



Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**



Grado en Matemáticas e Informática

Trabajo Fin de Grado

**Desarrollo de una Aplicación Didáctica
en Ecuaciones Diferenciales: Sistemas
Lineales, Puntos de Equilibrio y
Estabilidad**

Autora: Lucía Jiang
Tutor: Javier López de la Cruz
Cotutor: Héctor Barge Yañez

Madrid, Mayo - 2023

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Grado
Grado en Matemáticas e Informática*

Título: Desarrollo de una Aplicación Didáctica en Ecuaciones Diferenciales: Sistemas Lineales, Puntos de Equilibrio y Estabilidad

Mayo - 2023

Autora: Lucía Jiang

Tutor: Javier López de la Cruz

Departamento de Matemática Aplicada a las TIC

Escuela Técnica Superior de Ingenieros Informáticos

Universidad Politécnica de Madrid

Cotutor: Héctor Barge Yañez

Departamento de Matemática Aplicada a las TIC

Escuela Técnica Superior de Ingenieros Informáticos

Universidad Politécnica de Madrid

Agradecimientos

Me gustaría expresar mi más sincero agradecimiento a todas las personas e instituciones que han contribuido al éxito de mi Trabajo Fin de Grado:

En primer lugar, me gustaría agradecer a mis supervisores, Javier y Héctor, por su orientación, sabiduría y retroalimentación a lo largo de todo el proceso. También quiero agradecer a todos los profesores del Grupo de Innovación Educativa del departamento de DMATIC por liderar este proyecto, y a los alumnos de Practicum involucrados en el mismo. Su colaboración ha sido fundamental para el desarrollo de esta aplicación de software matemático y la realización de este Trabajo Fin de Grado.

Además, quiero agradecer a todos mis amigos, en especial a Laura y a Paula, quienes han estado a mi lado durante todo este proceso del Trabajo Fin de Grado. Vuestro compañerismo y apoyo constante han sido un gran impulso para superar los desafíos y mantenerme motivada. Gracias por escucharme cuando necesitaba desahogarme y por celebrar mis logros junto a mí.

Por último, quiero expresar mi profundo agradecimiento a mis familiares por vuestro constante apoyo y comprensión a lo largo de mi Trabajo Fin de Grado y toda la carrera. Vuestro aliento y amor incondicional han sido un pilar fundamental en este camino académico. Gracias por creer en mí y por ser una fuente inagotable de motivación y apoyo en cada etapa.

Resumen

El presente Trabajo Fin de Grado se centra en el desarrollo de un software matemático enfocado en los sistemas de ecuaciones diferenciales de 2 ecuaciones de primer orden linealmente independientes.

Este trabajo forma parte de un proyecto liderado por el Grupo de Innovación Educativa del departamento DMATIC de la Escuela Técnica Superior de Ingenieros Informáticos, con el objetivo de mejorar la comprensión y el aprendizaje de los estudiantes universitarios. La aplicación desarrollada permite a los alumnos acceder a una calculadora que resuelve ejercicios de diferentes materias relacionadas con las carreras de nuestra facultad.

El desarrollo de la herramienta se divide en dos partes: el *frontend* y el *backend*. El *frontend* de la aplicación está a cargo de los alumnos en prácticas que se encargan de diseñar la interfaz del usuario. Por otro lado, el *backend* y la conexión de la aplicación son desarrollados por los estudiantes de Trabajo Fin de Grado. Estos alumnos nos encargamos de implementar los algoritmos de resolución de las asignaturas, en mi caso, Ecuaciones Diferenciales, utilizando como lenguaje de programación Python. Nuestro principal objetivo es proporcionar unos paquetes de resolución que permitan resolver mediante pasos detallados los diferentes ejercicios planteados por los usuarios.

A través de la implementación de estos algoritmos y el uso de tecnologías modernas, como Angular o Python, hemos logrado desarrollar esta aplicación, que fomenta el estudio y la comprensión de las disciplinas matemáticas. Asimismo, la colaboración entre los estudiantes, profesores y tutores ha permitido la creación de esta herramienta alineada con los Objetivos de Desarrollo Sostenible.

Abstract

The present Bachelor's Thesis focuses on the development of mathematical software focused on systems of differential equations with two linearly independent first-order equations.

This work is part of a project led by the Educational Innovation Group of the DMATIC department at the Escuela Técnica Superior de Ingenieros Informáticos, with the aim of improving the understanding and learning of university students. The developed application allows students to access a calculator that solves exercises from different subjects related to our faculty's programs.

The development of the tool is divided into two parts: the frontend and the backend. The frontend of the application is handled by Practicum students responsible for designing the user interface. On the other hand, the backend and the application's connection are developed by the students working on their Bachelor's Thesis. In my case, I have implemented resolution algorithms specific to the Differential Equations subject, using the Python programming language. Our main objective is to provide resolution packages that allow users to solve different exercises step by step with detailed explanations.

Through the implementation of these algorithms and the use of modern technologies such as Angular and Python, we have successfully developed this application, which promotes the study and understanding of mathematical disciplines. Furthermore, the collaboration between students, professors, and tutors has facilitated the creation of this tool aligned with the Sustainable Development Goals.

Tabla de contenidos

1. Introducción	1
2. Marco teórico	3
2.1. Solución explícita	4
2.1.1. Diagonalizable. Autovalores reales y distintos	6
2.1.2. Diagonalizable. Autovalores reales e iguales	8
2.1.3. Diagonalizable. Autovalores complejos	9
2.1.4. No diagonalizable. Autovalor real e igual	12
2.2. Clasificación de órbitas y puntos por autovalores	13
2.2.1. Autovalores reales y distintos	13
2.2.1.1. Nodo estable	15
2.2.1.2. Nodo inestable	16
2.2.1.3. Punto de silla	17
2.2.2. Autovalor doble	18
2.2.2.1. Punto estelar	19
2.2.2.2. Nodo impropio	20
2.2.3. Autovalores complejos	22
2.2.3.1. Centro estable	24
2.2.3.2. Foco estable	25
2.2.3.3. Foco inestable	25
2.3. Clasificación de puntos por traza y determinante	27
2.3.1. Autovalores reales y distintos	28
2.3.2. Autovalores reales e iguales	29
2.3.3. Autovalores complejos	29
2.4. Clasificación dinámica	31
2.5. Transformación de ecuación de segundo orden en sistema de primer orden	32
3. Código	33
3.1. Estructura código	33
3.2. Conexión Front-End y Back-End	35
3.3. Pseudocódigo y algoritmos	37
3.3.1. Sistema Fundamental de Soluciones	37
3.3.2. Soluciones Explícitas	39
3.3.3. Ecuación de segundo orden a sistema	39
3.3.4. Clasificación puntos de equilibrio mediante los autovalores .	39

TABLA DE CONTENIDOS

3.3.5. Clasificación puntos de equilibrio mediante la traza y el de- terminante	41
3.3.6. Representar diagramas de fases	42
3.3.7. Conjugaciones topológicas	43
3.4. Librerías externas	43
4. Resultados y conclusiones	45
4.1. Resultados	45
4.1.1. Listado de ejercicios	45
4.1.2. Sistemas de ecuaciones diferenciales	46
4.1.3. Estabilidad	48
4.1.4. Conjugación topológica	50
4.1.5. Ecuaciones de segundo orden	52
4.2. Propuestas de uso y extensión	53
4.3. Conclusión	55
5. Análisis de impacto	57
Bibliografía	60
A. Demostraciones	61
B. Documentación código	67
B.1. Resolución ecuaciones y sistemas	67
B.2. Estabilidad	68
B.3. Conjugación topológica	68
C. Códigos desarrollados	69

Capítulo 1

Introducción

El presente Trabajo Fin de Grado forma parte del proyecto “Desarrollo de Tecnologías en la Enseñanza de las Matemáticas”, liderado por el Grupo de Innovación Educativa (GIE) del Departamento de Matemática Aplicada a las TIC (DMATIC) de la Escuela Técnica Superior de Ingenieros Informáticos. El objetivo principal de este proyecto consiste en crear un software que facilite el aprendizaje de las distintas disciplinas matemáticas a los estudiantes de nuestra universidad. Para ello, se ha previsto el desarrollo de diferentes paquetes de resolución de problemas correspondientes a cada una de las asignaturas de las titulaciones. De esta manera, se desarrollará una página web que servirá como herramienta de estudio, que permitirá a los estudiantes profundizar en los contenidos y mejorar su comprensión.

El proyecto se divide en dos partes: el *frontend* y el *backend* de la aplicación web. El *frontend* es desarrollado por alumnos en prácticas supervisados por miembros del DMATIC, mientras que del *backend* se encargan los estudiantes del TFG. En concreto, los alumnos de TFG eligen una asignatura del plan de estudios y programan en Python la resolución de los ejercicios correspondientes a esa asignatura, además de coordinarse con los estudiantes en prácticas y conectar las dos capas de la aplicación.

La asignatura escogida para implementar en este Trabajo Fin de Grado es Ecuaciones Diferenciales, una asignatura de 2º del Grado de Matemáticas e Informática. En particular, se desarrollarán los códigos para resolver los sistemas de ecuaciones autónomos lineales de dos ecuaciones con matriz de coeficientes con determinante no nulo.

Estudiar sistemas autónomos lineales de dos ecuaciones es importante por varias razones. En primer lugar, estos sistemas son comunes en la física, la economía y otras áreas de la ciencia, por lo que entenderlos es fundamental para comprender los modelos matemáticos que se utilizan en estas disciplinas. En segundo lugar, el análisis de sistemas autónomos lineales de dos ecuaciones también proporciona una base sólida para comprender sistemas más complejos y no lineales. Muchos sistemas no lineales se pueden aproximar por sistemas lineales en una vecindad de un punto de equilibrio, por lo que el estudio de sis-

temas lineales de dos ecuaciones es importante para entender la dinámica de sistemas más complejos.

Este trabajo se encuentra estructurado en distintas secciones que abordan diferentes aspectos relacionados con los sistemas de ecuaciones diferenciales autónomos lineales y su implementación en una aplicación informática. En el Capítulo 1, se introduce el proyecto de Desarrollo de Tecnologías en la Enseñanza de las Matemáticas y se describe el objetivo principal del proyecto: la creación de un software que facilite el aprendizaje de las distintas disciplinas matemáticas a los estudiantes de la universidad. Además, se explica la estructura del trabajo y se justifica la elección de la asignatura de Ecuaciones Diferenciales como objeto de estudio en este Trabajo Fin de Grado.

En el Capítulo 2 se profundiza en el marco teórico en el que se basa el trabajo, examinando de manera detallada el proceso para obtener soluciones explícitas de un sistema. Además, se aborda el cálculo de los puntos de equilibrio del problema y su clasificación. Se explora la representación gráfica de las soluciones mediante la visualización del diagramas de fases, que permite observar gráficamente el comportamiento de un sistema en el espacio de fases, y finalmente se estudian las conjugaciones topológicas.

Posteriormente, en el Capítulo 3, se describe la estructuración del código desarrollado, incluyendo cómo se ha logrado la conexión entre el *frontend* y el *backend*. Se muestran los pseudocódigos de los algoritmos de resolución implementados, se presentan las librerías externas utilizadas y se justifica su uso.

En el Capítulo 4 se presentan los resultados obtenidos a través la aplicación desarrollada, así como su funcionamiento y propuestas de uso en el aula. Además, se sugieren posibles mejoras y extensiones en la dirección del trabajo realizado para futuros proyectos.

Por último, en el Capítulo 5, se realiza un análisis crítico del impacto del trabajo en relación con los Objetivos de Desarrollo Sostenible establecidos en la Agenda de 2030. Se identifican los aspectos en los que el proyecto contribuye a la consecución de dichos objetivos.

Capítulo 2

Marco teórico

Como hemos mencionado en la Introducción, en este Trabajo Fin de Grado nos centraremos en estudiar los sistemas 2×2 diferenciales homogéneos lineales de la forma:

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

donde a, b, c y d son constantes reales.

Denotaremos la matriz de coeficientes como

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

y la matriz de incógnitas como $X = (x, y)^t$. Así, podemos expresar el sistema lineal como

$$X' = AX.$$

En un sistema de ecuaciones diferenciales, un **punto de equilibrio** es un punto en el espacio de fases del sistema donde ninguna variable cambia con el tiempo. En otras palabras, es un punto donde las derivadas de todas las variables son iguales a cero (véase [1]).

El punto de origen $(0, 0)$ es siempre un punto de equilibrio para del sistema $X' = AX$ y para encontrar otros puntos de equilibrio resolvemos el sistema homogéneo asociado:

$$\begin{cases} ax + by = 0, \\ cx + dy = 0. \end{cases}$$

Según el Teorema de Rouché-Frobenius (véase [2]), el sistema tiene una solución distinta del vector cero si y solo si $\det(A) = 0$. Por tanto, se pueden dar los siguientes dos casos:

- Si $\det(A) = 0$, existe una recta que pasa por el origen y todos sus puntos son soluciones del sistema.
- Si $\det(A) \neq 0$, el único punto de equilibrio es el punto de origen.

Para resolver el caso $\det(A) = 0$, sería necesario la intervención humana (véase [1]) y en este trabajo, no contemplaremos resolver sistemas con múltiples soluciones. Entonces, a los sistemas que vamos a estudiar añadimos la restricción de que $\det(A) \neq 0$.

En conclusión, en este trabajo se estudiarán los sistemas de ecuaciones diferenciales homogéneos de dos variables y dos ecuaciones, con coeficientes reales, cuyo determinante asociado a la matriz de coeficientes es distinto de cero.

En las siguientes secciones se abordará cómo obtener la solución explícita de estos sistemas en función de sus autovalores, se explorará la transformación de ecuaciones diferenciales de segundo orden a sistemas, se analizarán las clasificaciones de los puntos de equilibrio y se estudiará el concepto de conjugación topológica. Para más información sobre la base teórica del presente trabajo, referimos al lector a [1, 3, 4, 5, 6, 7, 2], en las que se basa el contenido de este capítulo

2.1. Solución explícita

En esta sección se explicará cómo obtener explícitamente las soluciones generales de un sistema. Para ello, se deberán considerar varios casos dependiendo de si la matriz de coeficientes $A_{2 \times 2}$ asociada al sistema es diagonalizable o no, y de cómo sean los autovalores. Los casos a estudiar son los siguientes:

1. Diagonalizable.
 - a) Autovalores reales y distintos.
 - b) Autovalores reales e iguales.
 - c) Autovalores complejos.
2. No diagonalizable: autovalores reales e iguales.

Pero antes de adentrarnos en estos casos, daremos un par de definiciones.

Definición 1 (Matriz fundamental). *Una matriz fundamental asociada al sistema $X' = AX$ es cualquier matriz cuadrada $\phi(t)$ cuyas columnas son soluciones de $X' = AX$ linealmente independientes en \mathbb{R} (véase [7]).*

Observación 1. *Si $\phi(t)$ una matriz fundamental del sistema $X' = AX$, se cumple que:*

1. $\phi'(t) = A\phi(t)$ para todo $t \in \mathbb{R}$, ya que $\phi(t)$ está formado por soluciones del sistema $X' = AX$.
2. $\det(\phi(t)) \neq 0$ para todo $t \in \mathbb{R}$. Como las soluciones fundamentales que conforman la matriz fundamental son linealmente independientes, se garantiza que la matriz fundamental es una base del espacio de soluciones del sistema de ecuaciones.

Definición 2 (Wronskiano). *El Wronskiano de un conjunto de funciones diferenciables x_1, \dots, x_n se define como sigue:*

Marco teórico

$$W[x_1, \dots, x_n](t) = \begin{vmatrix} x_1(t) & x_2(t) & \cdots & x_n(t) \\ x'_1(t) & x'_2(t) & \cdots & x'_n(t) \\ \vdots & \vdots & & \vdots \\ x_1^{(n-1)}(t) & x_2^{(n-1)}(t) & \cdots & x_n^{(n-1)}(t) \end{vmatrix},$$

donde $x_i^{(k)}(t)$ denota la k -ésima derivada de la función $x_i(t)$ evaluada en t (véase [7]).

Definición 3. Sea $\mathcal{M}_n(\mathbb{K})$ el conjunto de matrices de dimensión n en el cuerpo \mathbb{K} de los números reales o de los números complejos, y sea $A \in \mathcal{M}_n(\mathbb{K})$. Se define la aplicación matricial exponencial $e^{At} : \mathbb{R} \rightarrow \mathcal{M}_n(\mathbb{K})$ que para todo $t \in \mathbb{R}$ le hace corresponder:

$$e^{At} = \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}.$$

Teorema 1 (Fórmula de Jacobi). *Sea I un intervalo abierto. Si $x_1(t), \dots, x_n(t)$ son soluciones del sistema $X' = AX$. Entonces, se verifica*

$$W(t) = W(t_0) e^{\int_{t_0}^t \text{Tr}(A(s)) ds}$$

para cualquier $t_0 \in I$ y siendo $W[x_1, \dots, x_n](t)$ el wronskiano definido en la Definición 2 (véase [7]).

La demostración de este teorema se deja en el Anexo A.

Una vez enunciados los teoremas, definiciones y observaciones necesarias podemos proseguir buscando las soluciones explícitas de los sistemas diferenciales lineales $X' = AX$ de dos ecuaciones. Se cumple la siguiente proposición.

Proposición 1. *Sea el sistema de ecuaciones diferenciales de primer orden $X' = AX$, se verifica que:*

1. e^{At} es una matriz fundamental.
2. La solución general es de la forma:

$$X(t) = e^{At} c, \quad c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}, \quad c_i \in \mathbb{R}, i = 1, \dots, n. \quad (2.1)$$

Demostración. Comencemos demostrando que e^{At} es una matriz fundamental. Recordemos la definición matriz fundamental, indicado en la Observación 1.

Por una parte, se cumple que $\frac{d}{dt} e^{At} = Ae^{At}$. Por otra, veamos que $\det(e^{At}) \neq 0$.

Utilizando la fórmula de Jacobi, enunciada en el Teorema 1, tenemos que

$$W(t) = W(t_0) e^{\int_{t_0}^t \text{Tr}(A(s)) ds},$$

con $W(t) = W[Y_1, \dots, Y_n](t)$ el Wronskiano. Entonces,

$$\det(e^{At}) = W(t) = \det(e^{A \cdot 0}) e^{\int_0^t \text{Tr}(A(s)) ds} = e^{\text{Tr}(A)t},$$

de donde $\det(e^{At}) = e^{\text{Tr}(A)t} \neq 0$.

Ahora, veamos que la solución general tiene la forma $X(t) = e^{At}c$, como indican las ecuaciones (2.1).

Puesto que e^{At} es una matriz fundamental, sus columnas son un sistema fundamental de soluciones. La combinación lineal de soluciones es otra solución del sistema. Para demostrar esto, consideremos la función $Z(t) = \alpha X_1(t) + \beta X_2(t)$ donde $X_1(t), X_2(t)$ son soluciones de $X' = AX$. Veamos que $Z(t)$ también es solución:

$$Z'(t) = \alpha X'_1(t) + \beta X'_2(t),$$

y por ser $X_1(t)$ y $X_2(t)$ soluciones del sistema,

$$Z'(t) = \alpha AX_1(t) + \beta AX_2(t).$$

Además, por la linealidad de las matrices podemos concluir con que

$$Z'(t) = A(\alpha X_1(t) + \beta X_2(t)).$$

Hemos demostrado que $Z'(t) = AZ(t)$ y por tanto, la solución general se puede expresar como una combinación lineal de las columnas de e^{At} , esto es, $X(t) = e^{At}c$ donde c es un vector de constantes. \square

Dependiendo de cómo sea la matriz A , diagonalizable o no diagonalizable, con autovalores reales y distintos, iguales o complejos, la expresión e^{At} de la solución general variará. En las próximas secciones estudiaremos de qué forma es e^{At} .

2.1.1. Diagonalizable. Autovalores reales y distintos

Sea A una matriz diagonalizable con dos autovalores $\lambda_1, \lambda_2 \in \mathbb{R}$ tal que λ_1 es diferente de λ_2 y con autovectores v_1, v_2 (que existen porque los autovalores son simples) asociados a λ_1 y λ_2 , respectivamente. Entonces, tenemos que $A = PDP^{-1}$, con

$$P = \begin{pmatrix} | & | \\ v_1 & v_2 \\ | & | \end{pmatrix}, \quad D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

En consecuencia,

$$e^{At} = e^{PDP^{-1}t}. \tag{2.2}$$

Teorema 2. *En las condiciones previamente descritas se tiene que*

$$e^{At} = Pe^{Dt}P^{-1}.$$

Marco teórico

Demostración. Comencemos estudiando cómo son las potencias de PAP^{-1} .

$$(PAP^{-1})^2 = PAP^{-1}PAP^{-1} = PA(P^{-1}P)AP^{-1} = PA \cdot I \cdot AP^{-1} = PA^2P^{-1}.$$

Elevándolo a la k -potencia tenemos que

$$(PAP^{-1}) \cdots (PAP^{-1}) = PA^kP^{-1}.$$

Utilizando la Definición 3 de elevar al número e la matriz A ,

$$\begin{aligned} e^{(PDP^{-1})t} &= \sum_{k=0}^{\infty} \frac{(PDP^{-1})^k t^k}{k!} \\ &= I + \frac{PDP^{-1}t}{1!} + \frac{(PDP^{-1})^2 t^2}{2!} + \cdots + \frac{(PDP^{-1})^k t^k}{k!} + \cdots \\ &= I + \frac{PAP^{-1}t}{1!} + \frac{PA^2P^{-1}t^2}{2!} + \cdots + \frac{PA^kP^{-1}t^k}{k!} + \cdots \\ &= I + P \left(\frac{At}{1!} + \frac{A^2t^2}{2!} + \cdots + \frac{A^k t^k}{k!} + \cdots \right) P^{-1} \\ &= PIP^{-1} + P \left(\frac{At}{1!} + \frac{A^2t^2}{2!} + \cdots + \frac{A^k t^k}{k!} + \cdots \right) P^{-1} \\ &= P \left(I + \frac{At}{1!} + \frac{A^2t^2}{2!} + \cdots + \frac{A^k t^k}{k!} + \cdots \right) P^{-1} \\ &= Pe^{At}P^{-1}. \end{aligned}$$

Queda así demostrado que $e^{PDP^{-1}t} = Pe^{Dt}P^{-1}$. □

Teorema 3. Si

$$D = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

es una matriz diagonal, entonces

$$e^{Dt} = \begin{pmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_n t} \end{pmatrix}.$$

Demostración. Es claro que si

$$D = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} \quad \text{se tiene que} \quad D^k = \begin{pmatrix} \lambda_1^k & & \\ & \ddots & \\ & & \lambda_n^k \end{pmatrix}, \quad \text{con } k \in \mathbb{Z}^+,$$

y en consecuencia,

$$e^{Dt} = \begin{pmatrix} \sum_{k=0}^{\infty} \frac{\lambda_1^k t^k}{k!} & & \\ & \ddots & \\ & & \sum_{k=0}^{\infty} \frac{\lambda_n^k t^k}{k!} \end{pmatrix} = \begin{pmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_n t} \end{pmatrix}.$$

□

Entonces, siguiendo con la ecuación (2.2),

$$e^{At} = e^{PDP^{-1}t} = Pe^{Dt}P^{-1} = P \begin{pmatrix} e^{\lambda_1 t} & 0 \\ 0 & e^{\lambda_2 t} \end{pmatrix} P^{-1}.$$

Ahora, teniendo en cuenta que $\phi(t) = e^{At}P^{-1}$ es otra matriz fundamental, la solución general es de la forma:

$$X(t) = e^{At}c = Pe^{Dt} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$

es decir,

$$X(t) = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2, \quad \text{siendo } c_1, c_2 \in \mathbb{R}.$$

2.1.2. Diagonalizable. Autovalores reales e iguales

Supongamos que ahora A tiene un único autovalor real λ , con multiplicidad (algebraica) 2 y que $\dim(\ker(A - \lambda I)) = 2$. Es decir, que la multiplicidad algebraica coincide con la geométrica para que la matriz sea diagonalizable. La única manera que esto ocurra es que A ya sea una matriz diagonal, ya que de otra forma no podríamos encontrar ninguna matriz P tal que $A = PDP^{-1}$ con D una matriz diagonal.

Entonces, A tiene que ser de la forma

$$A = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$$

y por tanto,

$$e^{At} = \begin{pmatrix} e^{\lambda t} & 0 \\ 0 & e^{\lambda t} \end{pmatrix}.$$

Entonces, la solución general es similar al caso anterior:

$$X(t) = e^{At} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = e^{\lambda t} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad \text{donde } c_1, c_2 \in \mathbb{R}.$$

Otra manera de resolver el problema, es desarrollando el sistema, obteniendo que

$$\begin{cases} x' = \lambda x, \\ y' = \lambda y. \end{cases}$$

Teorema 4. *Todas las soluciones de la ecuación $x'(t) = ax(t)$ con $a \in \mathbb{R}$ son de la forma*

$$x(t) = ke^{at}, \quad \text{con } k \in \mathbb{R}.$$

Marco teórico

Demostración. Vemos primero que es solución. Esto es directo ya que

$$x'(t) = ake^{at} = ax(t).$$

Además, veamos que es la única solución. Para ello, veamos que si $u(t)$ es una solución cualquiera de la ecuación $x'(t) = ax(t)$, entonces existe una constante $c \in \mathbb{R}$ de manera que:

$$u(t) = ce^{at} \implies u(t)e^{-at} = c.$$

Derivando en función de t , obtenemos

$$\frac{d}{dt}(u(t)e^{-at}) = u'(t)e^{at} + u(t)(-ae^{-at}).$$

Como $u(t)$ es solución de la ecuación, entonces $u'(t) = au(t)$ y por tanto

$$\frac{d}{dt}(u(t)e^{-at}) = au(t)e^{-at} - au(t)e^{-at} = 0.$$

Luego como $\frac{d}{dt}u(t)e^{-at} = 0$, se tiene que $c = u(t)e^{-at}$, por lo que es una constante. Así, queda demostrado que todas las soluciones de la ecuación $x'(t) = ax(t)$ son de la forma $u(t) = ke^{at}$ con $k \in \mathbb{R}$ alguna constante. \square

Entonces, resolviendo cada ecuación por separado, obtenemos directamente la solución:

$$\boxed{\begin{cases} x(t) = c_1 e^{\lambda t}, & \text{con } c_1 \in \mathbb{R}, \\ y(t) = c_2 e^{\bar{\lambda} t}, & \text{con } c_2 \in \mathbb{R}. \end{cases}}$$

2.1.3. Diagonalizable. Autovalores complejos

Ahora supongamos que la matriz de coeficientes A del sistema lineal de ecuaciones diferenciales $X' = AX$ tiene dos autovalores complejos, que denotaremos como $\lambda = p + iq$ y $\bar{\lambda} = p - iq$. Además, denotemos $v = u + iw$ y $\bar{v} = u - iw$, con $u, w \in \mathbb{R}^2$, los autovectores asociados a λ y $\bar{\lambda}$, respectivamente.

Proposición 2. En las condiciones del párrafo anterior, se tiene que A puede expresarse como $A = PBP^{-1}$, siendo

$$P = \begin{pmatrix} | & | \\ u & w \\ | & | \end{pmatrix} \quad y \quad B = \begin{pmatrix} p & q \\ -q & p \end{pmatrix}.$$

Demostración. Como $v = u + iw$,

$$Av = A(u + iw) = Au + iAw.$$

Teniendo en cuenta la definición de autovalor:

$$Av = \lambda v = (p + iq)(u + iw) = pu + iqu + ipw - qw = pu - qw + i(qu + pw).$$

2.1. Solución explícita

Igualando las partes reales por un lado y las imaginarias por otro:

$$\begin{cases} Au = pu - qw, \\ Aw = qu + pw. \end{cases}$$

Entonces, podemos denotar como B a la matriz asociada a A en la base u, w , como

$$B = \begin{pmatrix} p & q \\ -q & p \end{pmatrix},$$

y se cumple que $A = PBP^{-1}$. □

Hemos demostrado que $A = PBP^{-1}$, entonces con esta expresión y utilizando lo demostrado en el Teorema 2

$$e^{At} = e^{PBP^{-1}t} = Pe^{Bt}P^{-1}.$$

Veamos cómo expresar e^{Bt} . Podemos escribir B como $B = D + Q$, con

$$D = \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix} \quad \text{y} \quad Q = \begin{pmatrix} 0 & q \\ -q & 0 \end{pmatrix}. \quad (2.3)$$

Proposición 3. Si Q está definida como en (2.3), se tiene que

$$e^{Qt} = \begin{pmatrix} \cos(qt) & \sin(qt) \\ -\sin(qt) & \cos(qt) \end{pmatrix}.$$

Demostración. Las potencias de la matriz Q para $n \in \mathbb{Z}^+$ son de la forma:

$$Q^{2n} = \begin{pmatrix} (-1)^n q^{2n} & 0 \\ 0 & (-1)^n q^{2n} \end{pmatrix} \quad \text{y} \quad Q^{2n+1} = \begin{pmatrix} 0 & (-1)^n q^{2n+1} \\ (-1)^{n+1} q^{2n+1} & 0 \end{pmatrix}.$$

Entonces, por definición de exponencial de una matriz,

$$e^{Qt} = \sum_{k=0}^{\infty} \frac{Q^k t^k}{k!}.$$

Separando el sumatorio y utilizando los valores de las potencias de Q^n , tenemos

$$\begin{aligned} e^{Qt} &= \sum_{k=0}^{\infty} \frac{Q^{2k} t^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{Q^{2k+1} t^{2k+1}}{(2k+1)!} \\ &= \sum_{k=0}^{\infty} \frac{t^{2k}}{(2k)!} \begin{pmatrix} (-1)^k q^{2k} & 0 \\ 0 & (-1)^k q^{2k} \end{pmatrix} + \sum_{k=0}^{\infty} \frac{t^{2k+1}}{(2k+1)!} \begin{pmatrix} 0 & (-1)^k q^{2k+1} \\ (-1)^{k+1} q^{2k+1} & 0 \end{pmatrix} \\ &= \begin{pmatrix} \sum_{k=0}^{\infty} \frac{(-1)^k (qt)^{2k}}{(2k)!} & 0 \\ 0 & \sum_{k=0}^{\infty} \frac{(-1)^k (qt)^{2k}}{(2k)!} \end{pmatrix} + \begin{pmatrix} 0 & \sum_{k=0}^{\infty} \frac{(-1)^k (qt)^{2k+1}}{(2k+1)!} \\ \sum_{k=0}^{\infty} \frac{(-1)^{k+1} (qt)^{2k+1}}{(2k+1)!} & 0 \end{pmatrix}. \end{aligned}$$

Marco teórico

Ahora, recordando los desarrollos de Taylor del seno y el coseno, sabemos que

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \quad \text{y} \quad \cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \quad \text{para todo } x \in \mathbb{R},$$

con lo que

$$e^{Qt} = \begin{pmatrix} \cos(qt) & \sin(qt) \\ -\sin(qt) & \cos(qt) \end{pmatrix}.$$

□

Veamos cómo expresar $e^{Bt} = e^{(D+Q)t}$.

Proposición 4. Sean D y Q dos matrices cuadradas de la misma dimensión y conmutan, es decir, $DQ = QD$, entonces $e^{(D+Q)t} = e^{Dt}e^{Qt}$.

Demostración. Sea $Z(t) = e^{(D+Q)t} - e^{Dt}e^{Qt}$.

Como D y Q conmutan, es claro que e^{Dt} y Q también. Entonces,

$$\begin{aligned} Z'(t) &= (D + Q)e^{(D+Q)t} - De^{Dt}e^{Qt} - e^{Dt}Qe^{Qt} \\ &= (D + Q)(e^{(D+Q)t} - e^{Dt}e^{Qt}) \\ &= (D + Q)Z(t). \end{aligned}$$

Además, como $Z(0) = 0$, queda demostrado que $Z(t) = 0$ para todo $t \in \mathbb{R}$. Entonces, queda demostrado que $e^{(D+Q)t} = e^{Dt}e^{Qt}$. □

Entonces, como las matrices $D = \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix}$ y $Q = \begin{pmatrix} 0 & q \\ -q & 0 \end{pmatrix}$ cumplen que

$$DQ = \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix} \begin{pmatrix} 0 & q \\ -q & 0 \end{pmatrix} = \begin{pmatrix} pq & 0 \\ -pq & p \end{pmatrix} = \begin{pmatrix} 0 & q \\ -q & 0 \end{pmatrix} \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix} = QD,$$

podemos expresar e^{Bt} como:

$$\begin{aligned} e^{Bt} &= e^{(D+Q)t} = e^{Dt}e^{Qt} = \begin{pmatrix} e^{pt} & 0 \\ 0 & e^{pt} \end{pmatrix} \begin{pmatrix} \cos(qt) & \sin(qt) \\ -\sin(qt) & \cos(qt) \end{pmatrix} \\ &= \begin{pmatrix} e^{pt} \cos(qt) & e^{pt} \sin(qt) \\ -e^{pt} \sin(qt) & e^{pt} \cos(qt) \end{pmatrix}. \end{aligned}$$

Así,

$$\begin{aligned} e^{At} &= e^{PBP^{-1}t} = Pe^{Bt}P^{-1} \\ &= P \begin{pmatrix} e^{pt} \cos(qt) & e^{pt} \sin(qt) \\ -e^{pt} \sin(qt) & e^{pt} \cos(qt) \end{pmatrix} P^{-1}. \end{aligned}$$

Y por tanto, la solución general es de la forma

$$X(t) = e^{At}c = Pe^{Bt} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad \text{con } c_1, c_2 \in \mathbb{R},$$

o equivalentemente,

$$X(t) = c_1 e^{pt} (\cos(qt)u - \sin(qt)w) + c_2 e^{pt} (\sin(qt)u + \cos(qt)w), \quad \text{con } c_1, c_2 \in \mathbb{R}.$$

2.1.4. No diagonalizable. Autovalor real e igual

En dimensión 2, una matriz A no es diagonalizable únicamente cuando A tiene un autovalor λ doble con $\dim(\ker(A - \lambda I)) = 1$. Como la multiplicidad algebraica no coincide con la geométrica, A no es diagonalizable, lo que quiere decir que no podemos escribir A como $A = PDP^{-1}$ con D alguna matriz diagonal. Por tanto, utilizaremos una matriz que, aunque no es una matriz diagonal, es lo suficientemente sencilla como para poder usarla cómodamente, que denotaremos como la matriz de Jordan J y cumple que $A = PJP^{-1}$.

La cadena de Jordan asociada al autovalor λ de longitud 2 se halla de la siguiente manera (véase [6]):

$$\begin{cases} v_2 \in \ker((A - \lambda I)^2) \quad \text{y} \quad v_2 \notin \ker(A - \lambda I), \\ v_1 = (A - \lambda I)v_2 \in \ker(A - \lambda I) \setminus \{0\}, \end{cases}$$

y por consiguiente,

$$\begin{cases} Av_1 = \lambda v_1, \\ Av_2 = \lambda v_2 + v_1. \end{cases}$$

Se cumple que $\{v_1, v_2\}$ son linealmente independientes, y que la forma canónica de Jordan de A es de la manera $A = PJP^{-1}$, con

$$P = \begin{pmatrix} & & | & \\ v_1 & v_2 & | & \\ & & | & \end{pmatrix} \quad \text{y} \quad J = \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}.$$

Entonces, con la forma canónica de Jordan de A , tenemos que

$$e^{At} = e^{PJP^{-1}} = Pe^{Jt}P^{-1} = Pe^{\lambda t} \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix} P^{-1},$$

con lo que $\phi(t) = Pe^{Jt}$ es una matriz fundamental del sistema. Entonces, la solución general del sistema $X' = AX$ es de la forma

$$X(t) = Pe^{Jt} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = e^{\lambda t} P \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \quad \text{con } c_1, c_2 \in \mathbb{R},$$

que es equivalente a

$$X(t) = e^{\lambda t} (c_1 v_1 + c_2 (v_1 t + v_2)) \quad \text{con } c_1, c_2 \in \mathbb{R}.$$

2.2. Clasificación de órbitas y puntos por autovalores

Se denomina diagrama de fases a la representación en el plano xy del campo vectorial asociado al sistema y de algunas de sus órbitas. El estudio de los diagramas de fases es útil porque nos permite examinar de manera cualitativa el sistema planteado sin obtener las soluciones explícitas, observando la relación entre las dos variables.

Como mencionamos al principio del trabajo, nos hemos centrado en aquellos sistemas de coeficientes reales cuyo determinante de la matriz de coeficientes es distinto del cero. Dependiendo de si la matriz es diagonalizable y de cómo sean los autovalores λ_1, λ_2 de la matriz de coeficientes A , el diagrama de fases tiene una forma u otra. En las próximas secciones estudiaremos qué forma tienen, separando los siguientes casos:

1. Dos autovalores reales diferentes.

- Nodo estable: $\lambda_1, \lambda_2 \in \mathbb{R}$ y $\lambda_2 < \lambda_1 < 0$.
- Nodo inestable: $\lambda_1, \lambda_2 \in \mathbb{R}$ y $0 < \lambda_2 < \lambda_1$.
- Punto de silla: $\lambda_1, \lambda_2 \in \mathbb{R}$ y $\lambda_2 < 0 < \lambda_1$.

2. Autovalor doble (reales).

- Punto estelar: diagonalizable.
- Nodo impropio: no diagonalizable.

3. Autovalores complejos.

- Centro estable: $\lambda_1, \lambda_2 \in \mathbb{C}$ y $\operatorname{Re}(\lambda_1) = \operatorname{Re}(\lambda_2) = 0$.
- Foco estable: $\lambda_1, \lambda_2 \in \mathbb{C}$ y $\operatorname{Re}(\lambda_1) = \operatorname{Re}(\lambda_2) < 0$.
- Foco inestable: $\lambda_1, \lambda_2 \in \mathbb{C}$ y $\operatorname{Re}(\lambda_1) = \operatorname{Re}(\lambda_2) > 0$.

Notamos que en una matriz de dimensión 2 y existen sus autovalores, estos serán o los dos reales o los dos complejos, y que solo aquellos autovalores con multiplicidad mayor que 1 pueden ser no diagonalizables. Por lo que están cubiertos todos los casos (véase [8]).

2.2.1. Autovalores reales y distintos

Sea el sistema $X' = AX$, donde la matriz de coeficientes A tiene dos autovalores λ_1, λ_2 reales distintos. Entonces, como los autovalores son simples (multiplicidad 1), A es diagonalizable. Así, A se puede expresar como $A = PDP^{-1}$ con $D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$.

Introduciendo el cambio de variable:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = P \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix} \quad \text{o equivalentemente,} \quad P^{-1} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix}, \quad (2.4)$$

2.2. Clasificación de órbitas y puntos por autovalores

el sistema es $X' = AX$ es equivalente a $X' = PDP^{-1}X$. Multiplicando por la izquierda P^{-1} a ambos lados de la ecuación, obtenemos que $P^{-1}X' = DP^{-1}X$, donde P es la matriz cuyas columnas son los vectores propios v_1, v_2 asociados a λ_1, λ_2 . Obtenemos como resultado un sistema que denominaremos como **sistema canónico**:

$$\begin{pmatrix} \xi'_1(t) \\ \xi'_2(t) \end{pmatrix} = D \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix},$$

cuyas soluciones son de la forma

$$\begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix} = \begin{pmatrix} e^{\lambda_1 t} & 0 \\ 0 & e^{\lambda_2 t} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = e^{Dt} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad c_1, c_2 \in \mathbb{R}.$$

Entonces, las ecuaciones de las órbitas de este sistema canónico son de la forma

$$\begin{cases} \xi_1(t) = c_1 e^{\lambda_1 t}, \\ \xi_2(t) = c_2 e^{\lambda_2 t}, \end{cases} \quad (2.5)$$

en el que dependiendo de los valores de c_1, c_2 diferenciamos los siguientes casos:

1. Si $c_1 = c_2 = 0$, $\xi = (0, 0)$ es el único punto de equilibrio.

2. Si $c_1 = 0$,

$$\begin{cases} \xi_1(t) = 0, \\ \xi_2(t) = c_2 e^{\lambda_2 t}, \end{cases}$$

los puntos de equilibrio están formados por el semieje de ordenadas ξ_2 positivo cuando $c_2 > 0$ y el semieje de ordenadas ξ_2 negativo cuando $c_2 < 0$.

3. Si $c_2 = 0$,

$$\begin{cases} \xi_1(t) = c_1 e^{\lambda_1 t}, \\ \xi_2(t) = 0, \end{cases}$$

los puntos de equilibrio están formados por el semieje de abscisas ξ_1 positivo cuando $c_1 > 0$ y el semieje de abscisas ξ_1 negativo cuando $c_1 < 0$.

4. Si $c_1 \neq 0$ y $c_2 \neq 0$, despejamos t de la primera ecuación de (2.5), obteniendo que:

$$\begin{aligned} \xi_1 &= c_1 e^{\lambda_1 t} \\ \ln(\xi_1) &= \ln(c_1 e^{\lambda_1 t}) \\ \ln(\xi_1) &= \ln(c_1) + \lambda_1 t \\ t &= \frac{1}{\lambda_1} \cdot \ln\left(\frac{\xi_1}{c_1}\right). \end{aligned}$$

Sustituyendo en la segunda ecuación de (2.5), resulta que

$$\xi_2 = c_2 \left(\frac{\xi_1}{c_1} \right)^{\frac{\lambda_2}{\lambda_1}}.$$

Marco teórico

Por simplicidad, denotaremos $r = \frac{\lambda_1}{\lambda_2}$ y $c = \frac{c_2}{c_1}$ para expresar la ecuación anterior como:

$$\xi_2 = c\xi_1^r. \quad (2.6)$$

Dependiendo del signo de los autovalores, el punto origen será un nodo estable, un nodo inestable o un punto de silla.

2.2.1.1. Nodo estable

Si $\lambda_2 < \lambda_1 < 0$ y $r = \frac{\lambda_2}{\lambda_1} > 1$, el punto origen $(0, 0)$ es un **nodo estable** y se verifican las siguientes características:

- Puesto que $\lambda_1, \lambda_2 < 0$, todas las órbitas se acercan al punto $(0, 0)$ cuando t tiende al infinito. Esto se ve mediante las siguientes ecuaciones:

$$\lim_{t \rightarrow \infty} \xi_1(t) = \lim_{t \rightarrow \infty} c_1 e^{\lambda_1 t} = 0 \quad \text{y} \quad \lim_{t \rightarrow \infty} \xi_2(t) = \lim_{t \rightarrow \infty} c_2 e^{\lambda_2 t} = 0.$$

- Las órbitas descritas en (2.6), $\xi_2 = c\xi_1^r$, tienen aspecto de semiparábolas con el eje ξ_2 como eje de la parábola. Como $r > 1$, tenemos que

$$\frac{d\xi_2}{d\xi_1} = r c \xi_1^{r-1} \quad \text{y} \quad \lim_{\xi_1 \rightarrow 0} \frac{d\xi_2}{d\xi_1} = 0.$$

Las órbitas se acercan al origen cuando $t \rightarrow \infty$ tangencialmente al eje de abscisas ξ_1 .

Finalmente, recordamos las ecuaciones de (2.4) y procedemos a hacer un **cambio de variable**:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = P \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix} \quad \text{con} \quad P = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix},$$

siendo $v_1 = \begin{pmatrix} v_{11} \\ v_{21} \end{pmatrix}$ y $v_2 = \begin{pmatrix} v_{12} \\ v_{22} \end{pmatrix}$ asociados a los autovalores λ_1, λ_2 . A continuación, estudiaremos en qué se transforman el punto $(0, 0)$, y en qué se convierten los ejes ξ_1, ξ_2 .

- El punto $(0, 0)$ en las coordenadas de (ξ_1, ξ_2) se transforma en $(0, 0)$ en los ejes (x, y) .
- El eje ξ_1 que tiene los puntos de la forma $(\alpha, 0)$ se transforma en

$$\begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} \begin{pmatrix} \alpha \\ 0 \end{pmatrix} = \alpha \begin{pmatrix} v_{11} \\ v_{21} \end{pmatrix} = \alpha v_1,$$

de manera que en las coordenadas originales (x, y) , obtenemos el subespacio propio $\mathcal{L}(v_1)$. Esta recta, sin el origen, está formada por dos órbitas (semiejes) que tienden al origen cuando $t \rightarrow \infty$.

2.2. Clasificación de órbitas y puntos por autovalores

- El eje ξ_2 que tiene los puntos de la forma $(0, \alpha)$ se transforma en

$$\begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} \begin{pmatrix} 0 \\ \alpha \end{pmatrix} = \alpha \begin{pmatrix} v_{12} \\ v_{22} \end{pmatrix} = \alpha v_2,$$

de manera que en las coordenadas originales (x, y) , obtenemos el subespacio propio $\mathcal{L}(v_2)$. Esta recta, sin el origen, está formada por dos órbitas (semiejes) que tienden al origen cuando $t \rightarrow \infty$.

En la Figura 2.1 se muestra el diagrama de fases de un sistema descrito por las ecuaciones diferenciales:

$$\begin{cases} x' = -3x + y, \\ y' = x - 3y. \end{cases}$$

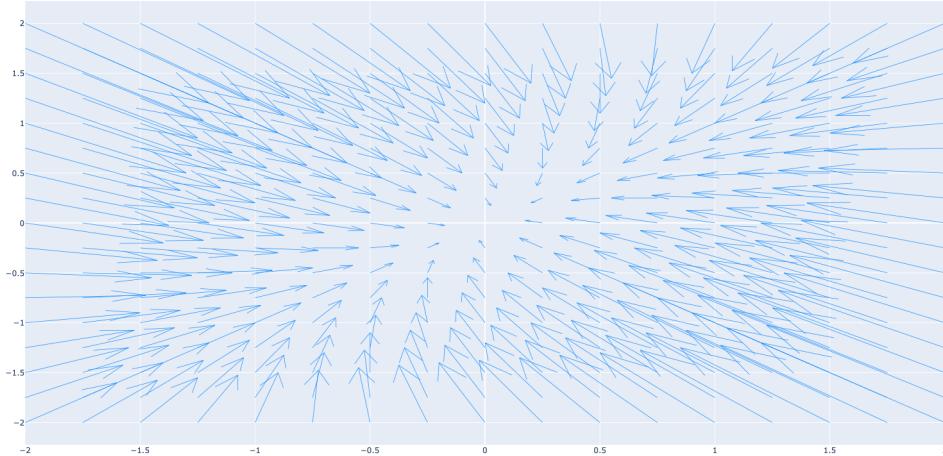


Figura 2.1: Ejemplo nodo estable, $x' = -3x + y$, $y' = x - 3y$

En el diagrama de fases, se representa gráficamente la evolución de las variables del sistema en función de sus tasas de cambio. En este caso, se observa que todas las flechas en el diagrama apuntan hacia el punto de equilibrio en el origen. Esto indica que todas las soluciones cercanas al punto de equilibrio se acercan a él a medida que el tiempo avanza.

El hecho de que todas las flechas se acerquen al origen en el diagrama de fases indica que el punto de equilibrio en el origen es un nodo estable. Además, cuanto más cerca esté una solución del origen, más rápido se acercará a él.

2.2.1.2. Nodo inestable

Si $0 < \lambda_2 < \lambda_1$, el punto $(0,0)$ se conoce como **nodo inestable**. Las órbitas son del mismo tipo que las del nodo estable (ver Sección 2.2.1.1), pero en este caso, todas las órbitas se alejan del origen a medida que t tiende a infinito.

Marco teórico

En la Figura 2.2 representamos el diagrama de fases para un sistema cuyo punto de equilibrio es un nodo inestable. El sistema está descrito por las ecuaciones diferenciales:

$$\begin{cases} x' = x, \\ y' = 2y. \end{cases}$$

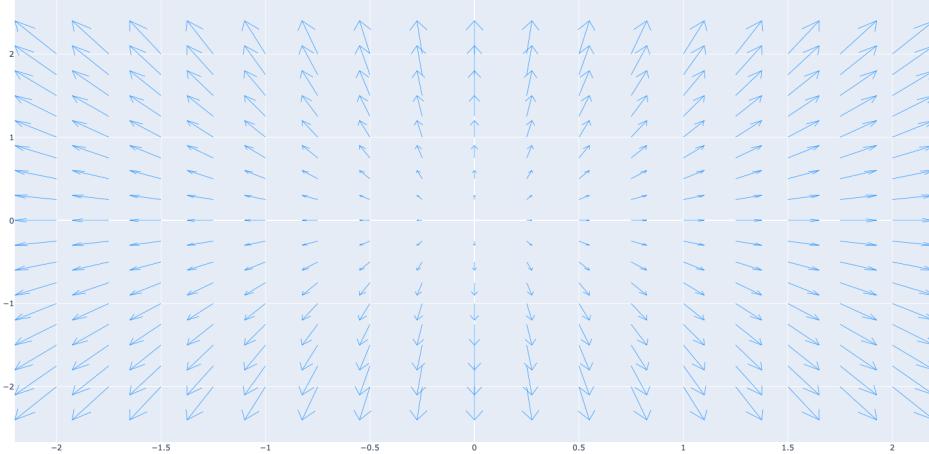


Figura 2.2: Ejemplo nodo inestable, $x' = x$, $y' = 2y$

El hecho de que todas las flechas se alejen del origen en el diagrama de fases indica que el punto de equilibrio en el origen es un nodo inestable. Es decir, cualquier perturbación en las variables x e y cuando estamos en el punto de equilibrio $(0,0)$ hará que la correspondiente solución del sistema se aleje del origen y no regrese a él.

2.2.1.3. Punto de silla

Denominamos **punto de silla** al punto origen $(0,0)$ cuando $\lambda_2 < 0 < \lambda_1$ y por tanto, $r = \frac{\lambda_2}{\lambda_1} < 0$. Las órbitas del sistema en coordenadas (ξ_1, ξ_2) satisfacen que:

$$\lim_{t \rightarrow \infty} \xi_1(t) = \lim_{t \rightarrow \infty} c_1 e^{\lambda_1 t} = \begin{cases} \infty, & \text{si } c_1 > 0 \\ -\infty, & \text{si } c_1 < 0 \end{cases}$$

y por otro lado,

$$\lim_{t \rightarrow \infty} \xi_2(t) = \lim_{t \rightarrow \infty} c_2 e^{\lambda_2 t} = 0.$$

- Las dos órbitas rectilíneas sobre el eje ξ_1 se alejan del origen cuando $t \rightarrow \infty$.
- Las dos órbitas rectilíneas sobre el eje ξ_2 tienden al origen cuando $t \rightarrow \infty$.
- Las órbitas $\xi_2 = c\xi_1^r$ tienen aspecto de ramas de hipérbolas con asíntotas $\xi_1 = 0$ y $\xi_2 = 0$.

2.2. Clasificación de órbitas y puntos por autovalores

Sean v_1, v_2 los vectores propios asociados a los autovalores λ_1, λ_2 , respectivamente. Las soluciones son

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2, \quad c_1, c_2 \in \mathbb{R}.$$

Como $\lambda_2 < 0 < \lambda_1$, en las coordenadas (x, y) obtenemos que:

- Si $c_1 = c_2 = 0$, entonces $(x, y) = (0, 0)$ es el único punto de equilibrio.
- Si $c_1 = 0$ y $c_2 \neq 0$, las órbitas son los dos semiejes de $\mathcal{L}(v_2)$, eliminando el origen, que tienden al origen cuando $t \rightarrow \infty$.
- Si $c_2 = 0$ y $c_1 \neq 0$, las órbitas son los dos semiejes de $\mathcal{L}(v_1)$, eliminando el origen, que se alejan del origen cuando $t \rightarrow \infty$.
- Si $c_1, c_2 \neq 0$, las órbitas toman el aspecto de ramas de hipérbolas, que tienden asintóticamente a una de las órbitas rectilíneas. Estas son los semiejes de $\mathcal{L}(v_1)$ cuando $t \rightarrow \infty$ y los semiejes de $\mathcal{L}(v_2)$ cuando $t \rightarrow -\infty$.

La Figura 2.3 representa el diagrama de fases para un sistema cuyo punto de equilibrio es un punto de silla, descrito por las ecuaciones diferenciales:

$$\begin{cases} x' = 2x + y, \\ y' = -y. \end{cases}$$

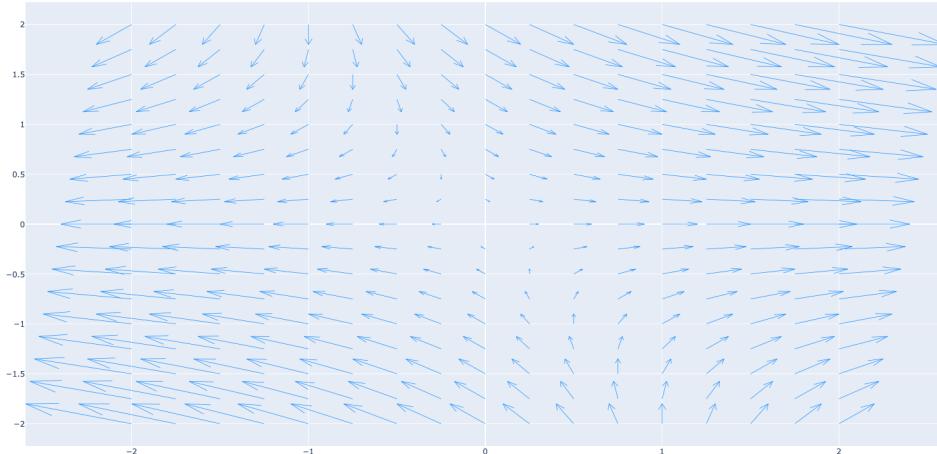


Figura 2.3: Ejemplo punto de silla, $x' = 2x + y, \quad y' = -y$

2.2.2. Autovalor doble

En el sistema $X' = AX$, si la matriz A posee un autovalor doble, podemos distinguir entre dos casos: cuando A es diagonalizable y cuando no lo es. En el primer caso, el punto de equilibrio se conoce como punto estelar, y las órbitas se asemejan a las de un nodo estable o inestable, dependiendo de si el autovalor es negativo o positivo, respectivamente.

Marco teórico

Por otro lado, en el segundo caso, cuando A no es diagonalizable y tiene un autovalor doble real, el punto de equilibrio se llama nodo impropio. En este caso, las órbitas se acercan o alejan del origen tangencialmente a ciertos ejes. A continuación, describiremos con más detalle cada uno de estos casos y cómo se presentan las órbitas y los diagramas de fases.

2.2.2.1. Punto estelar

Como vimos anteriormente en la Sección 2.1.2, para que la matriz del sistema $X' = AX$ sea diagonalizable y tenga un autovalor doble λ , el sistema debe tener la siguiente forma:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix},$$

y la solución general del sistema es:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} c_1 e^{\lambda t} \\ c_2 e^{\lambda t} \end{pmatrix}.$$

Entonces, se cumple que:

- Si $c_1 = c_2 = 0$, el origen es el único punto de equilibrio.
- Si $c_1 = 0$ y $c_2 \neq 0$, las órbitas son los dos semiejes que se obtienen al eliminar el origen del eje vertical.
- Si $c_1 \neq 0$, las órbitas son las semirrectas que se obtienen al eliminar el origen de las rectas $y = \frac{c_2 x}{c_1}$.

Si $\lambda < 0$, todas las órbitas se acercan al origen cuando t tiende al infinito, y se alejan cuando $\lambda > 0$.

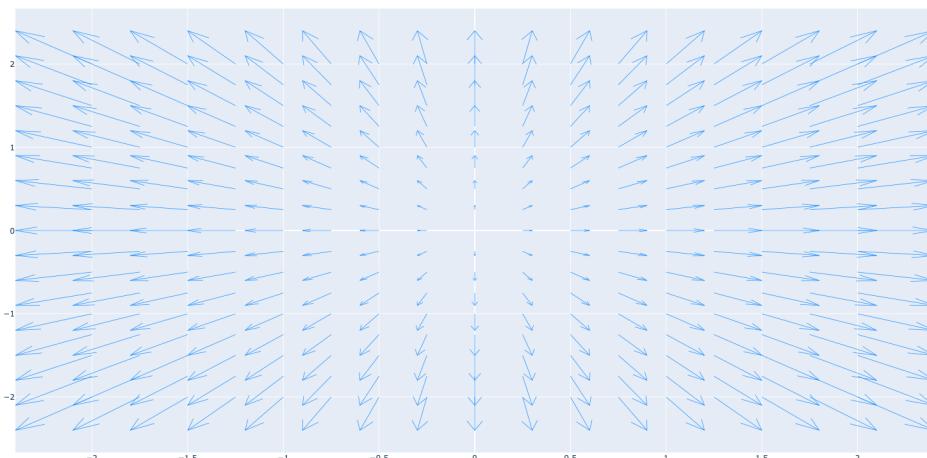


Figura 2.4: Ejemplo punto estelar, $x' = 2x$, $y' = 2y$

En la Figura 2.4 se ha mostrado el diagrama de fases de un sistema cuyo punto

2.2. Clasificación de órbitas y puntos por autovalores

de origen es un punto estelar. El sistema que se ilustraba en la figura era el siguiente:

$$\begin{cases} x' = 2x, \\ y' = 2y. \end{cases}$$

Observamos que en este caso el punto estelar tiene un autovalor doble y positivo, ya que todas las flechas del diagrama de fases se alejan del origen. Este tipo de punto estelar se asemeja a un nodo inestable, donde las órbitas se alejan del origen en el tiempo.

2.2.2.2. Nodo impropio

Sea el sistema $X' = AX$ en el que A tiene un autovalor doble λ y resulta que $\dim(\ker(A - \lambda I)) = 1$. Puesto que la multiplicidad algebraica y geométrica no coinciden, la matriz de coeficientes A no es diagonalizable. En este caso, se utilizan dos vectores propios linealmente independientes y una matriz de transición para llevar la matriz A a su forma canónica de Jordan.

Como ya vimos en la sección de solución general, podemos expresar la matriz de coeficientes A como:

$$A = PJP^{-1}, \quad \text{con} \quad J = \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}. \quad (2.7)$$

Introduciendo el cambio de variable $\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = P \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix}$, siendo P la matriz que tiene por columnas los vectores v_1, v_2 de una cadena de Jordan:

$$\begin{cases} v_2 \in (\ker(A - \lambda I))^2 \setminus \ker(A - \lambda I), \\ v_1 = (A - \lambda I)v_2 \in (\ker(A - \lambda I)) \setminus \{0\}, \end{cases} \quad (2.8)$$

y obtenemos el sistema

$$X' = PJP^{-1}X \quad \text{o, equivalentemente,} \quad P^{-1}X' = JP^{-1}X.$$

Llamaremos a estas ecuaciones sistema canónico equivalente al sistema

$$\begin{pmatrix} \xi'_1(t) \\ \xi'_2(t) \end{pmatrix} = J \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix},$$

cuyas soluciones son de la forma

$$\begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix} = e^{Jt}c = \begin{pmatrix} e^{\lambda t} & te^{\lambda t} \\ 0 & e^{\lambda t} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad c_1, c_2 \in \mathbb{R},$$

que es equivalente al sistema

$$\begin{cases} \xi_1(t) = e^{\lambda t}(c_1 + tc_2), \\ \xi_2(t) = c_2e^{\lambda t}. \end{cases} \quad (2.9)$$

Diferenciamos los siguientes casos para las órbitas:

Marco teórico

- $c_1 = c_2 = 0$, el origen es el único punto de equilibrio.
- Si $c_2 = 0$ y $c_1 \neq 0$, obtenemos el semieje de abscisas ξ_1 positivo cuando $c_1 > 0$ y el semieje de abscisas ξ_1 negativo si $c_1 < 0$.
- Si $c_2 \neq 0$, eliminando la variable t en la segunda ecuación de (2.9), $t = \frac{1}{\lambda} \ln \left(\frac{\xi_2}{c_2} \right)$ y, sustituyendo en la primera ecuación, obtenemos que:

$$\begin{aligned}\xi_1 &= e^{\lambda \frac{1}{\lambda} \ln \left(\frac{\xi_2}{c_2} \right)} \left(c_1 + \frac{1}{\lambda} \ln \left(\frac{\xi_2}{c_2} \right) c_2 \right) \\ \xi_1 &= \frac{\xi_2}{c_2} \left(c_1 + \frac{c_2}{\lambda} \ln \left(\frac{\xi_2}{c_2} \right) \right).\end{aligned}$$

Entonces, suponiendo que $\xi'_2(t) \neq 0$, podemos aplicar la regla de la cadena:

$$\frac{d\xi_1}{d\xi_2} = \frac{d\xi_1}{dt} \frac{dt}{d\xi_2} = \frac{d\xi_1}{dt} \frac{1}{\frac{d\xi_2}{dt}} = \frac{\lambda e^{\lambda t} (c_1 + tc_2) + c_2 e^{\lambda t}}{\lambda c_2 e^{\lambda t}} = \frac{\lambda \xi_1 + \xi_2}{\lambda \xi_2},$$

de manera que la tangente vertical se calcula igualando $\frac{d\xi_1}{d\xi_2} = 0$. Operando, obtenemos que $\xi_2 = -\lambda \xi_1$.

Similarmente,

$$\frac{d\xi_2}{d\xi_1} = \frac{\lambda c_2}{(c_1 + tc_2) + c_2} \quad \text{y aplicando límites,} \quad \lim_{t \rightarrow \infty} \frac{d\xi_2}{d\xi_1} = \lim_{t \rightarrow \infty} \frac{\lambda c_2}{(c_1 + tc_2) + c_2} = 0.$$

Si $\lambda < 0$, entonces $\lim_{t \rightarrow \infty} \xi_1(t) = \lim_{t \rightarrow \infty} \xi_2(t) = 0$ y las órbitas se acercan al origen con tangente horizontal según el eje ξ_1 .

Por el contrario, si $\lambda > 0$, entonces $\lim_{t \rightarrow \infty} |\xi_1(t)| = \lim_{t \rightarrow \infty} |\xi_2(t)| = \infty$ y las órbitas se alejan del origen con tangente horizontal según el eje ξ_1 .

Entonces, deshaciendo el cambio de variable descrito en (2.7):

- Si $c_1 = c_2 = 0$, el origen es el único punto de equilibrio.
- Si $c_2 = 0$, las órbitas son los semiejes que se obtienen de $\mathcal{L}(v_1)$ eliminando el punto $(0, 0)$. Estas órbitas se acercan al origen si $\lambda < 0$ y se alejan si $\lambda > 0$.
- Si $c_2 \neq 0$, las órbitas tienen tangente vertical en $ax - by = 0$, siendo a, b coeficientes de la matriz $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

Si $\lambda < 0$, las órbitas se acercan al origen en dirección de la tangente horizontal según el eje $\mathcal{L}(v_1)$ y se alejan si $\lambda > 0$ con tangente horizontal según el mismo eje $\mathcal{L}(v_1)$.

La Figura 2.5 presenta el diagrama de fases para un sistema cuyo punto origen es un nodo impropio. El sistema está dado por las ecuaciones:

$$\begin{cases} x' = -4x + y, \\ y' = -x - 2y. \end{cases}$$

2.2. Clasificación de órbitas y puntos por autovalores

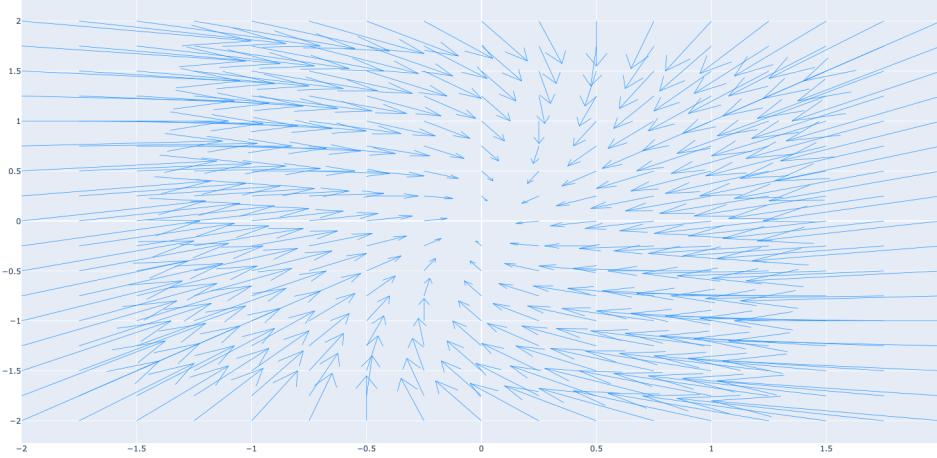


Figura 2.5: Ejemplo nodo impropio, $x' = -4x + y$, $y' = -x - 2y$

Observamos que todas las flechas se acercan al origen, pero a medida que se acercan al punto, giran en el sentido de las agujas del reloj. Esto indica que, a medida que las soluciones se acercan la origen, estas giran alrededor del origen antes de converger hacia él. Este comportamiento es muy diferente a los casos anteriores (nodo estable, nodo inestable, punto de silla, o punto estelar), y se debe a la forma canónica de la matriz A del sistema.

2.2.3. Autovalores complejos

Sea el sistema de 2 ecuaciones diferenciales lineales autónomo $X' = AX$, en el que los autovalores de A son los complejos $\mu = p + iq$ y $\bar{\mu} = p - iq$, con $q > 0$, y sea $v \in \ker(A - (p + iq)I)$ el autovector asociado. Definiendo $u = \operatorname{Re}(v)$ y $w = \operatorname{Im}(v)$, podemos expresar la matriz de coeficientes A como:

$$A = PBP^{-1}, \quad \text{siendo} \quad B = \begin{pmatrix} p & q \\ -q & p \end{pmatrix} \quad \text{y} \quad P = \begin{pmatrix} u_1 & w_1 \\ u_2 & w_2 \end{pmatrix}.$$

Introduciendo el cambio de variable

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = P \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix} \quad \text{que es equivalente a} \quad P^{-1} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix},$$

obtenemos el siguiente sistema de ecuaciones equivalente a $X' = AX$:

$$\begin{aligned} P^{-1} \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} &= P^{-1} P B P^{-1} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \\ \begin{pmatrix} \xi'_1(t) \\ \xi'_2(t) \end{pmatrix} &= B \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix} \\ \begin{pmatrix} \xi'_1(t) \\ \xi'_2(t) \end{pmatrix} &= \begin{pmatrix} p & q \\ -q & p \end{pmatrix} \begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix}. \end{aligned}$$

Marco teórico

Las soluciones son, como vimos en la Sección 2.1.3 sobre soluciones explícitas de sistemas con autovalores complejos, de la forma

$$\begin{pmatrix} \xi_1(t) \\ \xi_2(t) \end{pmatrix} = e^{pt} \begin{pmatrix} c_1 \cos(qt) + c_2 \sin(qt) \\ -c_1 \sin(qt) + c_2 \cos(qt) \end{pmatrix}, \quad c_1, c_2 \in \mathbb{R}.$$

Sea $c > 0$ y $\gamma \in (\pi, \pi]$, definimos

$$c = \sqrt{c_1^2 + c_2^2}, \quad \begin{cases} \cos(\gamma) = \frac{c_1}{c}, \\ \sin(\gamma) = \frac{c_2}{c}. \end{cases}$$

Entonces, sustituyendo las expresiones anteriores, y recordando las ecuaciones trigonométricas para las restas:

$$\begin{cases} \sin(\alpha - \beta) = \sin(\alpha) \cos(\beta) - \cos(\alpha) \sin(\beta), \\ \cos(\alpha - \beta) = \cos(\alpha) \cos(\beta) + \sin(\alpha) \sin(\beta), \end{cases}$$

obtenemos que:

$$\begin{aligned} \xi_1(t) &= e^{pt}(c_1 \cos(qt) + c_2 \sin(qt)) \\ &= e^{pt}(c \cos(\gamma) \cos(qt) + c \sin(\gamma) \sin(qt)) \\ &= e^{pt}c \cos(\gamma - qt), \end{aligned}$$

y que

$$\begin{aligned} \xi_2(t) &= e^{pt}(-c_1 \sin(qt) + c_2 \cos(qt)) \\ &= e^{pt}(-c \cos(\gamma) \sin(qt) + c \sin(\gamma) \cos(qt)) \\ &= e^{pt}c \sin(\gamma - qt). \end{aligned}$$

Entonces, de las dos anteriores expresiones obtenemos que

$$\begin{cases} \xi_1(t) = e^{pt}c \cos(\gamma - qt), \\ \xi_2(t) = e^{pt}c \sin(\gamma - qt). \end{cases}$$

Introducimos el cambio de variable a polares, siendo $t \in (-\infty, \infty)$:

$$\begin{cases} r(t) = ce^{pt} \\ \theta(t) = \gamma - qt \end{cases} \quad \text{de lo que se sigue que} \quad \begin{cases} \xi_1(t) = r(t) \cos(\theta(t)), \\ \xi_2(t) = r(t) \sin(\theta(t)), \end{cases}$$

y se cumple que

$$r(t) = \sqrt{(\xi_1(t))^2 + (\xi_2(t))^2}, \quad \tan(\theta(t)) = \frac{\xi_2}{\xi_1}.$$

Observamos que si $\theta(t)$ pertenece al intervalo $((2n-1)\pi, (2n+1)\pi] \subset (-\infty, \infty)$ con $n \in \mathbb{Z}$, entonces

$$\theta(t) - 2\pi n \in (-\pi, \pi].$$

2.2. Clasificación de órbitas y puntos por autovalores

Por tanto, $r(t)$ y $\theta(t)$ representan las coordenadas polares del punto $(\xi_1(t), \xi_2(t))$.

Despejando t en la ecuación $\theta = \gamma - qt$, tenemos

$$t = \frac{1}{q}(\gamma - \theta)$$

y sustituyendo en $r = ce^{pt}$, obtenemos

$$r = ce^{\frac{p}{q}(\gamma - \theta)}. \quad (2.10)$$

Estas curvas son circunferencias si $p = 0$ y en caso contrario, espirales logarítmicas. Además, cabe destacar que todas estas trayectorias se recorren en el sentido horario, ya que hemos considerado $q > 0$ y $\theta = \gamma - qt$.

Para clasificar el punto de equilibrio $(0,0)$ en el caso de autovalores complejos distinguiremos 3 casos, dependiendo del signo de p , la parte real del autovalor $\mu = p + iq$ y $\bar{\mu} = p - iq$.

- Centro estable: $p = 0$.
- Foco estable: $p < 0$.
- Foco inestable: $p > 0$.

2.2.3.1. Centro estable

Si $p = 0$, la ecuación (2.10) queda reducida a la forma $r = c$. Entonces, recordando que estamos coordinadas polares, las órbitas son circunferencias centradas en el origen y de radio c . Además, éstas son periódicas con periodo $\frac{2\pi}{q}$. En este caso decimos que el punto $(0,0)$ es un **centro estable**.

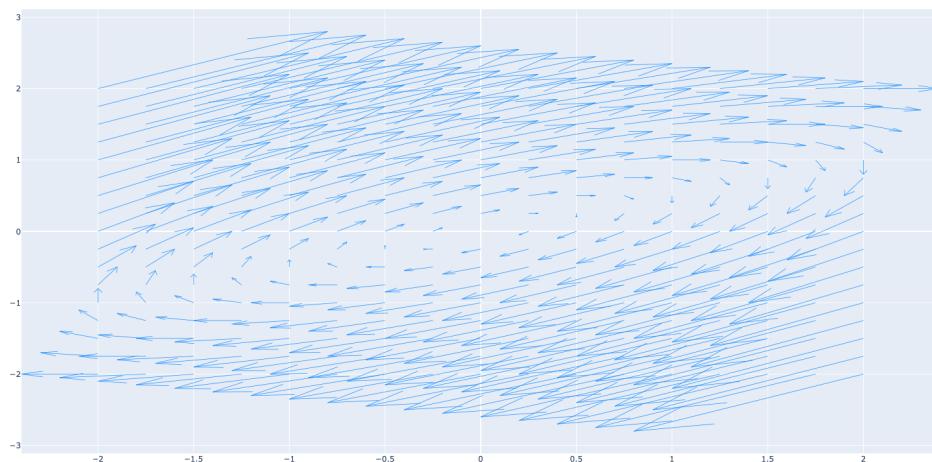


Figura 2.6: Ejemplo centro estable, $x' = -2x + 4y$, $y' = -2x - 2y$

La Figura 2.6 representa el diagrama de fases para un sistema cuyo punto origen

Marco teórico

es un centro estable. El sistema está dado por las siguientes ecuaciones.

$$\begin{cases} x' = -2x + 4y, \\ y' = -2x - 2y. \end{cases}$$

Observamos que las flechas describen curvas cerradas en forma de elipses centradas en el origen.

2.2.3.2. Foco estable

Si $p < 0$, entonces $\lim_{t \rightarrow \infty} r(t) = 0$ y las órbitas distintas del origen son espirales logarítmicas, que se mantienen cerca del origen para $t \rightarrow \infty$. Decimos entonces que el punto $(0,0)$ es un **foco estable**.

En la Figura 2.7, se muestra el diagrama de fases para un sistema cuyo punto origen es un foco estable. El sistema está dado por las ecuaciones diferenciales

$$\begin{cases} x' = -2x - 3y, \\ y' = 3x - 2y. \end{cases}$$

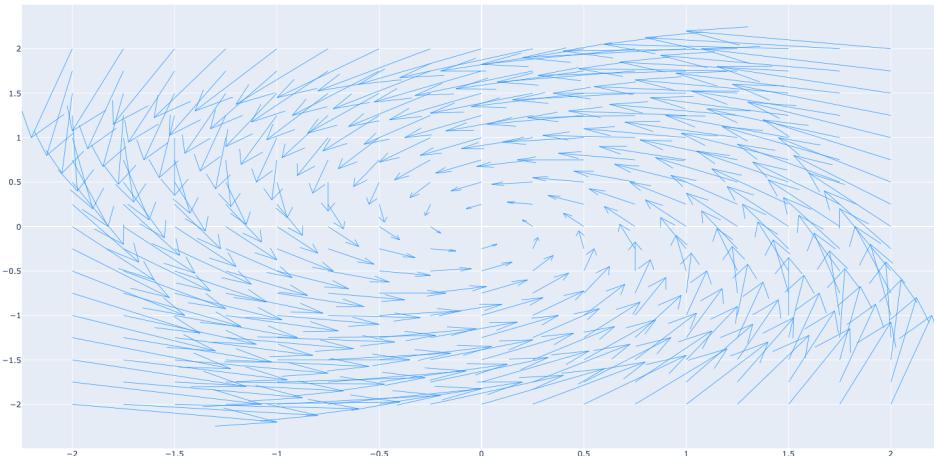


Figura 2.7: Ejemplo foco estable, $x' = -2x - 3y$, $y' = 3x - 2y$

Podemos observar que las flechas describen curvas suaves en espiral alrededor del origen. El punto $(0,0)$ es un punto de equilibrio estable, lo que quiere decir que cualquier perturbación del sistema en el origen provocará una evolución hacia el origen, a lo largo de las órbitas en espiral asintóticamente estables.

2.2.3.3. Foco inestable

Si $p > 0$, entonces $\lim_{t \rightarrow \infty} r(t) = \infty$ y las órbitas distintas del origen son espirales logarítmicas, que se alejan de un entorno del origen para $t \rightarrow \infty$. Decimos entonces que el punto $(0,0)$ es un **foco inestable**.

2.2. Clasificación de órbitas y puntos por autovalores

La Figura 2.8 muestra el diagrama de fases para un sistema cuyo punto origen es un foco inestable. El sistema está descrito por las ecuaciones diferenciales

$$\begin{cases} x' = 2x + 3y, \\ y' = -3x + 2y. \end{cases}$$

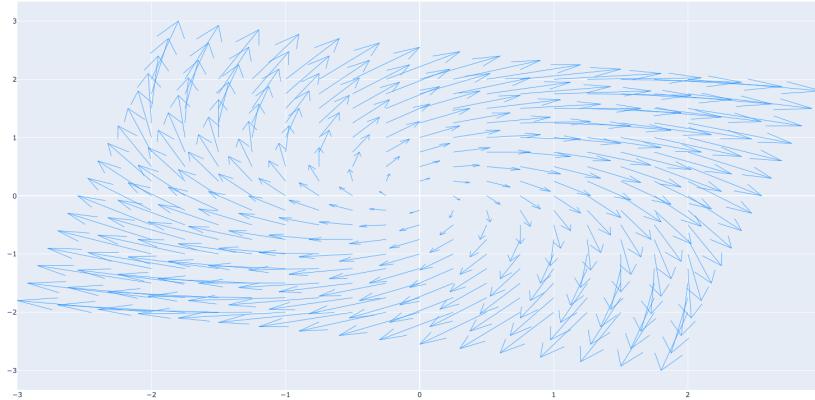


Figura 2.8: Ejemplo foco inestable, $x' = 2x + 3y$, $y' = -3x + 2y$

Podemos observar en La Figura 2.6 que todas las flechas del diagrama de fases se alejan del origen, lo que indica que el sistema se aleja del origen a medida que evoluciona.

Tabla resumen

A continuación, en la Tabla 2.1 se resumen los diferentes tipos de estabilidad en función de cómo son los autovalores (reales y diferentes, reales e iguales o complejos), el signo de los autovalores y si la matriz de coeficientes es diagonalizable.

Tipo de autovalores	Condiciones	Tipo de estabilidad
Reales y diferentes	$\lambda_2 < \lambda_1 < 0$	Nodo estable
	$0 < \lambda_2 < \lambda_1$	Nodo inestable
	$\lambda_2 < 0 < \lambda_1$	Punto de silla
Reales e iguales	Diagonalizable	Punto estelar
	No diagonalizable	Nodo impropio
Complejos	$\operatorname{Re}(\lambda_1) = \operatorname{Re}(\lambda_2) = 0$	Centro estable
	$\operatorname{Re}(\lambda_1) = \operatorname{Re}(\lambda_2) > 0$	Foco inestable
	$\operatorname{Re}(\lambda_1) = \operatorname{Re}(\lambda_2) < 0$	Foco estable

Tabla 2.1: Tabla resumen tipos de estabilidad en referencia a los autovalores

2.3. Clasificación de puntos por traza y determinante

En esta sección estudiaremos otra forma de clasificar los puntos de equilibrio, sin necesidad de buscar los autovalores. Determinaremos la estabilidad del sistema únicamente mediante la traza y determinante de la matriz de coeficientes.

Sea la matriz

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

cuyo polinomio característico asociado se obtiene resolviendo la ecuación

$$\begin{vmatrix} a - \lambda & b \\ c & d - \lambda \end{vmatrix} = 0 \quad \text{y desarrollando,} \quad \lambda^2 - (a + d)\lambda + (ad - bc) = 0.$$

Entonces, el polinomio característico $p(\lambda)$ asociado al autovalor λ se puede escribir como

$$p(\lambda) = \lambda^2 - (\text{Tr } A)\lambda + \det A,$$

siendo $\text{Tr } A$ y $\det A$ la traza y el determinante de la matriz A , respectivamente.

Entonces podemos hallar los autovalores λ_1, λ_2 resolviendo el polinomio característico mediante la fórmula cuadrática, obteniendo

$$\lambda_1 = \frac{\text{Tr } A + \sqrt{(\text{Tr } A)^2 - 4 \det A}}{2} \quad \text{y} \quad \lambda_2 = \frac{\text{Tr } A - \sqrt{(\text{Tr } A)^2 - 4 \det A}}{2}.$$

Observamos a continuación que la suma de autovalores es igual que la traza, y que el producto de autovalores es equivalente al determinante de la matriz:

$$\begin{aligned} \lambda_1 + \lambda_2 &= \frac{\text{Tr } A + \sqrt{(\text{Tr } A)^2 - 4 \det A}}{2} + \frac{\text{Tr } A - \sqrt{(\text{Tr } A)^2 - 4 \det A}}{2} \\ &= \frac{(\text{Tr } A + \text{Tr } A) + (\sqrt{(\text{Tr } A)^2 - 4 \det A} - \sqrt{(\text{Tr } A)^2 - 4 \det A})}{2} \\ &= \frac{2 \text{Tr } A}{2} = \text{Tr } A, \end{aligned}$$

$$\begin{aligned} \lambda_1 \cdot \lambda_2 &= \left(\frac{\text{Tr } A + \sqrt{(\text{Tr } A)^2 - 4 \det A}}{2} \right) \left(\frac{\text{Tr } A - \sqrt{(\text{Tr } A)^2 - 4 \det A}}{2} \right) \\ &= \frac{(\text{Tr } A)^2 - (\sqrt{(\text{Tr } A)^2 - 4 \det A})^2}{2^2} \\ &= \frac{(\text{Tr } A)^2 - (\text{Tr } A)^2 + 4 \det A}{4} \\ &= \frac{4 \det A}{4} = \det A. \end{aligned} \tag{2.11}$$

2.3. Clasificación de puntos por traza y determinante

Y también notaremos $T := \text{Tr } A$ y $D := \det A$.

Entonces, demostraremos que podemos clasificar la dinámica del correspondiente sistema diferencial estudiando los valores de la traza y el determinante de la matriz A .

Los autovalores se pueden expresar como

$$\lambda = \frac{T \pm \sqrt{T^2 - 4D}}{2}.$$

Estudiando el signo del discriminante, podemos sacar las siguientes conclusiones (véase [1]):

1. Si $T^2 - 4D > 0$, los autovalores son reales y distintos.
2. Si $T^2 - 4D = 0$, los autovalores son reales e iguales.
3. Si $T^2 - 4D < 0$, los autovalores son complejos (y diferentes).

2.3.1. Autovalores reales y distintos

Si $T^2 - 4D > 0$, diferenciamos los siguientes casos:

1. **Si $D < 0$, el punto de origen $(0, 0)$ es un punto de silla**

Si $D < 0$, $T^2 < T^2 - 4D$. Aplicando la raíz cuadrada a ambos lados de la expresión:

$$\sqrt{T^2} < \sqrt{T^2 - 4D},$$

que es equivalente a:

$$\pm T < \sqrt{T^2 - 4D}. \quad (2.12)$$

Esto es, por una parte,

$$T < \sqrt{T^2 - 4D} \quad \text{por lo que} \quad T - \sqrt{T^2 - 4D} < 0,$$

y por otro lado,

$$-T < \sqrt{T^2 - 4D} \quad \text{y despejando,} \quad T + \sqrt{T^2 - 4D} > 0.$$

Por tanto,

$$\lambda_1 = \frac{T - \sqrt{T^2 - 4D}}{2} < 0 \quad \text{y} \quad \lambda_2 = \frac{T + \sqrt{T^2 - 4D}}{2} > 0.$$

Entonces los autovalores tienen distinto signo, y por la Sección 2.2.1.3, concluimos que el origen es un punto de silla.

Otra manera de estudiar los signos de los autovalores, es recordando que, como demostramos en (2.11), el producto de los dos autovalores es igual a D , es decir, $\lambda_1 \cdot \lambda_2 = D$. Entonces, como $D < 0$, λ_1 y λ_2 tienen que tener distinto signo.

Marco teórico

2. Si $D > 0$ y $T < 0$, el punto de origen $(0, 0)$ es un nodo estable

Si $D > 0$, entonces $T^2 > T^2 - 4D$, obteniendo $|T| > \sqrt{T^2 - 4D}$. Entonces, como $T < 0$, se tiene que

$$T \pm \sqrt{T^2 - 4D} < 0.$$

Los dos autovalores son reales y negativos, entonces por la Sección 2.2.1.1, el punto $(0, 0)$ es un nodo estable.

3. Si $D > 0$ y $T > 0$, el punto de origen $(0, 0)$ es un nodo inestable

Si $D > 0$, entonces $T^2 > T^2 - 4D$ con lo cual se tiene que $|T| > \sqrt{T^2 - 4D}$. Entonces, como $T > 0$

$$T \pm \sqrt{T^2 - 4D} > 0.$$

Los dos autovalores son reales y negativos, entonces por la Sección 2.2.3.3, el punto $(0, 0)$ es un nodo inestable.

2.3.2. Autovalores reales e iguales

Si $T^2 - 4D = 0$, entonces el autovalor es doble y es de la forma $\lambda = \frac{T}{2}$.

Recordamos que para matrices 2×2 con autovalor doble, el punto de equilibrio $(0, 0)$ podía ser un punto estelar o un nodo impropio, dependiendo de si es diagonalizable o no (Sección 2.2.2). No podemos determinar qué tipo de punto de equilibrio es sabiendo únicamente el valor de la traza y el determinante.

Para este caso nos tendremos que fijar además en los valores de la matriz

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Como ya vimos anteriormente, A solo es diagonalizable con autovalor doble si A ya es una matriz diagonal, es decir, $b = c = 0$. Entonces,

1. Si $b = c = 0$, la matriz es diagonalizable y el punto $(0, 0)$ es un punto estelar (ver Sección 2.2.2.1).
2. Si $b \neq 0$ ó $c \neq 0$, la matriz no es diagonalizable y el punto $(0, 0)$ es un nodo impropio (ver Sección 2.2.1.2).

2.3.3. Autovalores complejos

Si $T^2 - 4D < 0$, los autovalores son complejos de la forma

$$\lambda = \frac{T \pm i\sqrt{-(T^2 - 4D)}}{2},$$

con

$$\operatorname{Re}(\lambda) = \frac{T}{2} \quad \text{y} \quad \operatorname{Im}(\lambda) = \frac{\pm\sqrt{-(T^2 - 4D)}}{2}.$$

2.3. Clasificación de puntos por traza y determinante

Entonces, por la Sección 2.2.3, estudiando el signo de la parte entera concluimos que:

1. Si $T = 0$, $\text{Re}(\lambda) = 0$ y el punto $(0, 0)$ es un centro estable.
2. Si $T > 0$, $\text{Re}(\lambda) > 0$ y el punto $(0, 0)$ es un foco inestable.
3. Si $T < 0$, $\text{Re}(\lambda) < 0$ y el punto $(0, 0)$ es un foco estable.

A continuación se muestra en la Figura 2.9 un resumen visual de las relaciones de traza y determinante. No se muestran las condiciones para encontrar un punto estelar y un nodo impropio, ya que hemos determinado que además de la traza y el determinante, era necesario estudiar los valores de los coeficientes. Estos dos puntos se encontrarían sobre la parábola $T^2 = 4D$.

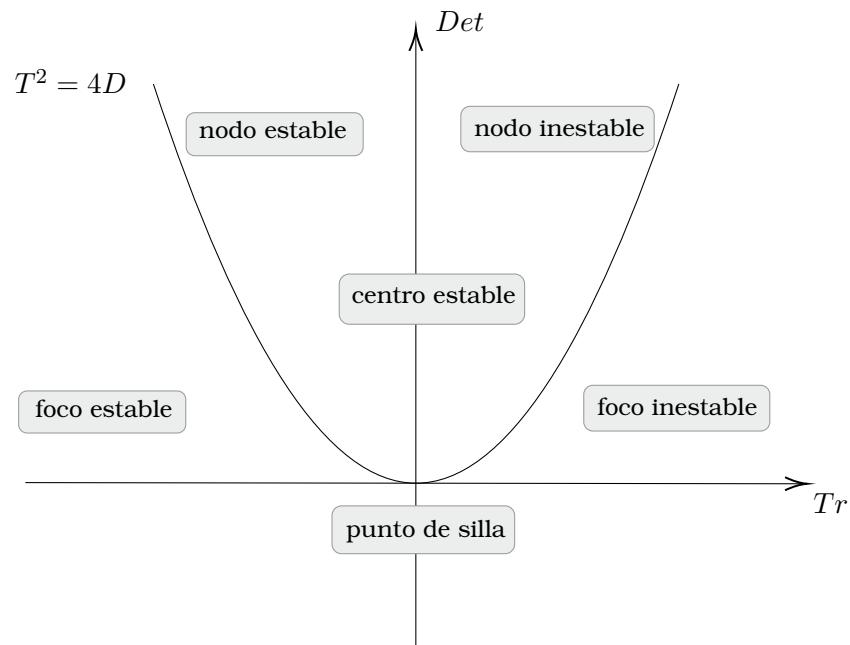


Figura 2.9: Plano traza-determinante

Tabla comparativa sobre la clasificación mediante autovalores y por traza-determinante

En la Tabla 2.2 se presenta una comparativa de las clasificaciones de los puntos de equilibrio de un sistema de dos ecuaciones diferenciales lineales de primer orden que tenga matriz de coeficientes con determinante no nulo. Para determinar la clasificación, se pueden utilizar diferentes criterios, como los valores de los autovalores o los signos de la traza y del determinante de la matriz de coeficientes.

Marco teórico

Tipo de autovalores	Discriminante	Condiciones traza y determinante	Condiciones autovalores	Tipo de estabilidad
Reales y diferentes	$T^2 - 4D > 0$	$D > 0$ y $T < 0$	$\lambda_2 < \lambda_1 < 0$	Nodo estable
		$D > 0$ y $T > 0$	$0 < \lambda_2 < \lambda_1$	Nodo inestable
		$D < 0$	$\lambda_2 < 0 < \lambda_1$	Punto de silla
Reales e iguales	$T^2 - 4D = 0$	$b = c = 0$	Diagonalizable	Punto estelar
		$b \neq 0$ ó $c \neq 0$	No diagonalizable	Nodo impropio
Complejos	$T^2 - 4D < 0$	$T = 0$	$\text{Re}(\lambda_1) = \text{Re}(\lambda_2) = 0$	Centro estable
		$T > 0$	$\text{Re}(\lambda_1) = \text{Re}(\lambda_2) > 0$	Foco inestable
		$T < 0$	$\text{Re}(\lambda_1) = \text{Re}(\lambda_2) < 0$	Foco estable

Tabla 2.2: Tabla con ambas clasificaciones

2.4. Clasificación dinámica

En este sección, daremos una clasificación más dinámica de los sistemas de primer orden lineales.

Desde la perspectiva de los sistemas dinámicos, es importante comprender el comportamiento de las soluciones de las ecuaciones diferenciales a largo plazo. Decimos que dos soluciones son equivalentes si tienen el mismo destino.

Para destacar la dependencia de las soluciones del tiempo y las condiciones iniciales X_0 , se utilizará la notación $\phi_t(X_0)$ para denotar la solución del sistema $X' = AX$ que parte de la condición inicial X_0 , es decir, $\phi_0(X_0) = X_0$. La función $\phi(t, X_0) = \phi_t(X_0)$ se denomina el **flujo** de la ecuación diferencial, y ϕ_t se conoce como la **aplicación** del flujo en el tiempo. Por lo tanto, el flujo es una función que depende tanto del tiempo como de los valores iniciales (véase [1]).

Dos sistemas son **dinámicamente equivalentes** si existe una función h que relaciona los correspondientes flujos y que además, es un homeomorfismo, esto es, que sea una función biyectiva, continua y su inversa también es continua (véase [1]). Así, veremos la definición de conjugación topológica:

Definición 4 (Conjugación topológica). *Sean los sistemas $X' = AX$ y $X' = BX$ con los flujos ϕ^A y ϕ^B , respectivamente. Dichos sistemas son topológicamente conjugados si existe un homeomorfismo $h : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ tal que:*

$$\phi^B(t, h(X_0)) = h(\phi^A(t, X_0)).$$

El homeomorfismo h se denomina conjugación y lleva las curvas solución del sistema $X' = AX$ a las del sistema $X' = BX$.

La idea detrás de la conjugación topológica es transformar la ecuación diferencial original en otra ecuación más fácil de analizar mediante un homeomorfismo adecuado, y luego estudiar las propiedades de la ecuación conjugada para obtener información sobre la ecuación original.

2.5. Transformación de ecuación de segundo orden en sistema de primer orden

Definición 5 (Matriz hiperbólica). *Se dice que una matriz A es hiperbólica si ninguno de sus autovalores tiene parte real igual a cero. Además, denominaremos a los sistemas $X' = AX$ con matriz hiperbólica como sistemas hiperbólicos.*

Teorema 5. *Sean A_1, A_2 dos matrices hiperbólicas de orden 2. Los sistemas lineales $X' = A_1X$ y $X' = A_2X$ son conjugados si, y solo si, ambas matrices tienen el mismo número de autovalores con parte real negativa (véanse [1, 8]).*

La demostración de este teorema se deja en el Anexo A.

Así, dos sistemas se consideran conjugados si pertenecen a la misma categoría:

1. Un autovalor positivo y otro negativo.
2. Ambos autovalores positivos.
3. Ambos autovalores negativos.

2.5. Transformación de ecuación de segundo orden en sistema de primer orden

Hasta ahora hemos estudiado sistemas diferenciales lineales de primer orden cuya matriz de coeficientes tiene determinante no nulo. No obstante, si nos encontramos ante una ecuación diferencial de segundo orden homogénea, podemos transformarla en un sistema de dos ecuaciones lineales de primer orden y, siempre que se cumplan las condiciones necesarias (es decir, matriz de coeficientes con determinante no nulo...) entonces es posible aplicar todo lo anterior (véase [1]).

Sea la ecuación lineal de orden 2

$$ax'' + bx' + cx = 0 \quad \text{con } a, b, c \in \mathbb{R} \text{ y } a \neq 0. \quad (2.13)$$

Podemos transformar la ecuación en un sistema de 2 ecuaciones de primer orden. Para ello introducimos el cambio de variable

$$x' = y,$$

y despejando x'' en (2.13) obtenemos que

$$x'' = -\frac{c}{a}x - \frac{b}{a}x'.$$

Entonces, recordando que $x' = y$ y por tanto, $x'' = y'$, obtenemos el sistema autónomo:

$$\begin{cases} x' = y, \\ y' = -\frac{c}{a}x - \frac{b}{a}y, \end{cases}$$

resultando en un sistema autónomo homogéneo, con determinante distinto de 0 siempre que $c \neq 0$.

Capítulo 3

Código

En este capítulo, se detallará el código creado para el desarrollo del *Back* de la aplicación, en concreto de los contenidos desarrollados en el capítulo anterior. Los ejercicios que se resuelven en dicha aplicación están estructurados en tres categorías principales y se han implementado los algoritmos de resolución en base a la información obtenida en la Sección 2 de *Marco Teórico*. Los códigos están escritos en *Python* y se han utilizado algunas librerías externas, de las que hablaremos más tarde.

Los ejercicios que se resuelven actualmente en la aplicación son los siguientes:

1. Resolución de sistemas de dos ecuaciones diferenciales lineales de primer orden (cuya matriz de coeficientes tiene determinante no nulo) y ecuaciones diferenciales de segundo orden.
 - Obtener el Sistema Fundamental de Soluciones.
 - Encontrar la solución explícita del sistema.
 - Transformar una ecuación de segundo orden a un sistema de dos ecuaciones diferenciales de primer orden.
2. Clasificación y visualización de la estabilidad de los correspondientes sistemas diferenciales.
 - En función de los autovalores.
 - En función de la traza y el determinante.
 - Representación del diagrama de fases.
3. Determinación de si dos sistemas son conjugados topológicamente.

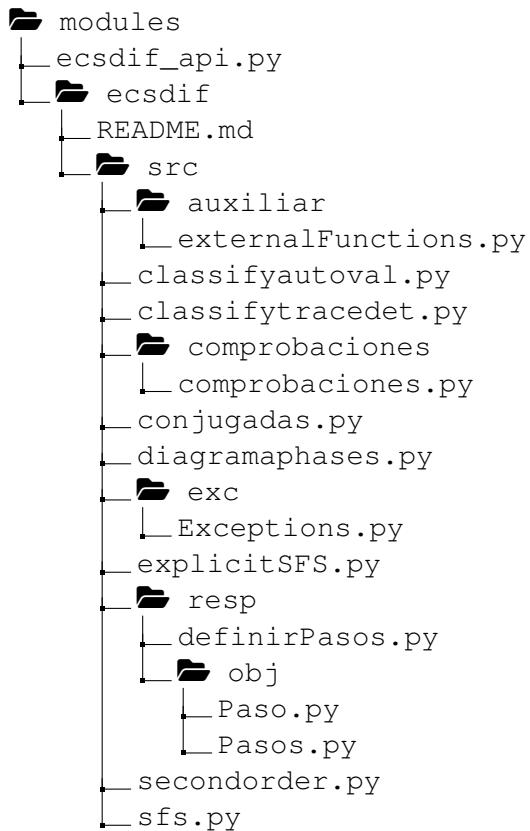
3.1. Estructura código

El código se ubica en el repositorio de GitHub denominado `Proyecto-Calculadora-Backend`. Este es el servidor de *backend* del Proyecto-Calculadora, que integra

3.1. Estructura código

las librerías creadas como parte de los TFGs de varios alumnos, incluido el mío, en la rama LJ-main.

A continuación se muestra el árbol de ficheros.



En la carpeta principal se encuentra el archivo `ecsdif_api.py`, que actúa como conector entre la interfaz de usuario y el motor que realiza los cálculos. En el directorio `ecsdif` se encuentran los archivos restantes, que contienen las funciones necesarias para realizar los ejercicios de la asignatura. En particular, esta carpeta está dividida en cuatro subcarpetas.

En la subcarpeta `comprobaciones` se encuentran las funciones auxiliares que verifican que los parámetros proporcionados al sistema sean válidos y no den lugar a resultados inesperados. Por ejemplo, se comprueba que el determinante no sea nulo y que los parámetros que se pasan como cadenas de texto puedan ser transformados en números. De esta manera, se evitan errores y se evita la duplicidad de código.

En el directorio `resp` se ubican las clases que generan los pasos para resolver los ejercicios en un formato específico. En particular, se encuentran las clases `Paso.py` y `Pasos.py`. Estas clases facilitan la comprensión de la resolución de los problemas y ayudan a visualizar los pasos realizados.

En `exc` se sitúan las clases que gestionan los distintos tipos de excepciones que pueden surgir durante la ejecución del código. Estas excepciones están diseñadas para ser descriptivas y ayudar al usuario a entender el problema que

Código

se ha producido. Algunas de estas excepciones son `ExceptionNotANumber` o `ExceptionDetZero`, que indican que los datos introducidos no son números, o que el determinante de la matriz asociada al sistema que se quiere resolver es nula.

Finalmente, en la subcarpeta auxiliar se localizan las funciones que utilizan librerías externas. Estas funciones son necesarias para realizar cálculos más complejos y para hacer posible la representación gráfica de los diagramas de fases.

Además de los archivos mencionados anteriormente, en la carpeta `ecsdif` se almacenan los ficheros que permiten resolver los diferentes tipos de ejercicios relacionados con la asignatura de Ecuaciones Diferenciales. Entre ellos, se encuentran los archivos para resolver ecuaciones explícitas, transformar ecuaciones de segundo orden a sistemas o representar diagramas de fases.

3.2. Conexión Front-End y Back-End

El archivo principal del proyecto es `ecsdif_api.py` y su función principal es establecer un vínculo entre el *Front-End* y el *Back-End*. Este fichero se encuentra en la carpeta `modules`, junto con la carpeta que contiene el resto de los documentos: `ecsdif`.

Para la conexión entre el *Front-End* y el *Back-End* se utiliza la librería `flask` de Python. Más específicamente, se usan los objetos `Blueprint` y `Response`.

`Blueprint` permite organizar las vistas de una aplicación web en módulos reutilizables, por lo que es posible crear uno para cada módulo de la aplicación (cada asignatura) y luego agregarlo a la aplicación principal. Más información sobre el funcionamiento de `flask` y `Blueprint` se puede encontrar en [9, 10].

```
1  from flask import Blueprint
2
3  bp = Blueprint("ecsdif", __name__, url_prefix="/ecsdif")
4
5  @bp.route("/info/sistemas", methods=['GET'])
6  def ecsdif_info_sistemas():
7      return {
8          "titulo": "Resolución de sistemas",
9          "descripcion": "Obtener el Sistema Fundamental de Soluciones y las\\
10             soluciones explícitas de sistemas.",
11         "algoritmo": "Se obtiene el SFS mediante el estudio de los\\
12             ↳ autovalores\\
13             y se recurre a estas para encontrar las soluciones explícitas."
14     }
```

Código fuente 1: Llamada a la página de información del contenido *Resolver*

El Código Fuente 1 muestra cómo se ha creado un `Blueprint` con el prefijo `/ecsdif` para redirigir cualquier ruta que comience con ese prefijo a llamadas

3.2. Conexión Front-End y Back-End

del archivo `ecsdif_api.py`.

Es importante destacar que además del prefijo de la ruta, también se especifica el resto de la ruta encima de cada función. En este caso, la página de presentación de la asignatura de Ecuaciones Diferenciales se puede acceder mediante la ruta `/ecsdif/info`.

Otro aspecto interesante es que todas las funciones definidas en el archivo `ecsdif_api.py` comienzan con el prefijo `ecsdif`. Esto se hace para evitar posibles conflictos si dos funciones de distintas asignaturas tuvieran el mismo nombre. Al añadir un prefijo a cada función, se asegura que el nombre completo sea único.

Por último, para las funciones que requieren parámetros, se utilizan cadenas de texto para pasar los valores a través de la URL. Por ejemplo, la función `ecsdif_get_sfs` requiere cuatro parámetros a, b, c y d , que corresponden a los coeficientes de un sistema de dos ecuaciones lineales con dos incógnitas. Estos parámetros se pasan como parte de la URL en el siguiente formato:

```
<param-1>&<param-2>&...&<param-n>.
```

```
1 @bp.route("/sist/sfs/<a>&<b>&<c>&<d>", methods=['GET'])
2 def ecsdif_get_sfs(a: str, b: str, c: str, d: str):
3     # resto de código ...
```

Código fuente 2: Cabecera de una llamada que recibe parámetros

Los parámetros se reciben sin los símbolos de menor y mayor. Por ejemplo, una posible petición a esta llamada, sería la siguiente: `/sist/sfs/1&3&2&2`.

En el desarrollo de la aplicación mencionada, es importante destacar que se lleva a cabo un proceso de conversión de parámetros, ya que estos se envían como cadenas de texto y es necesario transformarlos a números para poder realizar los cálculos pertinentes. Para ello, se utiliza la función `string2float`, implementada en el archivo `comprobaciones.py`, que se encarga de transformar una cadena de texto en un número, ya sea fracción o real, siempre y cuando sea posible. En caso contrario, se lanzaría una excepción.

Cabe mencionar que, para introducir fracciones en los parámetros, se utiliza el signo de dos puntos `:` en lugar de la barra inclinada `/`, ya que esta última se interpreta como una subruta y puede generar confusiones.

Por otro lado, al recibir una solicitud del cliente, se utiliza la clase `Response` para enviar una respuesta. En caso de que se produzca un error, se devuelve un código de estado 400 si el error se debe a una acción del cliente, como por ejemplo introducir letras en lugar de números, y un código de estado 500 si el error se debe a un fallo en el servidor.

En cambio, si todo ha ido correctamente, se devuelve un JSON que será interpretado por el front-end para ser mostrado al usuario de manera clara y visual. Este JSON consta de un diccionario con una única clave ‘pasos’, que contiene una lista de diccionarios. Cada uno de estos diccionarios incluye las claves ‘pa-

Código

so', 'pasoLatex' y 'explicación', que representan el cálculo en formato Python, el mismo cálculo en formato \LaTeX y una explicación del proceso, respectivamente.

Es importante destacar que, para convertir las operaciones en formato Python a \LaTeX , se han utilizado las librerías `latexifier` y `pytexit`, que permiten transformar los cálculos a un formato que \LaTeX pueda entender. Sin embargo, estas librerías presentan algunas limitaciones, como por ejemplo la imposibilidad de transformar matrices. Por lo tanto, se ha tenido que desarrollar varios métodos propios para escribir matrices, vectores y otras operaciones más complejas.

A continuación, se enseña el código fuente de uno de los métodos, concretamente el de pasar una matriz a \LaTeX . Observamos que seguimos apoyándonos en la librería `latexifier`, puesto que la transformación de cada elemento se hace con ella.

```
1 def matrix2latex(matrix) -> str:
2     res = "\\begin{pmatrix}\n"
3     for row in matrix:
4         for elem in row:
5             res += latexify(elem) + ' & '
6         res = res[:-3] + '\\\\ \\n'
7     return res[:-3] + "\\n\\end{pmatrix}"
```

Código fuente 3: Función auxiliar que transforma una matriz de Python a formato \LaTeX

3.3. Pseudocódigo y algoritmos

En esta sección explicaremos el pseudocódigo de los procedimientos utilizados para obtener las soluciones de cada ejercicio. Para ello, se ha basado en la información recopilada en el *Marco Teórico - Solución Explícita 2.1* y se apoya en información expuesta en [11, 12, 13, 14, 15, 16]. Asimismo, mostraremos algunos de los fragmentos más relevantes del código fuente. Se deja en el Anexo C una selección más extensa de los códigos desarrollados.

3.3.1. Sistema Fundamental de Soluciones

El Sistema Fundamental de Soluciones (SFS) es un conjunto de soluciones linealmente independientes de una ecuación o sistema diferencial. Estas soluciones forman la base del espacio de soluciones, lo que significa que cualquier solución de la ecuación puede ser escrita como una combinación lineal de los elementos del Sistema Fundamental de Soluciones.

Para ver cómo es el SFS, primero tenemos que encontrar las soluciones del sistema. En la Sección 2.1, vimos cómo se calculan las soluciones explícitas en función de los autovalores y autovectores asociados. Estas soluciones eran unas constantes por expresiones. Entonces, es fácil darse cuenta de que el SFS está

3.3. Pseudocódigo y algoritmos

formado por cada una de las expresiones que acompañan a las constantes c_1, c_2 o c . Se recogen estos resultados en la Tabla 3.1.

Tipo autovalor	Solución	SFS
$\lambda_1, \lambda_2 \in \mathbb{R}$ y $\lambda_1 \neq \lambda_2$	$X(t) = c_1 e^{\lambda_1 t} v_1 + c_2 e^{\lambda_2 t} v_2$	$\{e^{\lambda_1 t} v_1, e^{\lambda_2 t} v_2\}$
$\lambda_1, \lambda_2 \in \mathbb{R}$, $\lambda_1 = \lambda_2$ y matriz diagonalizable	$X(t) = ce^{\lambda_1 t}$	$\{e^{\lambda_1 t}\}$
$\lambda_1, \lambda_2 \in \mathbb{R}$, $\lambda_1 = \lambda_2$ y matriz no diagonalizable	$X(t) = e^{\lambda_1 t}(c_1 v_1 + c_2(v_1 t + v_2))$	$\{e^{\lambda_1 t} v_1, e^{\lambda_1 t}(v_1 t + v_2)\}$
$\lambda_1, \lambda_2 \in \mathbb{C}$	$X(t) = c_1 e^{pt}(\cos(qt)u - \operatorname{sen}(qt)w) \\ + c_2 e^{pt}(\operatorname{sen}(qt)u + \cos(qt)w)$	$\{e^{pt}(\cos(qt)u - \operatorname{sen}(qt)w), \\ e^{pt}(\operatorname{sen}(qt)u + \cos(qt)w)\}$

Tabla 3.1: Sistema Fundamental de Soluciones en función de los autovalores

Así, los pasos del algoritmo para obtener el Sistema Fundamental de Soluciones dado un sistema queda como en el Algoritmo 1.

Algoritmo 1 Sistema Fundamental de Soluciones

Entrada: $a, b, c, d \leftarrow$ coeficientes matriz

```

 $aVal \leftarrow$  lista de autovalores
 $aVal_1 \leftarrow$  primer autovalor
 $t \leftarrow$  símbolo Sympy
if  $aVal_1$  es real then {Si el primer autovalor es real, el segundo también}
    if  $m.a(aVal_1) = 1$  then {m.a = multiplicidad algebraica, autovalores reales y simples}
         $aVal_2 \leftarrow$  segundo autovalor
         $aVect_1 \leftarrow$  primer autovector
         $aVect_2 \leftarrow$  segundo autovector
         $sfs \leftarrow \exp(aVal_1 * t) * aVect_1, \exp(aVal_2 * t) * aVect_2$ 
    else {autovalor real doble}
        if  $m.a(aVal_1) = m.g(aVal_1)$  then {m.g = multiplicidad geométrica}
             $sfs \leftarrow \exp(aVal_1 * t) * aVect_1$ 
        else {Matriz no diagonalizable}
             $sfs \leftarrow \exp(aVal_1 * t) * aVect_1, \exp(aVal_1 * t) * aVect_1 * (1 + t)$ 
        end if
    end if
else {Autovalores complejos}
     $p \leftarrow \operatorname{Re}(aVal_1)$ 
     $l \leftarrow \operatorname{Im}(aVal_1)$ 
     $aVect \leftarrow$  autovector asociado a  $aVal$ 
     $u \leftarrow \operatorname{Re}(aVect)$ 
     $w \leftarrow \operatorname{Im}(aVect)$ 
     $sfs \leftarrow \exp(pt) * \cos(qt)u - \sin(qt)w, \exp(pt) \sin(qt)u + \cos(qt)w$ 
end if

```

Salida: sfs

Código

3.3.2. Soluciones Explícitas

Para obtener las soluciones explícitas de un sistema podemos reutilizar los cálculos para hallar el Sistema Fundamental de Soluciones. De esta manera, el procedimiento se reduce a obtener el SFS y multiplicar cada elemento por constantes. Entonces, el pseudocódigo para obtener las soluciones explícitas es el que se muestra a continuación.

Algoritmo 2 Soluciones explícitas

Entrada: $a, b, c, d \leftarrow$ coeficientes matriz

$sistfs \leftarrow sfs(a, b, c, d)$

$sols \leftarrow (c_1, c_2)^t * sistfs$

$\{(c_1, c_2)^t$ es la traspuesta de $(c_1, c_2)\}$

Salida: $sols$

3.3.3. Ecuación de segundo orden a sistema

Dada una ecuación de segundo grado homogénea, la transformaremos en un sistema mediante los pasos indicados en la Sección 2.5 y posteriormente, la resolveremos recurriendo a la llamada de obtener la solución explícita.

Algoritmo 3 Ecuaciones de segundo grado

Entrada: $a, b, c \leftarrow$ coeficientes de $ax'' + bx' + cx = 0$

$coef_1 \leftarrow 0$

$coef_2 \leftarrow 1$

$coef_3 \leftarrow -c/a$

$coef_4 \leftarrow -b/a$

$sol \leftarrow \text{sol-explicita}(coef_1, coef_2, coef_3, coef_4)$

Salida: sol

3.3.4. Clasificación puntos de equilibrio mediante los autovalores

Algoritmo 4 Clasificación puntos de equilibrio mediante los autovalores

Entrada: $a, b, c, d \leftarrow$ coeficientes matriz

$aVal_1 \leftarrow$ primer autovalor

if $aVal_1$ es real **then** {Si el primer autovalor es real, el segundo también}

if $m.a(aVal_1) = 1$ **then** {m.a = multiplicidad algebraica, autovalores reales y simples}

$aVal_2 \leftarrow$ segundo autovalor

$aVect_1, aVect_2 \leftarrow$ primer y segundo autovector

if $aVal_1 < 0 < aVal_2$ o $aVal_1 > 0 > aVal_2$ **then**

$tipo \leftarrow$ punto de silla

else if $aVal_1, aVal_2 < 0$ **then**

$tipo \leftarrow$ punto estable

3.3. Pseudocódigo y algoritmos

```

else
    tipo ← punto inestable
end if
else                                {autovalor real doble}
    if  $m.a(aVal_1) = m.g(aVal_1)$  then {matriz diagonalizable}
        tipo ← punto estelar
    else
        if  $aVal_1 > 0$  then
            tipo ← nodo impropio inestable
        else
            tipo ← nodo impropio estable
        end if
    end if
end if
else                                {Autovalores complejos}
    if  $\text{Re}(aVal_1) < 0$  then
        tipo ← foco estable
    else if  $\text{Re}(aVal_1) > 0$  then
        tipo ← foco inestable
    else
        tipo ← centro estable
    end if
end if
Salida: tipo

```

Cabe destacar que, aunque por los pseudocódigos parezca que el algoritmo es una única función, esta está dividida y organizada en funciones más atómicas. Además, en cada paso se añade una explicación utilizando `Paso.py` o `Pasos.py`, como se ha explicado anteriormente en la Sección 3.2. A continuación, mostramos un fragmento del código de `clasificar_punto_autoval`, omitiendo las comprobaciones y verificaciones iniciales.

```

1  def clasificar_punto_autoval(a, b, c, d):
2      # ... Comprobaciones ...
3      aVal = autovalores(A)
4      aVect = autovectores(A)
5      aVal1 = next(iter(aVal))  # primer autovalor
6      keys = list(aVal.keys())
7      if esReal(aVal1):  # real
8          pasos.append(getPaso(keys,
9                               "Hallamos los autovalores asociados al
10                             ↪ sistema"))
11      if aVal[aVal1] == 1:  # autovalor simple
12          pasos.append(c_p_a_real_distinto(aVect))
13      else:  # autovalor doble
14          if matrizDiagonalizable(A):  # autovalor doble diagonalizable
15              pasos.append(c_p_a_doble_diag(aVal1))

```

Código

```
15         else: # autovalor doble, no diagonalizable
16             pasos.append(c_p_a_doble_no_diag(aVal1))
17     else: # complejo
18         pasos = pasos + c_p_a_complejo(aVect)
19
20 return Pasos(pasos).toJson()
```

Código fuente 4: Función que clasifica el origen dependiendo de los autovalores. Asimismo, se muestra cómo sería una de las funciones auxiliares que se llaman desde este método. En concreto, se presenta la función `c_p_a_complejo`.

```
1 def c_p_a_complejo(aVect) -> list[Paso]:
2     aVal = aVect[0][0]
3     alpha = re(aVal)
4     signo = "menor que" if alpha < 0 else ("mayor que" if alpha > 0 else
5         "igual a")
6     tipo = "foco estable" if alpha < 0 else ("foco inestable" if alpha > 0
7         else "centro estable")
8     pasos = [getPaso(aVal, "Como los autovalores son complejos nos fijamos en
9         la parte real"),
10            getPaso(alpha, "Como la parte real es {} cero, el punto (0,0) es
11                un {}".format(signo, tipo))]
12
13 return pasos
```

Código fuente 5: Método auxiliar para determinar la estabilidad del punto origen cuando el autovalor es complejo

3.3.5. Clasificación puntos de equilibrio mediante la traza y el determinante

Veremos que este pseudocódigo tiene una estructura muy similar a la clasificación anterior. Esto era de esperar, ya que esta clasificación se puede considerar como una variante de la clasificación anterior, en el que no tenemos que hallar los autovalores y autovectores, que aumentan la complejidad del algoritmo. En este caso, utilizaremos solo la traza, el determinante y los valores de los parámetros b, c para determinar el tipo de estabilidad de los puntos de equilibrio.

Algoritmo 5 Clasificación del origen mediante la traza y el determinante

Entrada: $a, b, c, d \leftarrow$ coeficientes matriz

```
T, D, disc <- a + d, ad - bc, T2 - 4D
if disc < 0 then
    {complejos}
    if T < 0 then
        tipo <- foco estable
    else if T > 0 then
```

3.3. Pseudocódigo y algoritmos

```
    tipo ← foco inestable
else
    tipo ← centro estable
end if
else if disc > 0 then                                {reales y distintos}
    if D < 0 then
        tipo ← punto de silla
    else
        if T > 0 then
            tipo ← punto inestable
        else
            tipo ← punto estable
        end if
    end if
else
    if b = c = 0 then                                {reales e iguales}
        tipo ← punto estelar
    else if T > 0 then                                {diagonalizable}
        tipo ← nodo impropio inestable
    else
        tipo ← nodo impropio estable
    end if
end if
Salida: tipo
```

3.3.6. Representar diagramas de fases

Para representar diagramas de fases recurrimos a la librería Plotly perteneciente a Matplotlib. Además, permitimos que el usuario elija los límites de los ejes, así como la precisión de la representación.

```
1 import plotly.figure_factory as ff
2 # ... otras importaciones ...
3
4 def diagramaFase(a, b, c, d, delta, xlimInf, xlimSup, ylimInf, ylimSup, col):
5     comprobarCoeficientes(a, b, c, d)
6
7     xrange = arange(xlimInf, xlimSup + delta, delta)
8     yrange = arange(ylimInf, ylimSup + delta, delta)
9     X, Y = meshgrid(xrange, yrange)
10    U, V = a * X + b * Y, c * X + d * Y
11
12    fig = ff.create_quiver(X, Y, U, V, line=dict(width=0.75, color='#' + col))
13    return getResponseGraph(fig)
```

Código fuente 6: Código fuente para la representación diagrama de fases

Código

3.3.7. Conjugaciones topológicas

Algoritmo 6 Determinación de si dos sistemas son conjugados topológicos

Entrada: $a_1, b_1, c_1, d_1 \leftarrow$ coeficientes matriz 1

Entrada: $a_2, b_2, c_2, d_2 \leftarrow$ coeficientes matriz 2

$notHyperbolic_1, notHyperbolic_2 \leftarrow$ determinar

if alguno de los sistemas no es hiperbólico **then**

$conjugados \leftarrow$ False

else

$negativos_1, negativos_2 \leftarrow$ Número de autovalores negativos del sistema 1 y 2

if $negativos_1 \neq negativos_2$ **then**

$conjugados \leftarrow$ False

else

$conjugados \leftarrow$ True

end if

end if

Salida: $conjugados$

3.4. Librerías externas

Además de las bibliotecas mencionadas anteriormente, flask, pytexit y latexifier para la conexión y matplotlib para la representación, se han utilizado dos bibliotecas numéricas adicionales: sympy y numpy. El objetivo final del proyecto es crear paquetes retroalimentativos que no dependan de bibliotecas matemáticas externas. La creación de paquetes matemáticos internos no solo reducirá la dependencia de bibliotecas externas, sino que también aumenta la seguridad de la herramienta, minimizando el riesgo de posibles vulnerabilidades de seguridad en estas bibliotecas y mejorando su robustez y fiabilidad.

No obstante, dado que este es el primer año de desarrollo de la herramienta, no es factible tener las librerías numéricas internas ya implementadas. Por lo tanto, se ha creado un archivo denominado externalFunctions.py en la carpeta  auxiliar, que contiene todas las funciones de sympy y numpy que se utilizan en el código. Esto significa que en el futuro, cuando se implementen estas funciones, tales como la definición de coseno o el cálculo de autovalores, solo será necesario modificar este archivo.

Capítulo 4

Resultados y conclusiones

4.1. Resultados

En esta sección se mostrarán algunas capturas de pantalla del resultado de la aplicación en acción, así como demostraciones de algunos de los cálculos y capturas de error que la herramienta es capaz de realizar.

4.1.1. Listado de ejercicios

El contenido de “Sistemas de Ecuaciones Diferenciales” se encuentra implementado en su totalidad y ofrece una variedad de ejercicios. En particular, se pueden encontrar ejercicios que requieren la resolución de sistemas lineales como los mencionados a lo largo del trabajo, incluyendo la búsqueda de soluciones explícitas y del Sistema Fundamental de Soluciones. También se incluyen ejercicios para determinar la estabilidad de un sistema mediante diferentes métodos, como el uso de autovalores y autovectores, y la visualización gráfica de los diagramas de fases. Además, se resuelven ecuaciones de segundo grado y se determinan las conjugaciones topológicas de sistemas.



Figura 4.1: Contenido “Sistemas de Ecuaciones Diferenciales”

Cabe mencionar que en la sección de “Ecuaciones Diferenciales de Orden Superior” también se encuentra el ejercicio de “Resolución de ecuaciones de 2º

4.1. Resultados

grado”, que es el mismo ejercicio que aparece en la sección de sistemas, pero está categorizado en ambas secciones para facilitar su búsqueda.

4.1.2. Sistemas de ecuaciones diferenciales

En esta sección se presenta el proceso de resolución de un sistema de ecuaciones diferenciales utilizando la herramienta. En este caso, se ha seleccionado el siguiente sistema para ilustrar los cálculos de este ejercicio:

$$\begin{cases} x' = 2x, \\ y' = 3x + 2y. \end{cases}$$

Para resolver este sistema, se introducen los coeficientes en la herramienta y se selecciona la opción de resolver. A continuación, se muestran las capturas de pantalla del proceso completo. En la Figura 4.2 vemos la descripción del ejercicio y cómo se introducen los coeficientes del sistema.

Resolución de sistemas de ecuaciones diferenciales

Dado un sistema $X' = AX$

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

con $a, b, c, d \in \mathbb{R}$, y

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0,$$

se indica el Sistema Fundamental de Soluciones (SFS) y la solución explícita general.

Algoritmo utilizado

□² x[□] √□ □/√□ □/□ ln π sen cos tg

Introduzca los coeficientes del sistema.

$$\begin{cases} x' = 2x + 0y \\ y' = 3x + 2y \end{cases}$$

Obtener SFS Obtener soluciones explícitas

Figura 4.2: Enunciado $x' = 2x, y' = 3x + 2y$

En la Figura 4.3 se muestra la solución explícita del sistema diferencial. Esta se expresa tanto en formato Python como en estilo L^AT_EX y se permite copiar el resultado haciendo click en los iconos de portapapeles ubicados a la derecha.

Resultados y conclusiones

Solución

Descripción:
La solución explícita es la combinación lineal de los elementos en el Sistema Fundamental de Soluciones

Forma lineal:

$$X(t) = c_1 * \text{Matrix}([[0], [3.0*\exp(2.0*t)]]) + c_2 * \text{Matrix}([[1.00000000000000], [3.0*t*\exp(2.0*t)]])$$

En formato LaTeX:

$$X(t) = c_1 \begin{pmatrix} 0 \\ 3e^{2.0t} \end{pmatrix} + c_2 \begin{pmatrix} 1 \\ 3te^{2.0t} \end{pmatrix}$$

[Mostrar Pasos](#) [Descargar TeX](#)

Figura 4.3: Solución $x' = 2x, y' = 3x + 2y$

Pulsando en la opción de “Mostrar Pasos”, se despliega una lista de los procedimientos llevados a cabo para obtener la solución explícita del sistema.

Pasos

Paso 1: La matriz de coeficientes asociado al sistema es

$$A = \begin{pmatrix} 2 & 0 \\ 3 & 2 \end{pmatrix}$$

Paso 2: Hallamos los autovalores asociados al sistema para obtener el Sistema Fundamental de Soluciones

$$\lambda = 2 \quad (\text{doble})$$

Paso 3: Como el autovalor es doble y la matriz no es diagonalizable, el Sistema Fundamental de Soluciones es:

$$\left\{ \begin{pmatrix} 0 \\ 3e^{2.0t} \end{pmatrix}, \begin{pmatrix} 1 \\ 3te^{2.0t} \end{pmatrix} \right\}$$

Paso 4: La solución explícita es la combinación lineal de los elementos en el Sistema Fundamental de Soluciones

$$X(t) = c_1 \begin{pmatrix} 0 \\ 3e^{2.0t} \end{pmatrix} + c_2 \begin{pmatrix} 1 \\ 3te^{2.0t} \end{pmatrix}$$

Figura 4.4: Pasos $x' = 2x, y' = 3x + 2y$

Además, cabe destacar que hay controles de error en todos los ejercicios. Mostraremos algunos de los que detecta este ejercicio como ejemplo. En la Figura 4.5, introducimos una matriz nula, por lo que el determinante es 0. Recordamos

4.1. Resultados

que solo tratamos con sistemas cuyo determinante es distinto del nulo (entre otras condiciones), por lo que no es posible obtener la solución del sistema.

Introduzca los coeficientes del sistema.

$$\begin{cases} x' = \boxed{0} & x + \boxed{0} & y \\ y' = \boxed{0} & x + \boxed{0} & y \end{cases}$$

ⓘ Error: el determinante de la matriz de coeficientes no puede ser nula.

Figura 4.5: Determinante nulo

En la siguiente Figura 4.6, nos encontramos ante el caso en el que un usuario introduce incorrectamente algún coeficiente. En este caso se ha introducido un carácter en lugar de un número.

Introduzca los coeficientes del sistema.

$$\begin{cases} x' = \boxed{r} & x + \boxed{0} & y \\ y' = \boxed{0} & x + \boxed{1} & y \end{cases}$$

ⓘ Error: por favor, introduzca números reales

Figura 4.6: Coeficiente no válido

4.1.3. Estabilidad

En esta sección se estudia la estabilidad del punto origen para cualquier sistema homogéneo de dos ecuaciones de primer orden y con determinante distinto de cero. La aplicación permite visualizar el diagrama de fases y clasificarlo mediante dos métodos: estudiando los autovalores o fijándose en los valores de la traza y el determinante.

A continuación, se muestra unas imágenes del sistema que se va a resolver,

$$\begin{cases} x' = x + 3y, \\ y' = x - y, \end{cases}$$

y mostraremos el diagrama de fases. Para la representación se pueden elegir los límites inferiores y superiores de los ejes, así como la precisión y el color de la gráfica. Se han escogido los valores por defecto: límites inferiores en -2, superiores en 2, color `dodgeblue` (#005A9C) y precisión 0.25.

Resultados y conclusiones

Estudio estabilidad

Dado un sistema

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

con $a, b, c, d \in \mathbb{R}$, y

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0,$$

se estudia la estabilidad del único punto de equilibrio $(0, 0)$.
Además, se puede visualizar el diagrama de fases y personalizar los límites de los ejes, el color y la precisión de la representación.

Algoritmo utilizado

Elija el tipo de ejercicio que quiera resolver:

Estabilidad **Diagrama de fases**

Introduzca los coeficientes del sistema.

$$\begin{cases} x' = 1x + 3y \\ y' = 1x - 1y \end{cases}$$

Introduzca los intervalos de representación, la precisión y el color:

Valor mínimo del eje X: -2 Valor máximo del eje X: 2
Valor mínimo del eje Y: -2 Valor máximo del eje Y: 2
Precisión de la representación: 0.25 Color del dibujo:

Dibujar

Figura 4.7: Introducción coeficientes

Solución

La representación gráfica de la función es la siguiente:

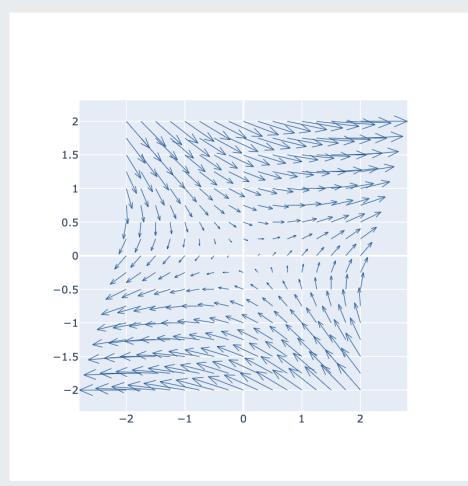


Figura 4.8: Diagrama de fases

4.1. Resultados

Observando las flechas, observamos que tienden al infinito en una dirección y en otra hacia el centro. Entonces podemos concluir con que es el punto $(0, 0)$ es un punto de silla.

Aun así, se puede conseguir explicitamente el tipo de estabilidad del origen, bien mediante la clasificación de autovalores o la de traza y determinante. En las siguientes figuras mostramos los resultados.

Pasos

Paso 1: Esta es la matriz de coeficientes asociado al sistema

$$A = \begin{pmatrix} 1 & 3 \\ 1 & -1 \end{pmatrix}$$

Paso 2: Como el determinante es $-4 \neq 0$, el único punto de equilibrio es el origen $(0, 0)$

$$\det(A) = -4$$

Paso 3: Hallamos los autovalores asociados al sistema

$$\lambda_1 = 2, \quad \lambda_2 = -2$$

Paso 4: Como los dos autovalores son reales y de diferente signo, el punto $(0, 0)$ es un punto de silla.

Figura 4.9: Clasificación por autovalor

Pasos

Paso 1: Recordemos que siendo T la traza y D el determinante, los autovalores se pueden expresar como:

$$\lambda = \frac{T \pm \sqrt{T^2 - 4D}}{2}$$

Paso 2: Los valores de la traza y el determinante son:

$$T = 0, \quad D = -4$$

Paso 3: El discriminante es $\Delta = 16 > 0$. Por lo que los autovalores son reales y diferentes

$$\Delta = T^2 - 4D = 16$$

Paso 4: Como el determinante $D = -4$ es menor que 0, los autovalores son de signo contrario, el punto $(0, 0)$ es un punto de silla.

Figura 4.10: Clasificación por traza y determinante

4.1.4. Conjugación topológica

Finalmente, el último tipo de ejercicios que resuelve la aplicación sobre la asignatura de Ecuaciones Diferenciales es determinar si dos sistemas son conjugados topológicamente.

Resultados y conclusiones

Mostraremos un ejemplo utilizando los sistemas

$$\begin{cases} x' = x \\ y' = 2y \end{cases} \quad \text{y} \quad \begin{cases} x' = -x \\ y' = -y \end{cases}$$

que fácilmente se ve que no son conjugados topológicamente, ya que los autovalores son los dos positivos para el primer sistema, y ambos negativos para el segundo.

Conjugación topológica

Dados dos sistemas

$$\begin{cases} x' = a_1x + b_1y, \\ y' = c_1x + d_1y, \end{cases} \quad \text{y} \quad \begin{cases} x' = a_2x + b_2y, \\ y' = c_2x + d_2y, \end{cases}$$

con todos los coeficientes números reales y los determinantes distintos del 0, se determina si los sistemas son conjugados topológicos.

Algoritmo utilizado

▼

\square^2
 x^2
 $\sqrt{\square}$
 $\square\sqrt{\square}$
 \ln
 π
sen
cos
tg

Introduzca los coeficientes de los sistemas.

$$\begin{cases} x' = \boxed{1}x + \boxed{0}y \\ y' = \boxed{0}x + \boxed{2}y \end{cases}$$

$$\begin{cases} x' = \boxed{-1}x + \boxed{0}y \\ y' = \boxed{0}x + \boxed{-1}y \end{cases}$$

Obtener conjugación

Figura 4.11: Enunciado

Pasos

Paso 1: Dos sistemas son conjugados topológicamente si las matrices son hiperbólicas y tiene mismo número de autovalores negativos. Las matrices asociadas a los sistemas son:

$$A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

Paso 2: Obtenemos los autovalores de ambas matrices:

Autovalores de la primera matriz : $\lambda_1^1 = 1, \quad \lambda_2^1 = 2$
 Autovalores de la segunda matriz : $\lambda_1^2 = -1, \quad \lambda_2^2 = -1$

Paso 3: Las dos matrices tienen diferente número de autovalores negativos. Mientras que el primer sistema tiene 0 autovalores negativos, el segundo tiene 2, luego los dos sistemas no son conjugados topológicos.

Figura 4.12: Solución

4.1.5. Ecuaciones de segundo orden

En este apartado se muestra el funcionamiento de la herramienta con ecuaciones diferenciales de segundo orden. En particular, se muestra cómo resolver la ecuación $2x'' + x' - x = 0$.

Asimismo, se presentan capturas de pantalla que muestran todo el proceso de resolución, incluyendo la introducción de la ecuación de segundo orden en la herramienta y la selección de la opción "Mostrar Pasos", la cual permite ver una lista detallada de los procedimientos utilizados para resolver la ecuación.

Resolución de ecuaciones diferenciales de 2º grado

Dada una ecuación

$$ax'' + bx' + cx = 0 \quad \text{con } a, b, c \in \mathbb{R},$$

se transforma la ecuación en un sistema de 2 ecuaciones de primer orden $X' = AX$.
 Además, se permite resolver la ecuación si la matriz de coeficientes del sistema resultante tiene determinante distinto de 0.

Algoritmo utilizado

▼

x^2
 x^3
 \sqrt{x}
 $\sqrt[3]{x}$
 $\ln x$
 π
sen
cos
tg

Introduzca los coeficientes de la ecuación de segundo grado.

$x'' +$ $x' +$ $x = 0$

Resolver
Transformar a sistema

Solución

Descripción:

Sabiendo que $y = 0.447214c_1e^{0.5t} + 0.745356c_2e^{-1.0t}$ y deshaciendo el cambio de variable $x' = y$,

Forma lineal:

$x=0.894427190999916*c1*exp(0.5*t) - 0.74535599249993*c2*exp(-1.0*t)$

En formato LaTeX:

$x = 0.894427c_1e^{0.5t} - 0.745356c_2e^{-1.0t}$

Mostrar Pasos
Descargar TeX

Figura 4.13: Solución de $2x'' + x' - x = 0$

Los pasos detallados de la resolución de la ecuación de segundo grado se muestran en la Figura 4.14.

Resultados y conclusiones

Pasos

Paso 1: Introducimos el cambio de variable

$$x' = y$$

Paso 2: Entonces, $x'' = y'$, y despejando x''

$$x'' = -\frac{c}{a}x - \frac{b}{a}x'$$

Paso 3: El sistema equivalente a la ecuación de segundo grado es

$$\begin{cases} x' = y \\ y' = \frac{x}{2} - \frac{y}{2} \end{cases}$$

Paso 4: La matriz de coeficientes asociado al sistema es

$$A = \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$$

Paso 5: Hallamos los autovalores asociados al sistema para obtener el Sistema Fundamental de Soluciones

$$\lambda_1 = \frac{1}{2}, \quad \lambda_2 = -1$$

Paso 6: Como los autovalores son reales y distintos, el Sistema Fundamental de Soluciones es:

$$\left\{ \begin{pmatrix} 0.894427e^{0.5t} \\ 0.447214e^{0.5t} \end{pmatrix}, \begin{pmatrix} -0.745356e^{-1.0t} \\ 0.745356e^{-1.0t} \end{pmatrix} \right\}$$

Paso 7: La solución explícita es la combinación lineal de los elementos en el Sistema Fundamental de Soluciones

$$X(t) = c_1 \begin{pmatrix} 0.894427e^{0.5t} \\ 0.447214e^{0.5t} \end{pmatrix} + c_2 \begin{pmatrix} -0.745356e^{-1.0t} \\ 0.745356e^{-1.0t} \end{pmatrix}$$

Paso 8: Sabiendo que $y = 0.447214c_1e^{0.5t} + 0.745356c_2e^{-1.0t}$ y deshaciendo el cambio de variable $x' = y$,

$$x = 0.894427c_1e^{0.5t} - 0.745356c_2e^{-1.0t}$$

Figura 4.14: Pasos resolución $2x'' + x' - x = 0$

4.2. Propuestas de uso y extensión

La aplicación software que hemos desarrollado está dirigida a los estudiantes de la ETSIINF que deseen estudiar o repasar conceptos de matemáticas impartidos en las clases. Esta herramienta les permitirá verificar la solución correcta de un ejercicio y visualizar cómo se resuelve un tipo de pregunta específica.

Además, nuestra aplicación puede ser utilizada como una herramienta complementaria en el aula. Por ejemplo, en la asignatura opcional de cuarto curso “Ecuaciones en Derivadas Parciales y Simulación Numérica”, a menudo se requiere resolver sistemas de ecuaciones diferenciales que verifican las condicio-

4.2. Propuestas de uso y extensión

nes de los tratados en este trabajo. Nuestra herramienta habría sido de gran ayuda al permitir a los estudiantes obtener rápidamente las soluciones a estos sistemas, ahorrando tiempo valioso en el aula y aumentando la eficiencia del aprendizaje.

En cuanto a las propuestas de extensión, durante el desarrollo de este trabajo, me he dado cuenta de la necesidad de una librería completa y exhaustiva que permita la transformación de fórmulas de Python a \LaTeX . Aunque existen paquetes como `latexifier` y `pytexit` que realizan esta tarea, tienen limitaciones importantes, como la incapacidad de procesar matrices con variables o con constantes en formato \LaTeX . Para superar esta limitación, he creado varios métodos que permiten realizar estas transformaciones. Para el futuro, se podría plantear el desarrollo de un compilador general de fórmulas en Python que convierta automáticamente el código en un formato compatible con \LaTeX , como una extensión de `latexifier` o `pytexit`.

Además, hay una oportunidad de ampliar el contenido actual de la aplicación mediante la inclusión de temas como la resolución de ejercicios de primer orden, ecuaciones lineales de órdenes superiores a 2 o algoritmos avanzados como el Método de Variación de Parámetros o Riccati. En definitiva, es importante cubrir todos los contenidos de la asignatura para desarrollar enteramente el proyecto.

Otra propuesta es la de incorporar la capacidad de graficar órbitas superpuestas a los diagramas de fases. Actualmente, las limitaciones de `plotly` y `meshgrid` han impedido la inclusión de esta función en el presente Trabajo Fin de Grado, pero se puede investigar otras opciones y métodos para resolver esta limitación en futuras actualizaciones.

En lo que respecta al diseño de la interfaz de la aplicación, se sugiere que los próximos estudiantes de Prácticum reconsideren algunos aspectos. Por ejemplo, resulta un tanto extraño que, al reducir el tamaño de la ventana, los contenidos se alineen a la derecha.

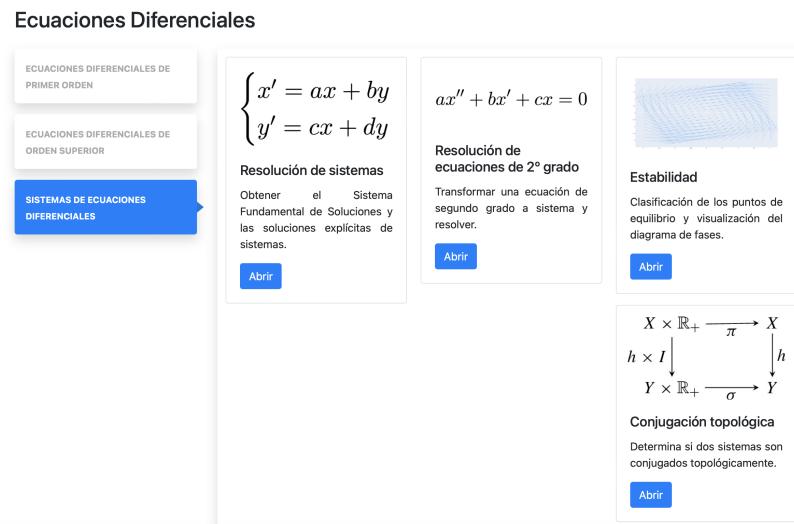


Figura 4.15: Contenidos centrados a la derecha

Resultados y conclusiones

Por otra parte, se considera innecesario que se permita la introducción de funciones cuando solo se esperan números reales. En consecuencia, se propone eliminar esta opción.

Resolución de ecuaciones diferenciales de 2º grado

Dada una ecuación

$$ax'' + bx' + cx = 0 \quad \text{con } a, b, c \in \mathbb{R},$$

se transforma la ecuación en un sistema de 2 ecuaciones de primer orden $X' = AX$.

Además, se permite resolver la ecuación si la matriz de coeficientes del sistema resultante tiene determinante distinto de 0.

Algoritmo utilizado

Introduzca los coeficientes de la ecuación de segundo grado.

0 $x'' +$ 0 $x' +$ 0 $x = 0$

Resolver Transformar a sistema

Figura 4.16: Contenidos centrados a la derecha

En conclusión, aunque he mantenido una comunicación constante con los responsables de la visualización de la página durante la realización de mi Trabajo Fin de Grado, no tuvieron tiempo para realizar más cambios. Por esta razón, se recomienda que los estudiantes futuros vuelvan a examinar estos aspectos.

Por último, sería deseable unificar el formato de todos los códigos utilizados en todas las asignaturas y en todos los repositorios del proyecto. Esto permitiría una mejor visión del proyecto en conjunto. Para lograr esto, es necesario establecer pautas de formato comunes y seguir una misma estructura de código en todas las asignaturas.

4.3. Conclusión

El desarrollo del *backend* que permite mostrar los pasos para la resolución de ejercicios sobre los contenidos teóricos tratados en este trabajo representa un logro importante en la integración de las Matemáticas y la Informática. Esta herramienta ofrece una solución práctica y útil para los estudiantes del Grado en Matemáticas e Informática, ya que les permitirá mejorar su comprensión de los conceptos y la metodología para la resolución de problemas de Ecuaciones Diferenciales.

Además, este proyecto ha sido una oportunidad para unir los conocimientos de Matemáticas e Informática en un proyecto interdisciplinar. Durante el desarrollo del trabajo, he podido comprender mejor el funcionamiento de una página web y la conexión entre el *backend* y el *frontend*, lo cual ha sido muy enriquecedor

4.3. Conclusión

para mi formación profesional y académica.

En cuanto a la coordinación entre el desarrollo del *frontend* y el *backend*, ha sido un desafío mantener una sincronización fluida debido a la ejecución en paralelo de ambos procesos. Frecuentemente, he tenido que realizar modificaciones en los métodos ya completados debido a cambios de último momento en ciertos aspectos del proyecto.

Este trabajo servirá como guía para futuros alumnos interesados en participar en proyectos de este tipo, tanto en el marco de un Practicum como en un TFG. También puede ser de gran utilidad para aquellos usuarios que tengan curiosidad y que deseen comprender el funcionamiento de la aplicación y la base teórica detrás de ella.

Por último, cabe destacar que este proyecto también queda como documentación del código utilizado en el *backend* de la aplicación. Esto permitirá que los desarrolladores futuros puedan revisar el código y utilizarlo como base para nuevas versiones de la aplicación o para proyectos similares.

Capítulo 5

Análisis de impacto

Los Objetivos de Desarrollo Sostenible (ODS) son unos objetivos de desarrollo establecidos por las Naciones Unidas para abordar de manera integral los problemas de desarrollo en tres dimensiones: social, económica y ambiental, desde 2015 hasta 2030, con el objetivo de avanzar hacia un camino de desarrollo sostenible (véase [17, 18]).

Para lograr el desarrollo sostenible, es necesario la coordinación de tres aspectos fundamentales: el crecimiento económico, la inclusión social y la protección del medio ambiente. Para ello, se han establecido exactamente 17 metas y 169 indicadores en áreas como el crecimiento económico.

En este capítulo se llevará a cabo un análisis del impacto potencial de los resultados obtenidos durante la realización del Trabajo Fin de Grado.

La pandemia COVID-19 ha transformado significativamente el sector educativo y se ha demostrado la importancia de la educación en línea. Aunque la Organización Mundial de la Salud ha decretado el fin de la emergencia internacional por la pandemia, el impacto de la pandemia ha sido permanente.



En este contexto, el **ODS 4: “Educación de calidad”**, ha adquirido una importancia aún mayor en la promoción de oportunidades de aprendizaje para todos, garantizando una educación inclusiva y equitativa de calidad. En este Trabajo Fin de Grado, se ha desarrollado un software matemático que pretende facilitar el aprendizaje de la disciplina de Ecuaciones Diferenciales. La creación de esta herramienta innovadora y efectiva puede mejorar la calidad de la educación, ya que ofrece a los estudiantes una alternativa para profundizar en los contenidos y mejorar su comprensión.

Además, el desarrollo de la herramienta de aprendizaje en línea también contribuye al logro del ODS 4, al promover la educación en línea y garantizar oportunidades de aprendizaje permanente para todos. La disponibilidad del software en línea, en cualquier momento y desde cualquier lugar, hace que sea accesible

para todos los estudiantes sin importar la ubicación geográfica o la situación personal.

Por otro lado, este trabajo también contribuye al logro del **objetivo número 8: “Trabajo decente y crecimiento económico”**. La realización de este proyecto de forma colaborativa con otros estudiantes nos ha permitido desarrollar nuevas habilidades útiles para nuestra futura carrera laboral, como la gestión de versiones o la programación de la conexión del Backend y el Frontend. En este sentido, hemos podido aplicar en práctica los conocimientos adquiridos en nuestra formación académica en el ámbito de las matemáticas y la informática.

Asimismo, el proyecto también al cumplimiento del **ODS 9: “Industria, innovación e infraestructura”**. La elaboración del software matemático se enmarca dentro de un contexto de innovación tecnológica, en el que hemos empleado las últimas tecnologías como Python o Angular. Además, este proyecto puede considerarse una iniciativa de investigación y desarrollo (I+D). Tal como lo señala la ONU (véase [17]), las inversiones en proyectos en I+D están creciendo, pero es necesario acelerar su implementación para impulsar el crecimiento económico y la innovación a nivel mundial.

Finalmente, los tres objetivos mencionados anteriormente también contribuyen en la obtención del **objetivo 17: “Alianzas para lograr los objetivos”**. Este último ODS tiene como objetivo fortalecer la Alianza Mundial para el Desarrollo Sostenible, mediante el intercambio de conocimientos, tecnología, capacidad técnica y recursos financieros. En este caso, la búsqueda de los objetivos ODS 4, 8 y 9 mediante en este trabajo en particular, demuestra nuestro compromiso con la colaboración y la cooperación en la consecución de la Agenda 2030. A través de alianzas entre estudiantes y profesores, estamos trabajando juntos para impulsar el desarrollo sostenible y alcanzar los objetivos establecidos.



Bibliografía

- [1] M. W. Hirsch, S. Smale, and R. L. Devaney, *Differential equations, dynamical systems, and an introduction to chaos.* Academic press, 2012.
- [2] J. Liesen and V. Mehrmann, *Linear algebra.* Springer, 2015.
- [3] A. Kisieliov, M. L. Krasnov, G. Makarenko, and E. A. Bernardo, “Problemas de ecuaciones diferenciales ordinarias,” Mir, Tech. Rep., 1968.
- [4] O. Plaat, *Ecuaciones diferenciales ordinarias.* Reverté, 2021.
- [5] J. Perdomo-Díaz, “Módulo de enseñanza para la introducción de las ecuaciones diferenciales ordinarias en un ambiente de resolución de problemas con tecnología,” *NÚMEROS. Revista de Didáctica de las Matemáticas*, vol. 78, pp. 113–134, 2011.
- [6] R. K. Miller and A. N. Michel, *Ordinary differential equations.* Academic Press, 2014.
- [7] E. A. Coddington, *An introduction to ordinary differential equations.* Courier Corporation, 2012.
- [8] L. Cesari, *Asymptotic behavior and stability problems in ordinary differential equations.* Springer Science & Business Media, 2012, vol. 16.
- [9] P. Lokhande, F. Aslam, N. Hawa, J. Munir, and M. Gulamgaus, “Efficient way of web development using python and flask,” 2015.
- [10] M. Grinberg, *Flask web development: developing web applications with python.* O'Reilly Media, Inc., 2018.
- [11] K. J. Millman and M. Aivazis, “Python for scientists and engineers,” *Computing in science & engineering*, vol. 13, no. 2, pp. 9–12, 2011.
- [12] S. Tosi, *Matplotlib for Python developers.* Packt Publishing Ltd, 2009.
- [13] N. Ari and M. Ustazhanov, “Matplotlib in python,” in *2014 11th International Conference on Electronics, Computer and Computation (ICECCO).* IEEE, 2014, pp. 1–6.
- [14] J. C. Robinson, *An introduction to ordinary differential equations.* Cambridge University Press, 2004.
- [15] G. Ortigoza Capetillo, “Resolviendo ecuaciones diferenciales ordinarias con

BIBLIOGRAFÍA

- maple y mathematica," *Revista mexicana de física E*, vol. 53, no. 2, pp. 155–167, 2007.
- [16] E. Bressert, "Scipy and numpy: an overview for developers," 2012.
- [17] "The 17 goals. united nations." [Online]. Disponible: <https://sdgs.un.org/goals>
- [18] C. G. Gil, "Objetivos de desarrollo sostenible (ods): una revisión crítica," *Papeles de relaciones ecosociales y cambio global*, vol. 140, no. 1, pp. 107–118, 2018.

Anexo A

Demostraciones

En este primer anexo demostraremos los teoremas que se han dado por supuesto, así como alguna demostración que era muy extensa.

Proposición. *La aplicación matricial $e^{A \cdot} : \mathbb{R} \rightarrow \mathcal{M}_n(\mathbb{K})$ que a cada t le hace corresponder*

$$e^{At} = \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}$$

converge uniformemente.

Demostración. Sea $C \subset \mathbb{R}$ un intervalo compacto. Sea $f_k : C \rightarrow \mathcal{M}_n(\mathbb{K})$ definida por

$$f_k(t) = \frac{A^k t^k}{k!}.$$

Sea la norma euclídea $\|\cdot\|$, recordamos que es submultiplicativa, queriendo decir que $\|AB\| \leq \|A\| \|B\|$. Entonces,

$$\|f_k(t)\| = \left\| \frac{A^k t^k}{k!} \right\| \leq \frac{\|A\|^k b^k}{k!} := M_k \quad \text{para todo } k \in \mathbb{Z}_+, t \in C,$$

donde $b = \max_{t \in C} |t|$. Como la serie numérica $\sum_{k=1}^{\infty} M_k = e^{\|A\|b} - 1 < \infty$, por el criterio mayorante de Weierstrass, se concluye con que $I_n + \sum_{k=1}^{\infty} \frac{A^k t^k}{k!}$ converge uniformemente sobre C . \square

Teorema (Teorema de Jacobi). *Sea I un intervalo abierto. Si $x_1(t), \dots, x_n(t)$ son soluciones del sistema $X' = AX$. Entonces, se verifica*

$$W(t) = W(t_0) e^{\int_{t_0}^t \text{Tr}(A(s)) ds}$$

para cualquier $t_0 \in I$ y siendo $W[x_1, \dots, x_n](t)$ el wronskiano definido en la Definición 2.

Demostración. Demostraremos el teorema para $n = 2$, por lo que

$$A(t) = \begin{pmatrix} a(t) & b(t) \\ c(t) & d(t) \end{pmatrix}.$$

En este caso, dadas dos soluciones del sistema $X' = A(t)X$

$$X_1(t) = \begin{pmatrix} x_{11}(t) \\ x_{21}(t) \end{pmatrix} \quad \text{y} \quad X_2(t) = \begin{pmatrix} x_{12}(t) \\ x_{22}(t) \end{pmatrix},$$

se tiene que

$$\begin{pmatrix} x'_{11}(t) \\ x'_{21}(t) \end{pmatrix} = \begin{pmatrix} a(t)x_{11}(t) & b(t)x_{21}(t) \\ c(t)x_{11}(t) & d(t)x_{21}(t) \end{pmatrix} \quad \text{y} \quad \begin{pmatrix} x'_{12}(t) \\ x'_{22}(t) \end{pmatrix} = \begin{pmatrix} a(t)x_{12}(t) & b(t)x_{22}(t) \\ c(t)x_{12}(t) & d(t)x_{22}(t) \end{pmatrix}. \quad (\text{A.1})$$

Entonces,

$$\begin{aligned} \frac{d}{dt}W(t) &= \frac{d}{dt} \begin{vmatrix} x_{11}(t) & x_{12}(t) \\ x_{21}(t) & x_{22}(t) \end{vmatrix} \\ &= \frac{d}{dt}(x_{11}(t)x_{22}(t) - x_{12}(t)x_{21}(t)) \\ &= x'_{11}(t)x_{22}(t) + x_{11}(t)x'_{22}(t) - x'_{12}(t)x_{21}(t) - x_{12}(t)x'_{21}(t) \\ &= (a(t) + d(t)) \cdot (x_{11}(t)x_{22}(t) - x_{12}(t)x_{21}(t)) \\ &= \text{Tr}(A(t))W(t), \end{aligned}$$

donde hemos sustituido las ecuaciones de (A.1). Resolviendo entonces la ecuación $\frac{d}{dt}W(t) = \text{Tr}(A(t))W(t)$, obtenemos que

$$W(t) = Ce^{\int_{t_0}^t \text{Tr}(A(s))ds}.$$

Como $W(t_0) = C$, se concluye que $W(t) = W(t_0)e^{\int_{t_0}^t \text{Tr}(A(s))ds}$. \square

Proposición. Los autovectores asociados a autovalores distintos son linealmente independientes.

Demostración. Supongamos que tenemos dos autovalores distintos $\lambda_1 \neq \lambda_2$. Por definición de autovalor, se cumple que

$$Av_1 = \lambda_1 v_1 \quad \text{y que} \quad Av_2 = \lambda_2 v_2.$$

Entonces, para ver que son linealmente independientes busquemos una combinación lineal de los vectores que dé como resultado el vector nulo. Sean los coeficientes $c_1, c_2 \in \mathbb{R}$,

$$c_1 v_1 + c_2 v_2 = 0. \quad (\text{A.2})$$

Multiplicando por A en ambos miembros,

$$\begin{aligned} 0 &= A(c_1 v_1 + c_2 v_2) \\ &= c_1 Av_1 + c_2 Av_2 \\ &= c_1 \lambda_1 v_1 + c_2 \lambda_2 v_2. \end{aligned} \quad (\text{A.3})$$

Demostraciones

Por otro lado, multiplicando la ecuación (A.2) por λ_1 ,

$$c_1\lambda_1v_1 + c_2\lambda_1v_2. \quad (\text{A.4})$$

Restando las ecuaciones (A.2) y (A.4), llegamos a que

$$c_2\lambda_2v_2 - c_2\lambda_1v_2 = c_2v_2(\lambda_2 - \lambda_1) = 0.$$

Como hemos establecido que $\lambda_1 \neq \lambda_2$, entonces la diferencia no puede ser nula. Así, $c_2 = 0$ y entonces, $c_1 = 0$.

Luego como $c_1 = c_2 = 0$, queda demostrado que los autovectores $\{v_1, v_2\}$ son linealmente independientes. \square

Proposición. *Una matriz A de dimensión 2 con autovalor doble es diagonalizable si y solo si A es una matriz diagonal.*

Demostración. Sea A una matriz de orden 2, con el autovalor doble λ . Entonces buscamos si esta matriz se puede diagonalizar, de manera que se pueda expresar como $\lambda I = PAP^{-1}$.

Multiplicando P^{-1} a la izquierda y P a la derecha, se tiene que

$$A = P^{-1}(\lambda I)P = \lambda I.$$

Por tanto, se concluye que A es diagonalizable si y solo si $A = \lambda I$. Es decir, que A sea diagonal. \square

Teorema. *Sean dos matrices 2×2 hiperbólicas A_1 y A_2 . Los sistemas $X' = A_1X$ y $X' = A_2X$ son conjugados si y solo si cada matriz tiene el mismo número de autovalores con parte real negativa.*

Demostración. Supondremos que todos nuestros sistemas están en forma canónica. Entonces, dividiremos la demostración en tres casos.

Caso 1

Suponemos que tenemos dos sistemas lineales $X' = A_1X$ y $X' = A_2X$ tal que A_1 y A_2 tienen los autovalores reales $\lambda_1 < 0 < \mu_1$ y $\lambda_2 < 0 < \mu_2$ respectivamente, por lo que los dos sistemas tienen un punto de silla en el origen. Para las ecuaciones $x' = \lambda_1x$ y $x' = \lambda_2x$ existe el flujo conjugado por el siguiente homeomorfismo:

$$h_1(x) = \begin{cases} x^{\frac{\lambda_2}{\lambda_1}} & \text{si } x \geq 0, \\ -|x|^{\frac{\lambda_2}{\lambda_1}} & \text{si } x < 0. \end{cases}$$

Análogamente, definimos la función h_2 para las ecuaciones $y' = \mu_1y$ y $y' = \mu_2y$:

$$h_2(y) = \begin{cases} y^{\frac{\mu_2}{\mu_1}} & \text{si } y \geq 0, \\ -|y|^{\frac{\mu_2}{\mu_1}} & \text{si } y < 0. \end{cases}$$

Así, la función

$$H(x, y) = (h_1(x), h_2(y))$$

es una conjugación entre los dos sistemas.

Caso 2

Consideremos el sistema $X' = AX$ y que la matriz A está en forma canónica con los autovalores que tienen parte real negativa. Además, asumiremos que A no es de la forma

$$\begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}$$

con $\lambda < 0$. Así, A es de alguna de las dos siguientes formas:

$$(a) \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix} \quad \text{o} \quad (b) \begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix}$$

con $\alpha, \lambda, \mu < 0$. Demostraremos que para cualquiera de las formas de A , el sistema es conjugado a $X' = BX$, siendo

$$B = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix},$$

ya que de este modo, se concluye con que cualquier sistema de estas formas son conjugados.

Consideramos la circunferencia unidad S^1 en el plano, parametrizado por la curva $X(\theta) = (\cos \theta, \sin \theta)$, con $0 \leq \theta \leq 2\pi$. Vemos que los campos vectoriales apuntan hacia dentro. Para el caso (a), el campo vectorial en S^1 viene dado por

$$AX(\theta) = \begin{pmatrix} \alpha \cos \theta + \beta \sin \theta \\ \beta \cos \theta + \alpha \sin \theta \end{pmatrix}.$$

Y el vector normal que apunta hacia fuera de S^1 en $X(\theta)$ es de la forma

$$N(\theta) = \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix}.$$

Entonces, el producto interno de $AX(\theta)$ y $N(\theta)$ satisface que

$$AX(\theta) \cdot N(\theta) = \alpha(\cos^2 \theta + \sin^2 \theta) < 0,$$

puesto que $\alpha < 0$. Esto demuestra que efectivamente, $AX(\theta)$ apunta al interior de S^1 . El caso (b) se demuestra de manera análoga.

Luego cada solución no nula de $X' = AX$ corta con S^1 una única vez. Denotemos como ϕ_t^A al mapeo del tiempo t para este sistema, y sea $\tau = \tau(x, y)$ el tiempo en el que ϕ_t^A se encuentra con S^1 . Entonces, como S^1 es la circunferencia unidad,

$$|\phi_{\tau(x,y)}^A(x, y)| = 1. \tag{A.5}$$

Ahora, sea ϕ_t^B el mapeo del tiempo t para el sistema $X' = BX$. Claramente, se tiene que

$$\phi_t^B(x, y) = (e^{-t}x, e^{-t}y).$$

Demostraciones

Definamos una conjugación H entre estos dos sistemas. Si $(x, y) \neq (0, 0)$, escribiremos H como sigue:

$$H(x, y) = \phi_{-\tau(x,y)}^B \phi_{\tau(x,y)}^A(x, y)$$

y estableciendo que $H(0, 0) = (0, 0)$.

Veamos que H da una conjugación. Como $\phi_{\tau-s}^A \phi_s^A(x, y) = \phi_\tau^A(x, y) \in S^1$, se tiene que

$$\tau(\phi_s^A(x, y)) = \tau(x, y) - s.$$

Así, tenemos que

$$\begin{aligned} H(\phi_s^A(x, y)) &= \phi_{-\tau+s}^B \phi_{\tau-s}^A(\phi_s^A(x, y)) \\ &= \phi_s^B \phi_{-\tau}^B \phi_\tau^A(x, y) \\ &= \phi_s^B(H(x, y)). \end{aligned}$$

Luego H es una conjugación.

Queda ver que H es un homeomorfismo. Podemos construir una inversa G para H invirtiendo el orden de las composiciones de funciones, es decir:

$$G(x, y) = \phi_{-\tau_1(x,y)}^A \phi_{\tau_1(x,y)}^B(x, y)$$

siendo $\tau_1(x, y) = \log r$, con $r^2 = x^2 + y^2$ y estableciendo que $G(0, 0) = (0, 0)$. Aquí, $\tau_1(x, y)$ es el tiempo necesario para que la solución de $X' = BX$ que pasa por (x, y) interseque con S^1 .

Así, es claro que $G = H^{-1}$, por lo que H es una función inyectiva y sobreyectiva. Además, G es continua en $(x, y) \neq (0, 0)$ ya que G se puede expresar como composición de funciones continuas:

$$G(x, y) = \phi_{-\log r}^A \left(\frac{x}{r}, \frac{y}{r} \right).$$

Y para el punto $(0, 0)$, supongamos que (x, y) es un punto muy cercano al origen, de manera que el radio r es pequeño. Entonces, como $r \rightarrow 0$, se tiene que $-\log r \rightarrow \infty$. Luego $(\frac{x}{r}, \frac{y}{r})$ es un punto de S^1 y para un valor de r lo suficientemente pequeño, $\phi_{-\log r}^A$ lleva a la circunferencia unidad muy cerca del origen. Por tanto, G también es continua en $(0, 0)$.

Finalmente queda demostrar la continuidad de H . Para ello veremos que $\tau(x, y)$ es continua. Recordamos que τ venía determinada por la ecuación (A.5)

$$\left| \phi_{\tau(x,y)}^A(x, y) \right| = 1.$$

Escribimos $\phi_{\tau(x,y)}^A$ y hacemos la derivada parcial de $\phi_{\tau(x,y)}^A$ respecto de t :

$$\begin{aligned} \frac{\partial}{\partial t} \left| \phi_{\tau(x,y)}^A(x, y) \right| &= \frac{\partial}{\partial t} \sqrt{(x(t))^2 + (y(t))^2} \\ &= \frac{1}{\sqrt{(x(t))^2 + (y(t))^2}} (x(t)x'(t) + y(t)y'(t)) \\ &= \frac{1}{\left| \phi_{\tau(x,y)}^A(x, y) \right|} \left(\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \cdot \begin{pmatrix} x'(t) \\ y'(t) \end{pmatrix} \right). \end{aligned}$$

El producto final es distinto de cero cuando $t = \tau(x, y)$, ya que el campo vectorial está dado por puntos $(x'(t), y'(t))$ interiores de S^1 . Entonces,

$$\frac{\partial}{\partial t} \left| \phi_{\tau(x,y)}^A(x, y) \right| \neq 0$$

en $(\tau(x, y), x, y)$. Aplicando el Teorema de la Función Implícita, τ es diferenciable en (x, y) y por tanto, continua. La continuidad de H en el origen se sigue como en la demostración de $G = H^{-1}$.

Queda entonces demostrado que H es un homeomorfismo y por tanto, hemos encontrado una conjugación entre los sistemas $X' = AX$ y $X' = BX$. La demostración es análoga para autovalores con parte real positiva.

Caso 3

Finalmente, supongamos que

$$A = \begin{pmatrix} \lambda & 1 \\ 0 & 1 \end{pmatrix}$$

con $\lambda < 0$. El campo vectorial asociado no apunta hacia el centro en este caso. Pero si definimos

$$T = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix},$$

entonces el campo vectorial viene dado por

$$Y' = (T^{-1}AT)Y$$

sí que tiende al punto $(0, 0)$ siempre que $\epsilon > 0$ sea lo suficientemente pequeño. En efecto,

$$T^{-1}AT = \begin{pmatrix} \lambda & \epsilon \\ 0 & \lambda \end{pmatrix},$$

por lo que

$$\left(T^{-1}AT \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} \right) \cdot \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} = \lambda + \epsilon \sin \theta \cos \theta.$$

Entonces, si elegimos $\epsilon < -\lambda$, este producto es negativo. Por tanto, el cambio de variable con T nos deja en la misma situación que el caso 2. La otra dirección de la demostración es inmediata. \square

Anexo B

Documentación código

A continuación, explicamos los diferentes tipos de ejercicios que se han implementado para la asignatura de Ecuaciones Diferenciales. Actualmente, de las tres grandes categorías que se han identificado sobre la asignatura, “Ecuaciones de primer orden”, “Ecuaciones de orden superior” y “Sistemas de dos ecuaciones diferenciales lineales de primer orden cuya matriz de coeficientes tiene determinante no nulo”, se ha comenzado a trabajar en los contenidos de los dos últimos grupos. Destacamos que la “Resolución de ecuaciones de segundo orden” se encuentra en ambos grupos, ya que se considera una ecuación de orden superior y también un sistema porque se puede resolver mediante una transformación a sistema de ecuaciones lineales de primer orden.

B.1. Resolución ecuaciones y sistemas

- `segundo_orden(a, b, c, solve)` – Método que recibe como parámetros los coeficientes a, b, c de la ecuación $ax'' + bx' + cx = 0$. Se transforma la ecuación de segundo orden en el sistema

$$\begin{cases} x' = y, \\ y' = -\frac{c}{a}x - \frac{b}{a}y. \end{cases}$$

Además, en función del valor de `solve` se resuelve opcionalmente el sistema.

- `sfs(a, b, c, d, finished)` – Método que recibe como parámetros los coeficientes a, b, c, d del sistema

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

para obtener el Sistema Fundamental de Soluciones estudiando los autovalores de la matriz de coeficientes. El parámetro `finished` indica si hemos terminado con los cálculos, devolviendo el JSON final, o por el contrario, es el paso intermedio de otro cálculo más extenso, por lo que se retorna como `Paso`.

B.2. Estabilidad

- `sol_explicita(a,b,c,d,finished)` – Método que recibe como parámetros los coeficientes `a,b,c,d` del sistema

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

y devuelve las soluciones explícitas haciendo uso de la función `sfs`. Además, el parámetro `finished` indica si hemos terminado con los cálculos, devolviendo el JSON final, o por el contrario, es el paso intermedio de otro cálculo más extenso, por lo que se retorna como `Paso`.

B.2. Estabilidad

- `clasificar_punto_autoval(a,b,c,d)` – Método que recibe como parámetros los coeficientes `a,b,c,d` del sistema

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

y determina el tipo de estabilidad del punto origen $(0,0)$ estudiando los tipos de los autovalores y autovectores asociados a la matriz de coeficientes.

- `clasificar_traza_det(a,b,c,d)` – Método que recibe como parámetros los coeficientes `a,b,c,d` del sistema

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

y determina el tipo de estabilidad del punto origen $(0,0)$ estudiando la traza y el determinante la matriz de coeficientes.

- `diagramaFase(a, b, c, d, delta, xlimInf, xlimSup, ylimInf, ylimSup, hex)` – Método que recibe como parámetros los coeficientes `a,b,c,d` del sistema

$$\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$$

y representa el diagrama de fases con precisión `delta` y con los límites de ejes en `xlimInf`, `xlimSup`, `ylimInf`, `ylimSup` en el color hexadecimal `hex`.

B.3. Conjugación topológica

- `conjugates(a1, b1, c1, d1, a2, b2, c2, d2)` – Método que recibe como parámetros los coeficientes `a1,b1,c1,d1` y `a2,b2,c2,d2` de los sistemas

$$\begin{cases} x' = a_1x + b_1y, \\ y' = c_1x + d_1y, \end{cases} \quad \text{y} \quad \begin{cases} x' = a_2x + b_2y, \\ y' = c_2x + d_2y, \end{cases}$$

para determinar si los dos sistemas son conjugados topológicamente.

Anexo C

Códigos desarrollados

En esta sección, presentaremos una selección de los códigos más relevantes para ilustrar el funcionamiento y la implementación de nuestra aplicación. Si bien no se incluyen todos los códigos debido a la extensión, los ejemplos seleccionados ofrecerán una visión clara de las principales funcionalidades y características del sistema.

Código Sistema Fundamental de Soluciones

```
1  from .auxiliar.externalFunctions import symbols, exp, Matrix, re, im, cos,
2      ↵ sen, autovalores, autovectores
3  from .comprobaciones.comprobaciones import esReal, comprobarCoeficientes,
4      ↵ matrizDiagonalizable
5  from .resp.obj.Pasos import Pasos
6  from .resp.definirPasos import getPasoMatriz, getPasoSFSMatrices,
7      ↵ getPasoAutovaloresComplejos, getPasoAutovalores
8
9
10 def autovaloresRealesDiagonalizable(aVect):
11     """
12         El sfs es {e^{lambda_1*t}}u, 2^{lambda_2}v
13         siendo u, v los autovectores asociados a los autovalores lambda_1 y
14         ↵ lambda_2 respectivamente.
15         :param aVect:
16         :return:
17         """
18         sol1 = exp(round(aVect[0][0],5) * t) * Matrix(aVect[0][2])
19         sol2 = exp(round(aVect[1][0],5) * t) * Matrix(aVect[1][2])
20         return [sol1, sol2]
21
```

```

22 def autovalorNoDiagonalizable(A, aVect):
23     """
24     El sfs es {e^{lambda*t}}*u, e^{lambda*t} (1+t)u
25     siendo u el autovector asociado al autovalor lambda (doble)
26     :param aVect: autovector
27     :return: sfs
28     """
29     # forma canónica
30     P, J = Matrix(A).jordan_form()
31     v1, v2 = P[:,0], P[:,1]
32
33     sol1 = exp(round(aVect[0][0],5) * t) * v1
34     sol2 = exp(round(aVect[1][0],5) * t) * v1 * t + v2
35     return [sol1, sol2]
36
37
38 def autovalorComplejo(aVect):
39     """
40     El sfs es {e^{pt}} (\cos(qt)u-\sin(qt)w), e^{pt} (\sin(qt)u+\cos(qt)w) }
41     siendo aVect = p+iq y u,v autovectores asociados
42     :param aVect: autovector
43     :return: sfs
44     """
45     p = round(re(aVect[0][0]), 5)
46     q = round(im(aVect[0][0]), 5)
47     u = Matrix(list(map(re, aVect[0][2])))
48     w = Matrix(list(map(im, aVect[0][2])))
49     sol1 = exp(p * t) * (cos(q * t) * u - sen(q * t) * w)
50     sol2 = exp(p * t) * (sen(q * t) * u + cos(q * t) * w)
51     return [sol1, sol2]
52
53
54 def sfs(a, b, c, d, finished):
55     """
56     Función que dada una matriz retorna el Sistema Fundamental de Soluciones
57     :param a: coeficiente 1ª fila 1ª columna
58     :param b: coeficiente 1ª fila 2ª columna
59     :param c: coeficiente 2ª fila 1ª columna
60     :param d: coeficiente 2ª fila 2ª columna
61     :param finished: False si es un paso intermedio
62     :return: JSON del sfs si finished==True, si no, Pasos para hallar sfs
63     """
64     A = comprobarCoeficientes(a, b, c, d)
65     pasos = [getPasoMatriz(A, "La matriz de coeficientes asociado al sistema
66     ↳ es")]
67
68     aVal = autovalores(A)
69     aVect = autovectores(A)

```

Códigos desarrollados

```
69     aVal1 = next(iter(aVal))  # primer autovalor
70
71     keys = list(aVal.keys())
72
73     if esReal(aVal1):  # real
74         pasos.append(getPasoAutovalores(keys,
75                                         "Hallamos los autovalores asociados al sistema
76                                         → para obtener el Sistema Fundamental de
77                                         → Soluciones"))
78
79     if aVal[aVal1] == 1:  # autovalor simple
80         res = autovaloresRealesDiagonalizable(aVect)
81         pasos.append(getPasoSFMatrices(res,
82                                         "Como los autovalores son reales
83                                         → y distintos, el Sistema
84                                         → Fundamental de Soluciones
85                                         → es:"))
86
87     else:  # autovalor doble
88         if matrizDiagonalizable(A):  # autovalor doble diagonalizable
89             res = autovaloresRealesDiagonalizable(aVect)
90             pasos.append(getPasoSFMatrices(res,
91                                         "Como el autovalor es doble y
92                                         → la matriz es
93                                         → diagonalizable, el
94                                         → Sistema Fundamental de
95                                         → Soluciones se calcula
96                                         → igual que en el caso de
97                                         → autovalores reales
98                                         → distintos:"))
99
100    else:  # autovalor doble, no diagonalizable
101        res = autovalorNoDiagonalizable(A, aVect)
102        pasos.append(getPasoSFMatrices(res,
103                                         "Como el autovalor es doble y
104                                         → la matriz no es
105                                         → diagonalizable, el
106                                         → Sistema Fundamental de
107                                         → Soluciones es:"))
108
109    else:  # complejo
110        pasos.append(getPasoAutovaloresComplejos(list(aVal.keys()),
111                                         "Hallamos los autovalores
112                                         → asociados al sistema
113                                         → para obtener el Sistema
114                                         → Fundamental de
115                                         → Soluciones"))
116
117    res = autovalorComplejo(aVect)
118    pasos.append(getPasoSFMatrices(res,
119                                         "Como los autovalores son complejos,
120                                         → el Sistema Fundamental de
121                                         → Soluciones es:"))
```

```
95
96     return Pasos(pasos).toJson() if finished else (pasos, res)
```

Código solución explícita

```
1  from .comprobaciones.comprobaciones import comprobarCoeficientes
2  from .resp.obj.Pasos import Pasos
3  from .sfs import sfs
4  from .resp.definirPasos import getPasoSolExplicita
5
6  '''-----EXPLÍCITAMENTE-----'''
7
8  def sol_explicita(a: float, b: float, c: float, d: float, finished: bool):
9      """
10         Función que obtiene las soluciones explícitas en función de unas
11             constantes.
12             Para ello utiliza el método que obtiene los Sistemas Fundamentales de
13             Soluciones
14             :param a: coeficiente 1ª fila 1ª columna
15             :param b: coeficiente 1ª fila 2ª columna
16             :param c: coeficiente 2ª fila 1ª columna
17             :param d: coeficiente 2ª fila 2ª columna
18             :param finished: False si es un paso intermedio
19             :return: JSON si finished==True, y Pasos en caso contrario, ya que se
20             """
21
22     comprobarCoeficientes(a, b, c, d)
23     sistfs = sfs(a, b, c, d, False) # pasamos coeficientes e indicamos que
24         ↳ buscamos la sol. explícita
25     pasos, sols = sistfs[0], sistfs[1]
26     pasos.append(getPasoSolExplicita(sols[0], sols[1],
27                                         "La solución explícita es la combinación
28                                         ↳ lineal de los elementos en el
29                                         ↳ Sistema Fundamental de Soluciones"))
30
31     return Pasos(pasos).toJson() if finished else (pasos, sols)
```

Código clasificación mediante autovalores

```
1  from latexifier import latexify
2
3  from .auxiliar.externalFunctions import re, autovalores, autovectores,
4      ↳ det_matriz
5  from .comprobaciones.comprobaciones import comprobarCoeficientes, esReal,
6      ↳ matrizDiagonalizable
```

Códigos desarrollados

```
5   from .resp.definirPasos import getPasoMatriz, getPaso, getPasoAutovalores,
→     getPasoAutovaloresComplejos
6   from .resp.obj.Paso import Paso
7   from .resp.obj.Pasos import Pasos
8
9
10  def c_p_a_real_distinto(aVect) -> Paso:
11      """
12          Clasificación del punto de equilibrio cuando los autovalores son reales y
→            distintos
13          :param aVect: autovalores, multiplicidad y autovectores
14          :return: paso
15      """
16      aVal1 = aVect[0][0]
17      aVal2 = aVect[1][0]
18      if (aVal1 < 0 < aVal2) or (aVal1 > 0 > aVal2):
19          signo, tipo = "de diferente signo", "punto de silla"
20      elif aVal1 < 0 and aVal2 < 0:
21          signo, tipo = "negativos", "punto estable"
22      else:
23          signo, tipo = "positivos", "punto inestable"
24      pasos = Paso('', '', "Como los dos autovalores son reales y {}, el punto
→      ${0,0}$ es un {}.".format(signo, tipo))
25      return pasos
26
27
28  def c_p_a_complejo(aVect) -> list[Paso]:
29      """
30          Clasificación del punto de equilibrio cuando los autovalores son
→            complejos (y distintos)
31          :param aVect: autovalores, multiplicidad y autovectores
32          :return: paso
33      """
34      aVal = aVect[0][0]
35      alpha = re(aVal)
36      signo = "menor que" if alpha < 0 else ("mayor que" if alpha > 0 else
→      "igual a")
37      tipo = "foco estable" if alpha < 0 else ("foco inestable" if alpha > 0
→      else "centro estable")
38      paso = 'Re(lambda_1)=Re(lambda_2)={}.'.format(latexify(alpha))
39      pasoLatex = '{}(\lambda_1)={}(\lambda_2) =
→      {}'.format('\\operatorname{Re}', '\\operatorname{Re}',
→      latexify(alpha))
40      pasos = [Paso(paso, pasoLatex, "Como los autovalores son complejos nos
→      fijamos en la parte real"),
41              Paso('', '', "Como la parte real es {} cero, el punto ${0,0}$ es
→      un {}.".format(signo, tipo))]
42
```

```

43     return pasos
44
45
46 def c_p_a_doble_diag():
47     """
48     Clasificación del punto de equilibrio cuando los autovalores iguales y la
49     → matriz diagonalizable
50     :param aVal: autovalores, multiplicidad y autovectores
51     :return: paso
52     """
53     pasos = Paso(' ', '',
54                 "Como el autovalor $\\lambda$ es doble y la matriz $A$ es
55                 → diagonalizable, el punto $(0,0)$ es un punto estelar.")
56
57     return pasos
58
59
60 def c_p_a_doble_no_diag(aVal):
61     """
62     Clasificación del punto de equilibrio cuando los autovalores son iguales
63     → y la matriz no es diagonalizable
64     :param aVal: autovalores, multiplicidad y autovectores
65     :return: paso
66     """
67     signo = "positivo" if aVal > 0 else "negativo"
68     tipo = "inestable" if aVal > 0 else "estable"
69     pasos = Paso(' ', '',
70                 "Como el autovalor $\\lambda$ es doble, la matriz $A$ no
71                 → es diagonalizable y el autovalor $\\lambda={} es {},$,
72                 → el punto $(0,0)$ es un nodo impropio ${}.".format(
73                     latexify(aVal), signo, tipo))
74
75     return pasos
76
77     """
78
79     Indica el tipo de punto de equilibrio que es el origen en función de cómo
80     → es el autovalor
81     :param a: coeficiente 1ª fila 1ª columna
82     :param b: coeficiente 1ª fila 2ª columna
83     :param c: coeficiente 2ª fila 1ª columna
84     :param d: coeficiente 2ª fila 2ª columna
85     :return: JSON
86     """
87
88     A = comprobarCoeficientes(a, b, c, d)

```

Códigos desarrollados

```
85     det = det_matriz(A)
86     pasos = [getPasoMatriz(A, "Esta es la matriz de coeficientes asociado al
87         sistema")]
88
88     paso = 'det={}'.format(latexify(det))
89     pasoLatex = '\\det(A)={}'.format(latexify(det))
90     pasos.append(Paso(paso, pasoLatex,
91                         "Como el determinante es ${}\neq 0$, el único punto de
92                         \u2192 equilibrio es el origen $(0,0)".format(
93                             latexify(det))))
93     aVal = autovalores(A)
94     aVect = autovectores(A)
95     aVal1 = next(iter(aVal)) # primer autovalor
96     keys = list(aVal.keys())
97     if esReal(aVal1): # real
98         pasos.append(getPasoAutovalores(keys,
99                               "Hallamos los autovalores asociados
100                                \u2192 al sistema"))
100    if aVal[aVal1] == 1: # autovalor simple
101        pasos.append(c_p_a_real_distinto(aVect))
102    else: # autovalor doble
103        if matrizDiagonalizable(A): # autovalor doble diagonalizable
104            pasos.append(c_p_a_doble_diag())
105        else: # autovalor doble, no diagonalizable
106            pasos.append(c_p_a_doble_no_diag(aVal1))
107    else: # complejo
108        pasos.append(getPasoAutovaloresComplejos(keys,
109                                         "Hallamos los autovalores
110                                         \u2192 asociados al sistema"))
111
112    pasos = pasos + c_p_a_complejo(aVect)
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
848
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1067
1068
1069
1070
1071
1072
1073
1074
1075
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1670
1671
1671
1672
1672
1673
1673
1674
1674
1675
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1680
1681
1681
1682
1682
1683
1683
1684
1684
1685
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1690
1691
1691
1692
1692
1693
1693
1694
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1700
1701
1701
1702
1702
1703
1703
1704
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1710
1711
1711
1712
1712
1713
1713
1714
1714
1715
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1720
1721
1721
1722
1722
1723
1723
1724
1724
1725
1725
1726
1726
1727
1727
1728
1728

```

```

11 def clasif_complejos(T: float, disc: float):
12     """
13     Clasificación para cuando los autovalores son complejos
14     :param T: traza
15     :param disc: discriminante
16     :return: pasos
17     """
18     paso = 'disc={}'.format(latexify(disc))
19     pasoLatex = '\Delta = T^2-4D = {}'.format(latexify(disc))
20
21     pasos = [Paso(paso, pasoLatex,
22                    "El discriminante es $\Delta = {} < 0$. Por lo que los
23                     ↪ autovalores son complejos".format(
24                         latexify(disc)))]
25     signo = "menor que" if T < 0 else ("mayor que" if T > 0 else "igual a")
26     tipo = "foco estable" if T < 0 else ("foco inestable" if T > 0 else
27                     "centro estable")
28     aVal = "negativa" if T < 0 else ("positiva" if T > 0 else "igual a 0")
29     pasos.append(
30         Paso(' ', ' ', "Como el determinante $T = {}$ es {} 0, la parte real
31                     ↪ del autovalor será {} y por tanto, \
32                         el punto $(0,0)$ es un {}".format(latexify(T), signo, aVal,
33                                         ↪ tipo)))
34     return pasos
35
36
37 def clasif_reales_distintos(T: float, D: float, disc: float):
38     """
39     Clasificación para cuando los autovalores son reales y distintos
40     :param T: traza
41     :param D: determinante
42     :param disc: discriminante
43     :return: pasos
44     """
45     paso = 'disc={}'.format(latexify(disc))
46     pasoLatex = '\Delta = T^2-4D = {}'.format(latexify(disc))
47     pasos = [Paso(paso, pasoLatex, "El discriminante es $\Delta ={} > 0$.
48                     ↪ Por lo que los autovalores son reales \
49                         y diferentes".format(latexify(disc)))]
50     signo = "el determinante $D={}$ es menor que 0".format(D) if D < 0 else (
51         "el determinante $D={} y la traza $T={} son mayores que
52             0".format(D, T) if T > 0 else
53             "el determinante $D={} es mayor que 0 y la traza $T={}$
54                 ↪ negativa".format(D, T))
55     tipo = "punto de silla" if D < 0 else (
56         "punto inestable" if T > 0 else "punto estable")
57     aVal = "de signo contrario" if D < 0 else (
58         "positivos" if T > 0 else "negativos")

```

Códigos desarrollados

```
52     pasos.append(Paso('', '', "Como {}, los autovalores son {}, el punto  
→  $(0,0)$ es un {}.".format(signo, aVal, tipo)))  
53     return pasos  
54  
55  
56 def clasif_reales_iguales(T: float, disc: float, b: float, c: float, A):  
    """  
        Clasificación para cuando los autovalores son reales e iguales  
    :param T: traza  
    :param disc: discriminante  
    :param b: coeficiente fila 1 columna 2  
    :param c: coeficiente fila 2 columna 1  
    :param A: matriz  
    :return: pasos  
    """  
    paso = 'disc={}'.format(latexify(disc))  
    pasoLatex = '\\Delta = T^2-4D = {}'.format(latexify(disc))  
    pasos = [  
        Paso(paso, pasoLatex, "El discriminante $\\Delta {} = 0$. Por lo que  
→  los autovalores son reales e iguales."),  
        getPasoMatriz(A, "Esta es la matriz de coeficientes y nos fijamos en  
→  los valores de la diagonal secundaria.")  
    ]  
    if b == 0 and c == 0:  
        pasos.append(Paso('', '', "Como $b=c=0$, la matriz es diagonalizable  
→  y el punto $(0,0)$ es un punto estelar."))  
    else:  
        cond_b = "$b={}\\neq 0$".format(b) if b != 0 else ""  
        cond_c = "$c={}\\neq 0$".format(c) if c != 0 else ""  
        pasoLatex = cond_b + (' y ' if b != 0 and c != 0 else '') + cond_c  
        tipo = "inestable" if T > 0 else "estable"  
        Tsigno = '>0' if T > 0 else '<0'  
        pasos.append(Paso('', '', "Como {} la matriz no es diagonalizable, y  
→  puesto que $T = {} ${}, el punto \\  
→  $(0,0)$ es un nodo impropio {}.".format(pasoLatex, T,  
→  Tsigno, tipo)))  
    return pasos  
85  
86  
87 def clasificar_traza_det(a: float, b: float, c: float, d: float):  
    """  
        Indica el tipo de punto de equilibrio que es el origen en función de la  
→  traza y el determinante de la matriz  
    :param a: coeficiente 1ª fila 1ª columna  
    :param b: coeficiente 1ª fila 2ª columna  
    :param c: coeficiente 2ª fila 1ª columna
```

```

93     :param d: coeficiente 2ª fila 2ª columna
94     :return: JSON
95     """
96
97     A = comprobarCoeficientes(a, b, c, d)
98     T = a + d # traza
99     D = a * d - b * c # determinante
100
101    paso = 'T={}, D={}'.format(latexify(T), latexify(D))
102    pasoLatex = 'T={}, \quad D={}'.format(latexify(T), latexify(D))
103    pasos = [Paso('lambda = T ** 2 - 4 * D', '\lambda = \frac{T}{4} - \sqrt{T^2 - 4D}', 'Recordemos que siendo $T$ la traza y $D$ el determinante, los autovalores se pueden expresar como:'.format(round(T, 3), round(D, 3))), Paso(paso, pasoLatex, "Los valores de la traza y el determinante son:")]
104
105    disc = T ** 2 - 4 * D
106
107    if disc < 0:
108        pasos = pasos + clasif_complejos(T, disc)
109    elif disc > 0:
110        pasos = pasos + clasif_reales_distintos(T, D, disc)
111    else: # disc = 0
112        pasos = pasos + clasif_reales_iguales(T, disc, b, c, A)
113
114    return Pasos(pasos).toJson()

```

Código para determinar si son conjugadas

```

1  from .resp.definirPasos import autovalList2latex
2  from .resp.obj.Paso import Paso
3  from .resp.obj.Pasos import Pasos
4  from .auxiliar.externalFunctions import Matrix, autovalores, re
5
6  """Determina si dos sistemas son conjugados topológicamente"""
7
8
9  def notHyperbolic(aVals) -> bool:
10    """
11      Determina si una matriz es hiperbólica
12      :param aVals: lista de autovalores
13      :return: True si se encuentra algún autovalor con parte real nula
14    """
15    return any(re(n) == 0 for n in aVals)
16
17

```

Códigos desarrollados

```
18 def countNegativeEigenValues(aVals) -> int:
19     """
20     Cuenta el número de autovalores negativos
21     :param aVals: lista de autovalores
22     :return: número de autovalores negativos
23     """
24     return sum(1 for i in aVals if re(i) < 0)
25
26
27 def conjugates(a1: float, b1: float, c1: float, d1: float, a2: float, b2:
28     float, c2: float, d2: float):
29     """
30     Determina si dos sistemas son conjugados topológicos
31     :param a1: coeficiente 1ª fila 1ª columna del primer sistema
32     :param b1: coeficiente 1ª fila 2ª columna del primer sistema
33     :param c1: coeficiente 2ª fila 1ª columna del primer sistema
34     :param d1: coeficiente 2ª fila 2ª columna del primer sistema
35     :param a2: coeficiente 1ª fila 1ª columna del segundo sistema
36     :param b2: coeficiente 1ª fila 2ª columna del segundo sistema
37     :param c2: coeficiente 2ª fila 1ª columna del segundo sistema
38     :param d2: coeficiente 2ª fila 2ª columna del segundo sistema
39     :return: JSON
40     """
41
42     paso = 'A1 = [[{},{}], [{}:{}]], '.format(a1, b1, c1, d1) + 'A2 =
43         [[{},{}], [{}:{}]]'.format(a2, b2, c2, d2)
44     pasoLatex = 'A_1 = \begin{pmatrix} {} & {} \\ {} & {} \end{pmatrix}, \quad '
45     pasoLatex += 'A_2 = \begin{pmatrix} {} & {} \\ {} & {} \end{pmatrix}'.format(a1, b1, c1, d1)
46     pasoLatex += '\begin{pmatrix} {} & {} \\ {} & {} \end{pmatrix}'.format(a2, b2, c2, d2)
47
48     pasos = [Paso(paso, pasoLatex, "Dos sistemas son conjugados
49             topológicamente si las matrices son hiperbólicas \
50                 y tiene mismo número de autovalores negativos. Las matrices
51             asociadas a los sistemas son:")]
52     A1, A2 = Matrix([[a1, b1], [c1, d1]]), Matrix([[a2, b2], [c2, d2]])
53
54     aVals1, aVals2 = autovalores(A1), autovalores(A2)
55
56     l_aVals1, l_aVals2 = [], []
57     for a in aVals1.keys():
58         l_aux = [a] * aVals1[a]
59         l_aVals1 = l_aVals1 + l_aux
60
61     for a in aVals2.keys():
62         l_aux = [a] * aVals2[a]
63         l_aVals2 = l_aVals2 + l_aux
64
65     notHyperbolic1, notHyperbolic2 = notHyperbolic(l_aVals1),
66         notHyperbolic(l_aVals2)
```

```

60
61     paso = 'Autovalores de la primera matriz = {}'.format(l_aVals1, l_aVals2)
62     matriz = '{}'.format(l_aVals1, l_aVals2)
63     pasoLatex = '\\text{{Autovalores de la primera matriz : }} {} \\\\'
64     pasoLatex += '\\text{{Autovalores de la segunda matriz : }} {}'.format(
65         autovalList2latex(l_aVals1, 1), autovalList2latex(l_aVals2, 2))
66     pasos.append(Paso(paso, pasoLatex, 'Obtenemos los autovalores de ambas
67     matrices: '))
68
69     if notHyperbolic1 and notHyperbolic2:
70         pasos.append(Paso(paso, pasoLatex, 'Ninguna de las matrices son
71             hiperbólicas, luego no podemos hablar de \
72                 conjugaciones topológicas.'))
73     return Pasos(pasos).toJson()
74 elif notHyperbolic1:
75     pasos.append(Paso(paso, pasoLatex, 'La primera matriz no es
76             hiperbólica, luego no podemos hablar de \
77                 conjugaciones topológicas.'))
78     return Pasos(pasos).toJson()
79 elif notHyperbolic2:
80     pasos.append(Paso(paso, pasoLatex, 'La segunda matriz no es
81             hiperbólica, luego no podemos hablar de \
82                 conjugaciones topológicas.'))
83     return Pasos(pasos).toJson()
83
84 negativos1, negativos2 = countNegativeEigenValues(l_aVals1),
85     countNegativeEigenValues(l_aVals2)
86
87 explConj = "Ambas matrices tienen el mismo número de autovalores
88     negativos: ${} $, " \
89         "luego los sistemas son conjugados
90             topológicos.".format(negativos1)
91 explNoConj = "Las dos matrices tienen diferente número de autovalores
92     negativos. " \
93         "Mientras que el primer sistema tiene ${} $ autovalores
94             negativos, el segundo tiene ${} $, " \
95                 "luego los dos sistemas no son conjugados
96                     topológicos.".format(negativos1, negativos2)
97
98 expl = explConj if negativos1 == negativos2 else explNoConj
99 pasos.append(Paso(' ', ' ', expl))
100
101 return Pasos(pasos).toJson()

```

Códigos desarrollados

Código para la resolución de ecuaciones de segundo orden

```
1  from latexifier import latexify
2  import sympy as sy
3
4  from .auxiliar.externalFunctions import integrar
5  from .explicitSFS import sol_explicita
6  from .resp.obj.Paso import Paso
7
8  from .resp.obj.Pasos import Pasos
9
10 '''-----ECUACIÓN DE SEGUNDO ORDEN-----'''
11
12
13 def segundo_orden(a: float, b: float, c: float, solve: bool):
14     """
15         Dada la ecuación:  $ax''+bx'+cx = 0$ 
16         Transformamos en el sistema  $x' = y$ 
17                          $y' = -(c/a)x - (b/a)y$ 
18
19         :param a: coeficiente acompañando al término  $x''$ 
20         :param b: coeficiente acompañando al término  $x'$ 
21         :param c: coeficiente acompañando al término  $x$ 
22         :param solve: True si se quiere resolver el sistema
23         :return: JSON con pasos
24     """
25
26     x, y, c1, c2, t = sy.symbols('x, y, c1, c2, t')
27     pasos = [Paso('x''=y', 'x''=y', "Introducimos el cambio de variable"),
28               Paso('x'' = - (c/a)x - (b/a)y',
29                     'x''=-\frac{c}{a}x-\frac{b}{a}y',
30                     'Entonces, $x''=y$', y despejando $x''')]
31
32     sist = latexify(-(c / a) * x - (b / a) * y)
33     pasoLatex = '\\begin{cases} x'=y \\\\" y'= ' + sist + '\\end{cases}'
34
35     pasos.append(
36         Paso('x' = y, y' = -(c/a)x-(b/a)y', pasoLatex, 'El sistema
37             equivalente a la ecuación de segundo grado es'))
38
39     if solve:
40         rpasos, sol = sol_explicita(0, 1, -c / a, -b / a, False)
41         pasos = pasos + rpasos
42         ysol = c1 * sol[0][1] + c2 * sol[1][1]
43         ysol_int = integrar(ysol, t)
44         pasoLatex = 'x =' + latexify(ysol_int)
45         paso = 'x={}.'.format(str(ysol_int))
46         pasos.append(Paso(paso, pasoLatex, 'Sabiendo que $y={}$ y deshaciendo
47             el cambio de variable $x''=y$', format(latexify(ysol))))
```

```
44
45     return Pasos(pasos).toJson()
```

Conexiones

A continuación una selección del archivo `ecsdif_api.py`, el cual contiene la implementación de la API para la resolución de ecuaciones diferenciales.

```
1  from flask import Blueprint
2  from flask import Response
3  from sympy import Matrix
4
5  from .ecsdif.src.auxiliar.externalFunctions import det_matriz
6  from .ecsdif.src.classifyautoval import clasificar_punto_autoval
7  from .ecsdif.src.classifytracedet import clasificar_traza_det
8  from .ecsdif.src.comprobaciones.comprobaciones import string2float
9  from .ecsdif.src.conjugadas import conjugates
10 from .ecsdif.src.diagramphases import diagramaFase
11 from .ecsdif.src.exc.Exceptions import ExceptionDetZero
12 from .reader import transform
13
14 from .ecsdif.src.sfs import sfs
15 from .ecsdif.src.explicitSFS import sol_explicita
16 from .ecsdif.src.secondorder import segundo_orden
17
18 bp = Blueprint("ecsdif", __name__, url_prefix="/ecsdif")
19
20
21 @bp.route("/info", methods=['GET'])
22 def ecsdif_info_general():
23     """Información de la asignatura"""
24     return {
25         "titulo": "Sistemas de Ecuaciones Diferenciales",
26         "descripcion": "Obtiene las soluciones explícitas de los sistemas de
27                         ecuaciones diferenciales de primer \
28                             orden de 2 ecuaciones de primer orden cuyo
29                         determinante no es nulo. También clasifica las soluciones en \
30                             función del tipo de punto de equilibrio, mostrando el
31                         plano de fases.",
32         "algoritmo": "Las soluciones explícitas se obtienen hallando primero
33                         el Sistema Fundamental de Soluciones.\n \
34                             La clasificación del punto de equilibrio se puede
35                         realizar tanto estudiando los autovalores, como \
36                             estudiando la traza y el determinante.\n \
37                             Se representa el diagrama de fases mediante una
38                         librería de Python, Plotly y el usuario puede \
```

Códigos desarrollados

```
33                     establecer los límites de los ejes, así como la
34             precisión de la gráfica."
35         }
36
37     @bp.route("/info/sistemas", methods=['GET'])
38     def ecstdif_info_sistemas():
39         """Contenido resolver: sfs y solución explícita"""
40         return {
41             "titulo": "Resolución de sistemas de ecuaciones diferenciales",
42             "descripcion": "Dado un sistema  $\begin{cases} x' = ax + by, \\ y' = cx + dy, \end{cases}$  con  $a, b, c, d \in \mathbb{R}$ , \n
43             y  $\begin{vmatrix} a & b \\ c & d \end{vmatrix} \neq 0$  se indica el Sistema Fundamental de Soluciones (SFS)
44             y la solución explícita general.",
45             "algoritmo": "Se estudian los autovalores de la matriz de
46             coeficientes asociados al sistema para obtener la SFS. \
47             La solución explícita es la combinación lineal del
48             Sistema Fundamental de Soluciones."
49
50     # ...
51
52     @bp.route("/info/conjugacion", methods=['GET'])
53     def ecstdif_info_conjugacion():
54         """Contenido determinar conjugaciones topológicas"""
55         return {
56             "titulo": "Conjugación topológica",
57             "descripcion": "Dados dos sistemas \
58             \begin{cases} x' = a_1x + b_1y, \\ y' = c_1x + d_1y, \end{cases} \quad \
59             \begin{cases} x' = a_2x + b_2y, \\ y' = c_2x + d_2y, \end{cases} \
60             con todos los coeficientes números reales y los
61             determinantes distintos del 0, \
62             se determina si los sistemas son conjugados
63             topológicos.",
64             "algoritmo": "Dos sistemas son conjugados topológicos si sus matrices
65             de coeficientes lo son. \
66             Se comprueba que las matrices sean hiperbólicas y que
67             tengan el mismo número de autovalores negativos."
68         }
69
70     # ...
71
72     @bp.route("/sist/sfs/explicit/<a>&<b>&<c>&<d>", methods=['GET'])
```

```

69 def ecsdif_explicitSolution(a: str, b: str, c: str, d: str):
70     """
71     Llamada para obtener las soluciones explicitas
72     :param a: coeficiente 1ª fila 1ª columna
73     :param b: coeficiente 1ª fila 2ª columna
74     :param c: coeficiente 2ª fila 1ª columna
75     :param d: coeficiente 2ª fila 2ª columna
76     :return: Response
77     """
78     try:
79         a, b, c, d = string2float(a), string2float(b), string2float(c),
80             ↪ string2float(d)
81     except Exception:
82         return Response("Error: por favor, introduzca números reales", 400)
83
84     if det_matriz([[a, b], [c, d]]) == 0:
85         return Response('Error: el determinante de la matriz de coeficientes
86             ↪ no puede ser nula.', 400)
87
88     try:
89         return sol_explicita(a, b, c, d, True)
90     except Exception:
91         return Response("Error interno del servidor", 500)
92
93 # ...
94
95 @bp.route("/paint/phases-diagram/<a>&<b>&<c>&<d>&<delta>&<xlimInf>&<xlimSup>&
96     ↪ <ylimInf>&<ylimSup>&<hex>", methods=['GET'])
97 def ecsdif_get_diagramaFases(a: str, b: str, c: str, d: str, delta: str,
98     ↪ xlimInf: str, xlimSup: str, ylimInf: str,
99                 ylimSup: str, hex: str):
100    """
101    Llamada para pintar el diagrama de fases
102    :param a: coeficiente 1ª fila 1ª columna
103    :param b: coeficiente 1ª fila 2ª columna
104    :param c: coeficiente 2ª fila 1ª columna
105    :param d: coeficiente 2ª fila 2ª columna
106    :param delta: precisión de la representación
107    :param xlimInf: límite inferior del eje de abscisas
108    :param xlimSup: límite superior del eje de abscisas
109    :param ylimInf: límite inferior del eje de ordenadas
110    :param ylimSup: límite superior del eje de ordenadas
111    :param hex: color en hexadecimal '#-----'
112    :return: Response
113    """
114    try:
115        a, b, c, d = string2float(a), string2float(b), string2float(c),
116            ↪ string2float(d)

```

Códigos desarrollados

```
112     delta = string2float(delta)
113     xlimInf, xlimSup = string2float(xlimInf), string2float(xlimSup)
114     ylimInf, ylimSup = string2float(ylimInf), string2float(ylimSup)
115 except Exception:
116     return Response("Error: por favor, introduzca números reales", 400)
117
118 if det_matriz([[a, b], [c, d]]) == 0:
119     return Response('Error: el determinante de la matriz de coeficientes
120                   → no puede ser nula.', 400)
121
122 if xlimInf >= xlimSup or ylimInf >= ylimSup:
123     return Response('Error: por favor, introduzca límites correctamente')
124
125 try:
126     return diagramaFase(a, b, c, d, delta, xlimInf, xlimSup, ylimInf,
127                         → ylimSup, hex)
128 except Exception:
129     return Response("Error interno del servidor", 500)
# ...
```