **dynamical system** is a tuple (T, S, Φ) where:

- T: part of the $(T, +)$ monoid of all possible $time$ moments of our system
- S: is the set of all possible states,and
- the function: $\Phi : U \subseteq (T \times S) \to S$ where values of the second component span all of S, meaning $proj_2(U) = S$.

In order to give the necessary properties of Φ we will need to also define:
$I(s) := \{t \in T | (t, s) \in U\}, \forall s \in S$

so Φ has the properties that:
$\Phi(0, s) = s$
$\Phi(t_2, \Phi(t_1, s)) = \Phi(t_1 + t_2, s); \forall t_1, t_1 + t_2 \in I(s), t_2 \in I(\Phi(t_1, s)), s \in S.$
We call such a function Φ the **evolution function** of the dynamical system, mapping every state and $time$ after which that specific state is found, to another unique state.

If we take a specific state s as an initial constant, we can form the function
$\Phi_s : I(s) \to S$
called the **flow** through s, whose image:
$\gamma_s := \{\Phi_s(t) | t \in I(s)\}$
is the **orbit** through s, which is a curve of parameter $t \in I(s)$ representing the values the dynamical system will take throughout the experiment's timespan, given the initial condition (state) s.

Since a point only belongs to one orbit, if we form the set
$\Gamma_{U,\Phi} := \{\gamma_s | s \in S\}$
of all orbits of the dynamical system (U, Φ), with $U \subseteq (T \times S)$ and Φ the evolution function, it follows that:
$\forall \gamma_1, \gamma_2 \in \Gamma_{U,\Phi}, \gamma_1 \bigcap \gamma_2 = \varnothing$, meaning the set of orbits forms a partition of the phase space S

Proof: Conditions for partitions:
Reminder: A partition Γ_U of a set U is a set of non-empty disjoint subsets of U s.t.

$$\bigcup_{\gamma \in \Gamma_U} \gamma = U$$

So we need to prove:

TODO: te apuci de asta mai incolo
I. $\gamma \subseteq U, \forall \gamma \in \Gamma_{U,\Phi}$
II. $\gamma \neq \varnothing, \forall \gamma \in \Gamma_{U,\Phi}$
III. $\gamma_1 \bigcap \gamma_2 = \varnothing, \forall \gamma_1, \gamma_2 \in \Gamma_{U,\Phi}$
IV. $\bigcup_{\gamma \in \Gamma_U} \gamma = U$

TODO: oki, deci aici zici cum se rezolva o ecuatie autonoma liniara in mod normal si despre solutiile fundamentale si independenta si dependa lor and stuff si dupa dai un mod echivalnt de a le rezolva cu metoda de mai jos care face un sistem din ele
We may, in fact construct an equivalent system

$$\{\dot{} y_1 = f_1(y_1, \ldots, y_n) \dot{y}_2 = f_2(y_1, \ldots, y_n) \vdots \dot{y}_n = f_n(y_1, \ldots, y_n)$$

Starting from any autonomous differential equation

$$f(y^{(n)}(t), y^{(n-1)}(t), \ldots, y(t)) = 0$$

via a technique which associates to each respective derivative a state unknown (variable). This

system's form depends on the solvability with respect to the highest derivative $y^{(n)}$.
The first form may be obtained if the above equation **can** be solved in terms of $y^{(n)}$, meaning it can be re-written as f(y$^{(n)}(t), y^{(n-1)}(t), \ldots, y(t)) = 0$
$\Updownarrow$
$y^{(n)} = G(y^{(n-1)}(t), \ldots, y(t))$

We first take each state unknown to be equal to their respective derivatives, so: y$_1 = y$
$y_2 = \dot{y}$
$\vdots$
$y_n = y^{(n-1)}$

Then we take the derivatives with respect to time of each of these equations, so starting with $\dot{y}_1 = \dot{y}$, but $\dot{y} = y_2$, so $\dot{y}_1 = y_2$. Continuing on like that, in a sort of diagonal fashion, we get: $_1 = y_2$
$\dot{y}_2 = y_3$
$\vdots$
$\dot{y}_{n-1} = y_n$
and this is where that beginning "solvability" part comes into play. The very last one will be exactly its solution defined earlier:

$$\dot{y}_n = \frac{d}{dt}y^{(n-1)} = y^{(n)} = G(y^{(n-1)}(t), \ldots, y(t))$$

So, putting this all together we can get the equivalent system of the autonomous differential equation

$$\{\dot{} y_1 = y_2 \dot{y}_2 = y_3 \vdots \dot{y}_{n-1} = y_n \dot{y}_n = G(y^{(n-1)}(t), \ldots, y(t))$$

Now, if instead of an actual function G we only get an implicit form for a solution (**??** isn't satisfied), we could write the original equation in terms of these new state variables:

f(y(t), (t),...,y$^{(n-1)}, y^{(n)}) = 0$
$\Updownarrow$
$f(y_1, \ldots, y_n, \dot{y}_n) = 0$

So the system would be:

$$\{\dot{} y_1 = y_2 \dot{y}_2 = y_3 \vdots \dot{y}_{n-1} = y_n f(y_1, \ldots, y_n, \dot{y}_n) = 0$$

This is a differential-algebraic system of equations (DAE), which is beyond the scope of this thesis but figured I'd cover all cases (this one's usually solved via numerical methods).

Okay, but now since we know how to form systems from any autonomous equation ... how does that make our lives easier in any way?

Let's consider **??**, the equation we worked with previously. Now we can express it differently:

we could first divide everything by a_n so everything looks a bit better.

$a_0 x + a_1 \dot{x} + \cdots + a_n x^{(n)} = 0 \Big| \frac{1}{a_n}$
$\Updownarrow$
$b_0 x + b_1 \dot{x} + \cdots + b_{n-1} x^{(n-1)} + x^{(n)} = 0$

And now apply the aforementioned technique:

x$_1 = x$
$x_2 = \dot{x}$
$\vdots$
$x_n = x^{(n-1)}$

then:

$$\{\dot{} x_1 = x_2\dot{x}_2 = x_3\dot{:}\dot{x}_{n-1} = x_n\dot{x}_n = x^{(n)} = -b_0x - b_1\dot{x} - \cdots - b_{n-1}x^{(n-1)} = -b_0x_1 - b_1x_2 - \cdots - b_nx_{n-1}$$

Waitt, I saw this somewhere in year 1, semester 1 during Linear Algebra (which I definitely didn't have to re-take twice)

That is the same as: $_1$

$\dot{x}_2$

$\dot{x}_3$

$\vdots$

$\dot{x}_{n-1}$

$$\dot{x}_n = \begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & \dots & x_n \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -b_0 & -b_1 & -b_2 & \dots & b_n \end{pmatrix} \end{array} \mathrm{x}_1$$

x_2

x_3

$\vdots$

x_{n-1}

x_n

care poate fi gen rezolvat sa-i gasest eigen valusurile pe care le folosesti cu solutiile fundamentale sa gasesti eigenvectorsii respectivi pt eigenvalurieile alea si poti astfelll gen sa-ti dai seama stabilitatea sistemului =¿ stabilitatea ecuatiei de la care ai plecat din prima gen.

//TODO:1. prima data dai exemple de sisteme liniare si cum fac ecuatiile diferentiale sa se pupe cu algebra liniara si de toate felurile in care poti sa rezolvi algebreic sisteme de ecuatii diferentiale ,eigenvalues eigenvectors din astea

2. SI dupaa zici okay but what if my system is not linear? **Then** how will forming its corresponding system help me? Si dai exemplul cu pendumulul, reintroducand idea de phase space si cum poti sa-l formezi tu acum folosind metoda de mai sus (o sa-i dau un nume mai incolo s-o referentiezi)

Taking, in particular, a more complex autonomous differential equation; the one for the damped pendulum:

$$\ddot{\theta} + \mu\dot{\theta} + \frac{g}{L}\sin(\theta) = 0 \tag{3.13}$$

where:

$\left.\begin{array}{l} \theta \text{ : angular position} \\ \dot{\theta} \text{ : angular velocity} \\ \ddot{\theta} \text{ : angular acceleration} \end{array}\right\}$ all functions of t (time)

$\mu = \frac{c}{mL}$: dampening factor

c : the dampening coefficient due to the resistive forces

m: mass at the end of the pendulum

$g \approx 9.81\frac{m}{s^2}$: earth's gravitational acceleration

L: the pendulum's length

We can express θ and $\dot{\theta}$ as two separate and independent variables in a 2D space of points

$(\theta, \dot{\theta}) = (\theta, \omega)$. And attach to each point the vector formed from the derivatives of each component, which would be (according to **??**):

d$\frac{}{dt\theta\dot{\theta}=\dot{\theta}\ddot{\theta}=\omega-\mu\cdot\omega-\frac{g}{L}\sin(\theta)}$

This would in turn form the phase portrait of the dynamical system, described by a system of first-order autonomous differential equations:

$$\{^{\cdot}\theta = \omega\dot{\omega} = -\mu \cdot \omega - \frac{g}{L}\sin(\theta)$$

TODO: faci tu dupa aici un plot in matlab frumusel si dupa mai gasesti un exemplu mistio in 3d

arat dupa ce faci toate demonstratiile pentru n dimensiuni, niste poze si cazuri particulare in 2,3 dimensiuni, cv ce se paote vedea cu poze

4. arata in state spaceu ala cum iti dai seama stabilitatea sistemului dand graficul oribtelor in vector spaceu ala (sa mai pui si snite poze)

We can observe that the pendulum will not move an inch if it is just left in its "equilibrium" point hanging down ($\omega \equiv \theta \equiv 0$), but that like a coin left on its side, it will also stay still if started at the **polar** opposite angle in relation to the stability angle ($\omega = 0, \theta = 180\circ$), standing up. Althought, like a coin left standing on its side, even the slightest deviation will get it tumbling down. This can be observed in the system's phase portrait. The study of how much such changes in initial conditions influence later behaviour is what we call the **stability** of certain points inside the system.

5. arata poza de clasificarea sistemelor, dat

3.4 What is a Phosphorylation–Dephosphorylation CRN?

3.5 What is a simple Hopf Bifurcation?

A *simple Hopf bifurcation* is a bifurcation in which a single complex-conjugate pair of eigenvalues of the Jacobian matrix crosses the imaginary axis, while all other eigenvalues remain with negative real parts. Such a bifurcation generates nearby oscillations or periodic orbits.

Chapter 4

Existence and Absence of Hopf Bifurcation in Phosphorylation–Dephosphorylation CRN

4.0.1 Proving the existence of simple Hopf bifurcations

4.0.2 Ruling out simple Hopf bifurcations

4.0.3 Convex parameters

4.1 Cyclic and mixed distributive and processive Phosphorylation–Dephosphorylation CRN

Chapter 5

CoNtRoL Simulations Web Application

Here we will present the open-source Web application developed for easing work involving chemical reaction networks, hopefully for students and researchers one day. It can be used for obtaining numerical analysis of chemical reaction networks, as well as plotting species against time, one another and so on. We'll present the technologies used throughout the project, how to run it locally and a couple of usage examples involving what we've presented in the previous chapters.

5.0.1 Overview of the technologies

The website is a back-end application built in **Python**, a programming language known for its use in basically every single science, including natural sciences so it's a no-brainer when in comes to plotting. The web server is built using **Flask**, a lightweight web app framework and the webpages served are server-side rendered by Flask's template engine depedency - **Jinja**.

The crux of the functionality is aided by the Python library **Tellurium**; which is, as their docs say; "A Python Environment for Reproducible Dynamical Modeling of Biological Networks". It uses a subset of the Systems Biology Markup Language (**SBML**) called **Antimony** which can be used in this app to create a Chemical Reaction Network, as well as the friendlier selects form. So the bits doing the magic are the calls to `road_runner.loada()` which are used to ***load*** **a**ntimony code into the model. the `road_runner..simulate()` function is then used for running and obtaining sumulation data, followed by `road_runner.plot()`, which in turn calls a `matplotlib` headless backend for writing the plotted results to a file.

5.0.2 Running it

As explained in the `README` of the project, running locally is done automatically by the `run_script.sh`, which figures out your machine's local IP, sets the required environment variables and runs `flask --debug run --host="$ip"`. Output regarding traffic to the server as well as internal workings of the app will now be redirected to `stdout` of the terminal.

All the user has to do beforehand is:

- clone the project

```
git clone https://github.com/viktorashi/Open-CoNtRol.git
```

- change directory into it.

```
cd Open-CoNtRol
```

- install the requirements

```
pip install -r requirements.txt
```

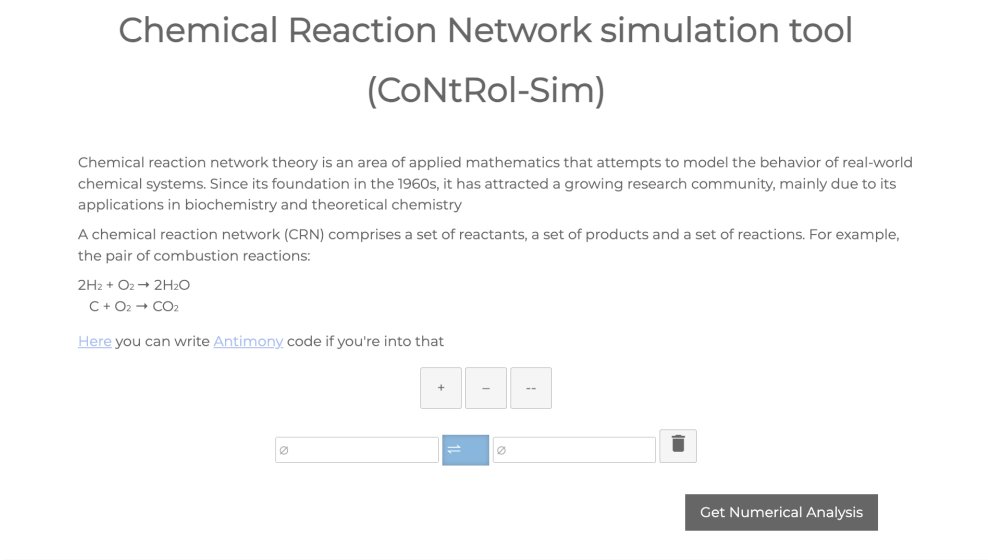
- give the `run_script.sh` execute permissions

```
chmod +x ./run_script.sh
```

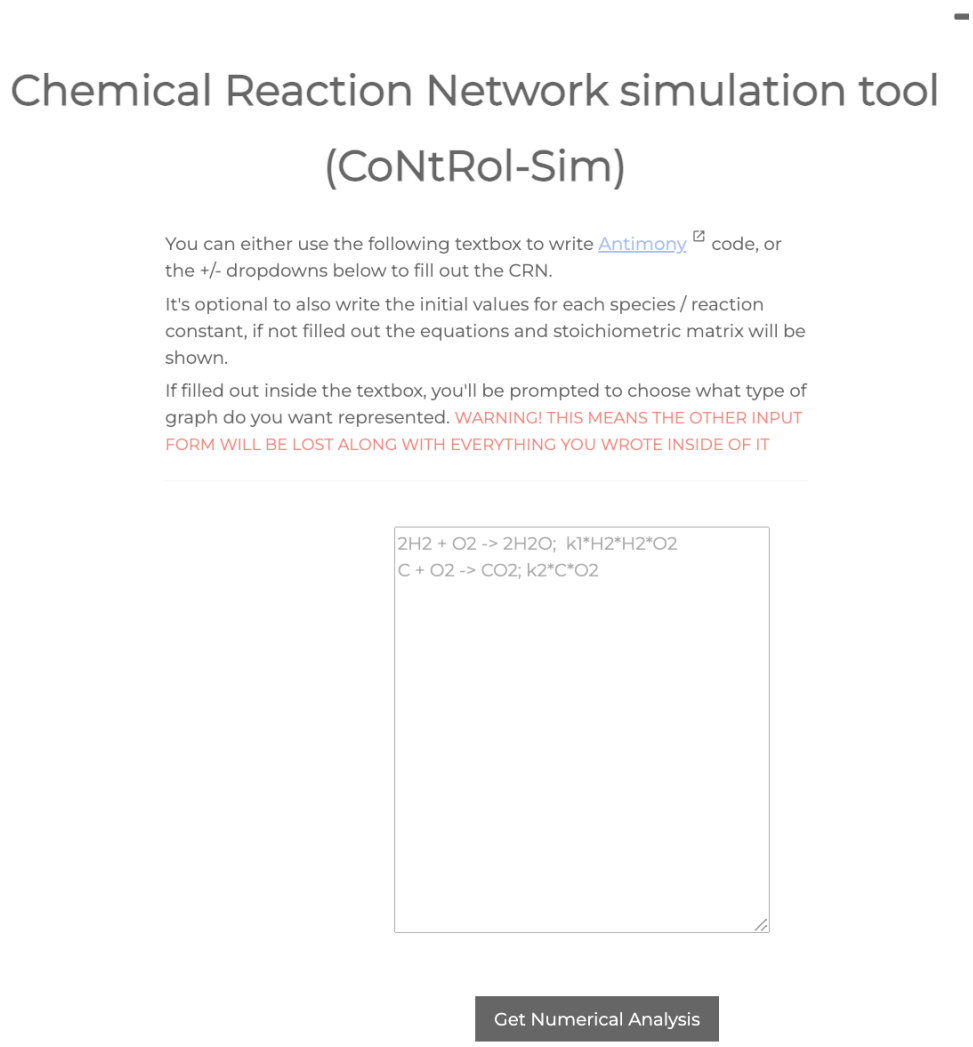
- and finally run it

```
./run_script.sh
```

Among the wall of output will also be the line showing the address your server is located at, for example: `* Running on http://192.168.0.94:5000`, address at which you'll be greeted with this screen



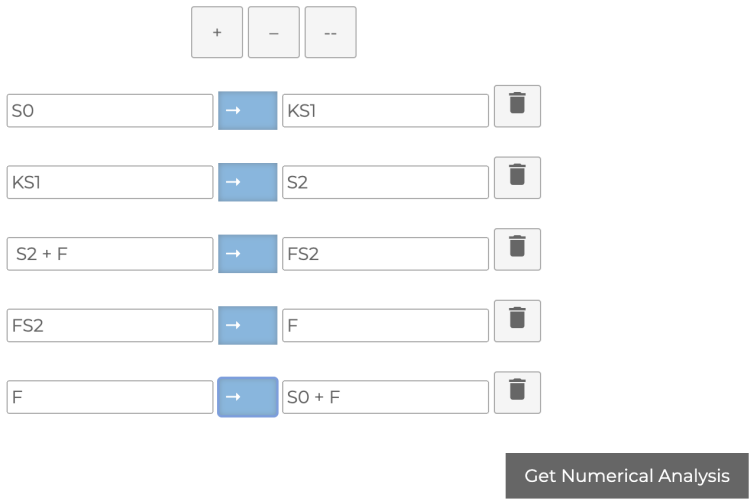
You can either use this as a starting point or the page located at the `/antimony` path:



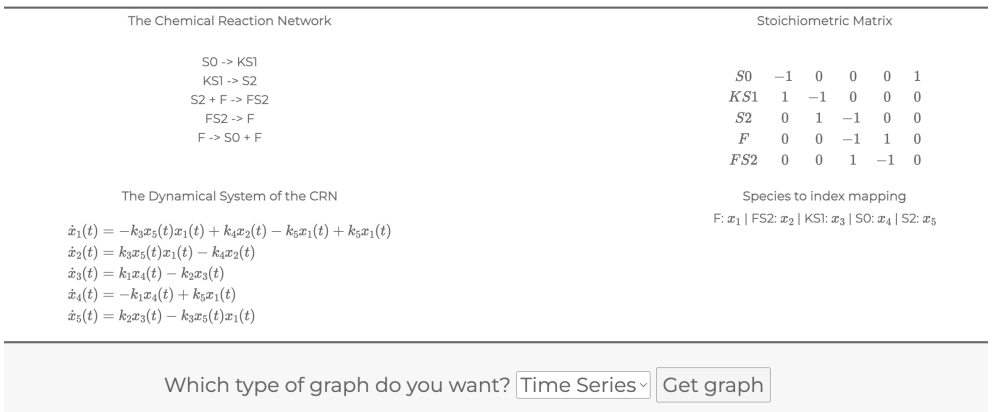
Both of these redirect to `/numerical_analysis` analysing the system ... well, numerically. As an example, this Antimony code:

```
S0 -> KS1; k1*S0
KS1 -> S2; k2*KS1
S2 + F -> FS2; k3*S2*F
FS2 -> F; k4*FS2
F -> S0 + F; k5*F
```

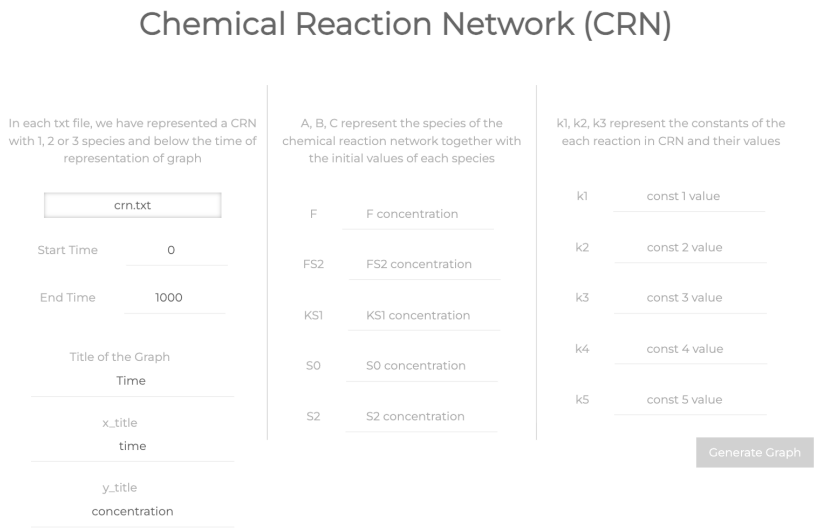
and its longer to write alternative:



both yield the same numerical analysis results:



from here, one can choose from a selection of graphs they can represent given this system, the default one being the time series representation:

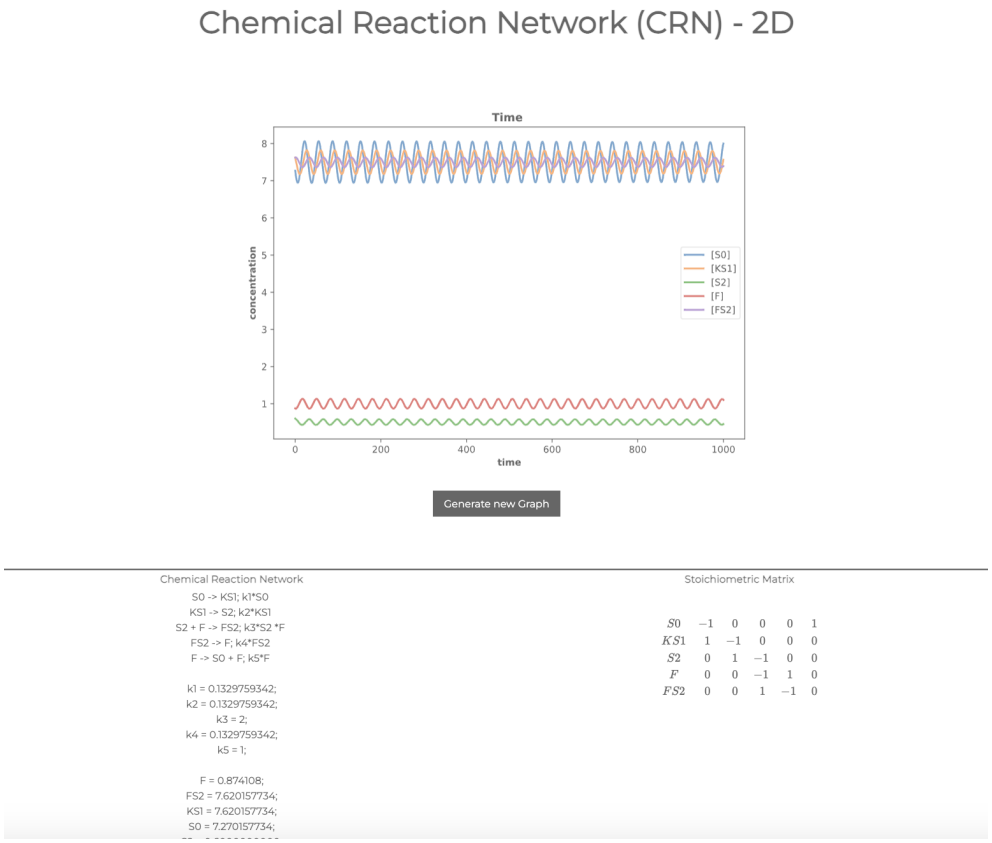


from which we fill out the initial values of the concentrations for each species as, well as the reaction rates. So given, for example the values:

```
F = 0.874108
FS2 = 7.620157734
KS1 = 7.620157734
S0 = 7.270157734
S2 = 0.6000000000

k1 = 0.1329759342
k2 = 0.1329759342
k3 = 2
k4 = 0.1329759342
k5 = 1
```

outcomes the graph:



.

5.0.3 The bottom line

So this is how the workflow of the app typically goes.

write out your system → get numerical analysis → choose your desired graph

Voilà