UNIVERSIDAD
DE GRANADA

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO DE FIN DE GRADO

Métodos para la resolución de ecuaciones integrales y su integración en un sistema para la simulación de la distribución de temperatura en edificios mediante el uso de servicios web

Presentado por:
David Cantón Ruiz

Tutor:
Manuel Ruiz Galán
Departamento de Matemática Aplicada

Ángel Ruiz Zafra
Departamento de Lenguajes y Sistemas Informáticos

Curso académico 2023-2024

Métodos para la resolución de ecuaciones integrales y su integración en un sistema para la simulación de la distribución de temperatura en edificios mediante el uso de servicios web

David Cantón Ruiz

David Cantón Ruiz *Métodos para la resolución de ecuaciones integrales y su integración en un sistema para la simulación de la distribución de temperatura en edificios mediante el uso de servicios web*.

Trabajo de fin de Grado. Curso académico 2023-2024.

Responsable de tutorización

Manuel Ruiz Galán
Departamento de Matemática Aplicada

Ángel Ruiz Zafra
Departamento de Lenguajes y Sistemas Informáticos

Doble Grado en Ingeniería Informática y Matemáticas

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Universidad de Granada

Declaración de originalidad

D./Dña. David Cantón Ruiz

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 17 de junio de 2024

Fdo: David Cantón Ruiz

Índice general

Summary

The principal objective of this project lies in the creation of a visually intuitive and highly functional tool that allows studying the behavior of temperatures throughout the day in a building. This tool will facilitate the modification of the building's parameters and structure in a comfortable and dynamic manner, enabling users to observe how temperatures vary in response to these changes. To achieve this, a mathematical model will be applied to predict temperature variations. Additionally, this tool aims to be accessible and beneficial for other users. To support this, an appendix containing a guide for its installation and use has been added, allowing it to be utilized for various purposes, regarding academic, professional, or personal contexts.

The main objective of this final degree project in the field of computer science is to design, implement, and deploy a system that, using mathematical models, allows users to simulate and predict the temperature inside a building based on certain parameter configurations. It will also integrate a third-party API (Application Programming Interface) to capture real-time meteorological data (such as external temperature and humidity) and provide one-week forecasts. The system to be developed will follow a Service-Oriented Architecture (SOA) model, offering a set of services to be consumed by the user's application. The entire system will be integrated into a RESTful API and a web platform application. The technologies used, Python for backend development and React for the frontend, will ensure the solution's extensibility, scalability, and usability.

All the specified purposes have been met, except for the integration of the real-time external API, which has been left as future work for a potential extension of the project, considering it excessive for a project of this scope.

In the field of mathematics, the objective is to study some of the methods for solving or approximating solutions to certain integral equations, as well as applying some of these methods to the study of a mathematical model for temperature distribution inside a building with multiple rooms. Some of the concepts, algorithms, and widely used results will be compiled, with a focus on fixed-point methods. Finally, a heating and cooling model for a building will be presented, leading to a problem that will be solved using some of the previously studied techniques. Additionally, various simulations will allow us to realistically analyze a building's energy efficiency in terms of the parameters and functions that describe the model.

All initially proposed objectives in the field of mathematics have been achieved.

The project is structured into two main parts: the first one, which involves a more theoretical and mathematical study, and the second, which focuses on computer science with a more practical approach, demonstrating its utility. Both parts complement each other exceptionally well, as the simulator to be implemented is based on the mathematical model developed later in this work. The mathematical model provides a solid theoretical foundation, detailing the equations and principles governing the phenomenon we are studying. This approach allows us not only to validate the simulator but also to adjust its parameters accurately to reflect real

conditions. Moreover, the simulator will serve as a practical tool enabling us to visualize and analyze the results obtained from the mathematical model, facilitating data understanding and interpretation. Thus, integrating both parts provides a comprehensive and detailed view of the problem, ensuring greater accuracy and robustness in our conclusions.
Additionally, in Appendix A, an installation and access manual for the application is shown so that anyone can access the code and use it.

The first part consists of six sections, starting with the classification of the main types of integral equations we will cover. In the second section, we will conduct a more theoretical study on the existence and uniqueness of the solution specifically for the linear Volterra integral equation of the second kind, using the Geometric Series Theorem and some of its variants. Subsequently, in the third and fourth sections, various methods for solving these equations including both scalar and vector forms will be studied and some examples will be provided. The fifth section will illustrate the relationship between differential equations and Volterra equations, allowing us to apply this model to the following section to the use case of building heating and cooling, where we will see many examples of this application.

The second part consists of two sections that begin by setting the theoretical framework related to software architecture, showing the characteristics of the main types of architecture found in applications, along with several examples of current applications using these architectures. Lastly, we will have the design, implementation, and deployment section of the simulator, in which we will show and explain the characteristics of the technologies used, together with the steps in designing and developing the simulator, providing some usage examples.

Finally, there will be a section regarding the conclusions and future work, highlighting the most important aspects of the project, along with possible updates that allow its scalability and improvement, enabling continued work on this topic in the future.

Introducción

La motivación principal de este trabajo radica en la creación de una herramienta visualmente intuitiva y altamente funcional que permita estudiar el comportamiento de las temperaturas a lo largo del día en un edificio. Esta herramienta facilitará la modificación de los parámetros y la estructura del edificio de manera cómoda y dinámica, permitiendo a los usuarios observar cómo varían las temperaturas en respuesta a estos cambios. Para ello, se aplicará un modelo matemático que permitirá realizar las predicciones de estas temperaturas. Además, se busca que esta herramienta sea accesible y beneficiosa para otras personas, se ha añadido como apéndice una guía para la instalación y uso de la misma, permitiéndo ser utilizada para otros fines, ya sea en el ámbito académico, profesional o personal.

El objetivo principal de este TFG en el ámbito informático es diseñar, implementar y desplegar un sistema que, haciendo uso de los modelos matemáticos, permita al usuario simular y predecir la temperatura en el interior de un edificio de acuerdo a la configuración de ciertos parámetros. A su vez integrará una API (Application Programming Interface) de terceros para la captura en tiempo real de datos meteorológicos (temperatura y humedad exterior) así como datos dados y predicciones a una semana vista. El sistema a desarrollar seguirá un modelo de Arquitectura Orientada a Servicios (SOA), ofreciendo un conjunto de servicios a consumir por la aplicación del usuario. Se integrará todo el sistema resultando en una API RESTful y una aplicación en una plataforma web. Las tecnologías que vamos a utilizar, Python para el desarrollo del backend y React en el caso del frontend nos garantizan la extensibilidad, escalabilidad y usabilidad de la solución.
Se han cumplido todos los objetivos especificados anteriormente, exceptuando la integración de la API externa de tiempo real, que se ha dejado como trabajo a futuro para una posible ampliación del trabajo, debido a que se ha considerado que era excesivo para un trabajo de estas características.

El objeto en el ámbito de las matemáticas consiste en realizar un estudio de algunos de los métodos de resolución o aproximación de la solución para ciertas ecuaciones integrales, así como la aplicación de algunos de ellos al estudio de un modelo matemático para la distribución de la temperatura en el interior de un edificio con varias estancias. Se recopilarán algunos de los conceptos, algoritmos y resultados de uso extendido, incidiendo en los de punto fijo. Finalmente, se presentará un modelo de calentamiento y enfriamiento de un edificio, que conduce a un problema que resolveremos con algunas de las técnicas previamente estudiadas. Además, diversas simulaciones permitirán analizar de manera realista la eficiencia energética de un edificio en términos de los parámetros y funciones que describen el modelo.
Se han cumplido todos los objetivos que inicialmente se propusieron en el ámbito de las matemáticas.

El trabajo está estructurado en dos partes principales, una primera parte en la cual se realiza un estudio matemático y más teórico, y una segunda parte centrada en la informática, la cual pretende dar un enfoque más práctico y mostrar la utilidad del trabajo. Ambas partes se

complementan de manera excepcional, ya que el simulador que implementaremos se fundamenta en el modelo matemático que desarrollaremos más adelante en este trabajo. El modelo matemático proporcionará una base teórica sólida, detallando las ecuaciones y principios que rigen el fenómeno que estamos estudiando. Este enfoque nos permitirá no solo validar el simulador, sino también ajustar sus parámetros de manera precisa para reflejar con exactitud las condiciones reales. Por otro lado, el simulador servirá como una herramienta práctica que nos permitirá visualizar y analizar los resultados obtenidos del modelo matemático, facilitando la comprensión y la interpretación de los datos. Así, la integración de ambas partes nos proporcionará una visión completa y detallada del problema, garantizando una mayor precisión y robustez en nuestras conclusiones.
Además, en el Apéndice A, se muestra un manual de instalación y acceso a la aplicación para que cualquier persona pueda acceder al código y utilizarlo.

La primera parte consta de seis secciones, comenzando por la clasificación de los principales tipos de ecuaciones integrales que vamos a ver, en la segunda sección realizaremos un estudio más teórico sobre la existencia y unicidad de solución concretamente para la ecuación lineal integral de Volterra de segunda clase, viendo para ello el Teorema de la serie geométrica y algunas de sus variantes. Posteriormente, en la tercera y cuarta sección estudiaremos y daremos ejemplos de varios métodos para resolver estas ecuaciones, tanto de forma escalar como vectorial. La quinta sección ilustrará la relación entre ecuaciones diferenciales y ecuaciones de Volterra, que nos permitirá aplicar este modelo en la sexta sección al caso de uso del calentamiento y enfriamiento de edificios, sección en la cual veremos gran cantidad de ejemplos de esta aplicación.

La segunda parte está compuesta de dos secciones que comienzan sentando el marco teórico relacionado con la arquitectura software, mostrando las características de los principales tipos de arquitectura que podemos encontrar en las aplicaciones junto con varios ejemplos de aplicaciones actuales que utilizan estas arquitecturas. Finalmente tendremos el apartado de diseño, implementación y despliegue del simulador, en el cual mostraremos y explicaremos las características de las tecnologías que hemos utilizado, junto con los pasos en el diseño y desarrollo del simulador, dando algunos ejemplos de su utilización.

Por último, un apartado de conclusiones y trabajo futuro, en el cual se destacarán los aspectos más importantes del trabajo, junto a posibles actualizaciones que permiten su escalabilidad y mejora, y permiten seguir trabajando en este tema a futuro.

1. Clasificación de ecuaciones integrales

Podemos encontrar ecuaciones integrales de muchos tipos distintos. Esto depende principalmente de los límites de integración y del núcleo de la ecuación. A continuación nos enfocaremos en los tipos de ecuaciones integrales.

1.1. Introducción

El objetivo que perseguimos en este primer capitulo es el estudio y clasificación de ecuaciones integrales, para proporcionar una comprensión más profunda de su estructura y comportamiento, y aunque no siempre se indique de forma explícita, **todas las ecuaciones integrales con las que vamos a trabajar son lineales**. A través de un análisis detallado y riguroso, vamos a establecer un marco teórico sólido que permita abordar con mayor eficacia una amplia gama de problemas en contextos prácticos. Además, ilustraremos ejemplos que nos ayudarán a entender más fácilmente la diferencia entre los distintos tipos de ecuaciones integrales.

El modelo de ecuación integral lineal con el que vamos a trabajar es el siguiente:

$$u(x) = f(x) + \int_{g(x)}^{h(x)} K(x,t)u(t)dt, \tag{1.1}$$

donde $g(x)$ y $h(x)$ son los límites de integración, y $K(x,t)$ es una función conocida de dos variables reales, x y t, que llamaremos el *núcleo* de la ecuación integral. Podemos ver cómo la función real desconocida $u(x)$, que queremos determinar, aparece tanto dentro como fuera de la integral, sin embargo, podría aparecer sólo dentro de la misma, siendo así, una ecuación de primera clase. En este trabajo sólo nos centraremos en las ecuaciones de segunda clase como la que tenemos en (1.1).

Las funciones $f(x)$ y $K(x,t)$ son conocidas de antemano, y los límites de integración pueden ser ambos variables, constantes, o uno variable y otro constante. Esto será clave para determinar el tipo de ecuación integral que tenemos delante.

Observación 1.1. Nótese que si la función $f(x)$ es idénticamente nula, diremos que la ecuación resultante

$$u(x) = \int_{g(x)}^{h(x)} K(x,t)u(t)dt$$

es *homogénea*.

Señalemos finalmente que hemos tomado [Waz11] como referencia para esta primera sección.

1.2. Ecuaciones integrales de Volterra

En las ecuaciones integrales de Volterra, uno de los límites de integración es una variable y el otro una constante. Para las ecuaciones integrales de Volterra de *segunda clase*, la función

desconocida $u(x)$ aparece tanto dentro como fuera de la integral. Se representa de la siguiente forma:

$$u(x) = f(x) + \int_0^x K(x,t)u(t)dt, \qquad x \in [0,B]. \tag{1.2}$$

Ejemplo 1.1. A continuación mostramos un ejemplo de una ecuación integral de Volterra de segunda clase:

$$u(x) = 5x + 2\int_0^x (x-t)u(t)dt, \qquad x \in [0,4].$$

1.2.1. Ecuaciones integrales singulares

Para continuar vamos a introducir el concepto de ecuación integral de Volterra singular:

Definición 1.1. Diremos que una ecuación integral de Volterra

$$u(x) = f(x) + \int_0^x K(x,t)u(t)dt, \qquad x \in [0,B]$$

es *singular* si satisface alguna de las siguientes condiciones:

- Uno de los límites de integración, o ambos, son infinitos.
- El núcleo $K(x,t)$ no está acotado en algún punto del intervalo de integración.

Vamos a centrarnos en una ecuación particular de segunda clase:

$$u(x) = f(x) + \int_0^x \frac{1}{(x-t)^\alpha}u(t)dt, \qquad 0 < \alpha < 1, \qquad x \in [0,B].$$

Se conoce como *ecuación integral singular débil*. Nótese que el núcleo no está acotado en el límite superior $t = x$.

Ejemplo 1.2. Una ecuación integral singular débil sería:

$$\frac{\sin x}{2} = \int_0^x \frac{1}{\sqrt{x-t}}u(t)dt,$$

1.3. Ecuaciones integrales de Fredholm

En las ecuaciones integrales de Fredholm, los límites de integración son fijos. Además, al igual que con las ecuaciones integrales de Volterra, al estudiar las ecuaciones de segunda clase, la función desconocida $u(x)$ aparece tanto dentro como fuera de la integral. De igual forma que hemos hecho con las ecuaciones de Volterra, vamos a definir la ecuación integral de Fredholm de *segunda clase*:

$$u(x) = f(x) + \int_a^b K(x,t)u(t)dt, \qquad x \in [a,b].$$

Ejemplo 1.3. Un ejemplo de una ecuación integral de Fredholm de segunda clase puede ser el siguiente:

$$u(x) = \sin x + \frac{1}{4}\int_0^2 (x-t)u(t)dt, \qquad x \in [0,2].$$

1.4. Ecuaciones integrales de Volterra-Fredholm

Como curiosidad, estas ecuaciones surgieron del modelo matemático del desarrollo espacio-tiempo de una epidemia, y de varios modelos físicos y biológicos (véase [Waz11]). Normalmente nos aparecen representadas así:

$$u(x) = f(x) + \int_a^x K_1(x,t)u(t)dt + \int_a^b K_2(x,t)u(t)dt, \qquad x \in [a,b]. \tag{1.3}$$

Es interesante ver que (1.3) contiene ecuaciones integrales disjuntas de Volterra y Fredholm, ya que la primera integral tiene un límite variable y la segunda tiene ambos límites fijos. Además, podemos ver que la función $u(x)$ aparece dentro y fuera de la integral, como ocurre en las ecuaciones de segunda clase.

Ejemplo 1.4. Un ejemplo de una ecuación integral de Volterra-Fredholm es el siguiente:

$$u(x) = 9x^3 - 2x + \int_0^x xu(t)dt - \int_0^1 tu(t)dt, \qquad x \in [0,1].$$

2. Existencia y unicidad de solución de la ecuación lineal integral de Volterra de segunda clase

2.1. Introducción

En esta sección nos hemos basado en el texto de [HA09], aunque como se verá, hemos realizado algunas modificaciones como en la demostración del teorema de la serie geométrica.

En la sección anterior hemos presentado una amplia gama de ecuaciones integrales, pero ahora estudiamos con detalle las lineales de Volterra de segunda clase, ya que serán las que permitan describir y resolver satisfactoriamente el modelo de distribución de la temperatura interna de un edificio.
Si reparamos en la propia naturaleza de este tipo de ecuaciones, debemos exigir cierto grado de regularidad (aunque sólo sea continuidad, incluso algún tipo de buen comportamiento integral) a las funciones dato y a la solución. Todo ello se detalla en este epígrafe.

2.2. Teorema de la serie geométrica y sus variantes

Podemos encontrar la aplicación de este teorema habitualmente en análisis numérico y análisis funcional además de matemática aplicada, esto se debe a que es una gran herramienta para analizar si tienen solución los problemas cercanos a otros problemas para los que sí podemos asegurar la existencia de una solución única.

Primero vamos a introducir un resultado muy importante que utilizaremos posteriormente para demostrar este teorema, que garantiza la existencia y unicidad de puntos fijos de ciertas funciones definidas sobre espacios métricos y proporciona un método para encontrarlos.

Definición 2.1. Sea X un espacio de Banach y $T : X \to X$ una aplicación. Se dice que T es *contractiva* si existe una constante c verificando $0 \leqslant c < 1$ tal que $\|T(x) - T(y)\| \leqslant c\|x - y\|$, para cualesquiera $x, y \in X$.

Teorema 2.1. *(Teorema del punto fijo de Banach) Sea X un espacio de Banach y sea $T : X \to X$ una aplicación contractiva en X. Entonces existe un único punto fijo de T.*
Además, para todo punto x_0 de X, la sucesión $\{T^n(x_0)\}_{n=0}^{\infty}$ converge a dicho punto fijo.

Este teorema tiene sentido incluso para espacios métricos completos, pero nosotros no necesitamos ese nivel de generalización.

Como notación, utilizaremos $\mathcal{L}(X)$ para referirnos al espacio de todos los operadores lineales de X en sí mismo, de igual forma $\mathcal{L}(X, Y)$ será el espacio de todos los operadores lineales de X en Y, ambos dotados de su norma usual. Ahora sí, podemos enunciar y demostrar el teorema de la serie geométrica:

Teorema 2.2. *Sean X un espacio de Banach y $L \in \mathcal{L}(X)$. Suponemos*

$$\|L\| < 1.$$

Entonces $I - L$ es una biyección en X, su inversa es un operador lineal y continuo,

$$(I-L)^{-1} = \sum_{n=0}^{\infty} L^n,$$

y

$$\|(I-L)^{-1}\| \leqslant \frac{1}{1-\|L\|}. \tag{2.1}$$

Demostración. Sea $y \in X$, definimos el operador

$$T : X \to X, \qquad T(x) = y + Lx.$$

Vamos a ver que T es contractivo: sean $u, v \in X$, y sea $c = \|L\|$, $0 \leqslant c < 1$; entonces

$$\|Tu - Tv\| = \|y + Lu - (y + Lv)\| = \|Lu - Lv\| = \|L(u-v)\| \leqslant c\|u - v\|.$$

Nótese que en la tercera igualdad hemos aplicado la linealidad de L y en la desigualdad su continuidad. Ahora aplicamos el teorema del punto fijo de Banach y tomamos como punto inicial $x_0 = y$, por tanto tenemos que

$$x_n = y + Lx_{n-1} = \sum_{j=0}^{n} L^j y, \qquad n \geqslant 1$$

converge al único punto fijo de T, $x = Tx$. Es decir,

$$x = y + Lx \Rightarrow (I-L)x = y,$$

y además $x = \sum_{j=0}^{\infty} L^j y$, como queríamos demostrar. Ahora vamos a ver que $(I-L)$ es biyectivo y continuo:

- Ya hemos visto que es sobreyectiva, pues $y \in X$ es arbitrario.
- Inyectiva:

  $$(I-L)u = (I-L)v \Rightarrow 0 = \|u - Lu - v + Lv\| \geqslant \|u - v\| - \|L(u-v)\| \geqslant$$
  $$\geqslant \|u - v\| - \|L\|\|u - v\| = (1 - \|L\|)\|u - v\| \Rightarrow u = v.$$

  (Puesto que $(1 - \|L\|) > 0$).
- El teorema de isomorfismos de Banach (véase [Bre11]) nos asegura la continuidad.

A la vista de lo anterior,

$$(I-L)^{-1} = \sum_{j=0}^{\infty} L^j.$$

Por último, para probar (2.1), teniendo en cuenta una vez más que $\|L\| < 1$, tenemos que

$$\|(I-L)^{-1}\| \leqslant \sum_{j=0}^{\infty} \|L^j\| \leqslant \sum_{j=0}^{\infty} \|L\|^j = \frac{1}{1-\|L\|}.$$

□

Hay una demostración alternativa utilizando la complitud del espacio y la suma de una serie geométrica (véase [HA09]).

Observación 2.1. El teorema dice que bajo las hipótesis que hemos establecido, para cualquier $f \in X$, la ecuación

$$(I - L)u = f \tag{2.2}$$

tiene solución única $u = (I - L)^{-1} f \in X$. Además, la solución depende continuamente de la parte derecha f: siendo $(I - L)u_1 = f_1$ y $(I - L)u_2 = f_2$, de esto se sigue que

$$u_1 - u_2 = (I - L)^{-1}(f_1 - f_2),$$

y por tanto,

$$\|u_1 - u_2\| \leqslant c\|f_1 - f_2\|$$

con $c = 1/(1 - \|L\|)$.

Observación 2.2. Este teorema también nos da una forma de aproximar la solución de la ecuación (2.2). Bajo las hipótesis del teorema, tenemos

$$u = \lim_{n\to\infty} u_n$$

donde

$$u_n = \sum_{j=0}^{n} L^j f.$$

Si aproximamos la solución u por una suma parcial u_n, el error cometido puede estimarse explícitamente:

$$\|u - u_n\| = \|\sum_{j=n+1}^{\infty} L^j f\| \leqslant \sum_{j=n+1}^{\infty} \|L\|^j \|f\| = \frac{\|L\|^{n+1}}{1 - \|L\|}\|f\|.$$

Como consecuencia del Teorema 2.2 presentamos un resultado importante que nos garantiza la existencia y unicidad de solución para una ecuación lineal integral de Fredholm de segunda clase.

Corolario 2.1. *Sea la ecuación integral de Fredholm de segunda clase*

$$u(x) = f(x) + \int_a^b K(x,t)u(t)dt, \qquad x \in [a,b],$$

donde K y f son continuas y $\|K\|_\infty < \dfrac{1}{b-a}$, entonces la ecuación tiene solución única $u \in \mathcal{C}[a,b]$.

Demostración. Sea $X = \mathcal{C}[a,b]$ con la norma $\|\cdot\|_\infty$. Vamos a reescribir la ecuación de forma simbólica, organizando los términos para poder aplicar posteriormente el teorema de la serie geométrica:

$$(I - \mathcal{K})u = f,$$

donde $\mathcal{K}$ es un operador integral lineal gerenado por el núcleo $K(\cdot,\cdot)$.

Escribimos la ecuación de esta manera puesto que así se puede convertir en la forma que necesitamos para aplicar el teorema de la serie geométrica:

$$(I - L)u = f, \qquad L = \mathcal{K}.$$

Aplicando el teorema de la serie geométrica, afirmamos que si

$$\|L\| = \|\mathcal{K}\| < 1,$$

entonces $(I - L)^{-1}$ existe y

$$\|(I - L)^{-1}\| \leqslant \frac{1}{1 - \|L\|}.$$

Equivalentemente, si

$$\rho := \max_{a \leqslant x \leqslant b} \int_a^x |K(x,t)|dt < 1,$$

entonces $(I - \mathcal{K})^{-1}$ existe y

$$\|(I - \mathcal{K})^{-1}\| \leqslant \frac{1}{1 - \rho}.$$

Como $\|K\|_\infty < \dfrac{1}{b-a}$, entonces $\rho < 1$, luego para cualquier $f \in \mathcal{C}[a,b]$, la ecuación integral tiene solución única $u \in \mathcal{C}[a,b]$ y

$$\|u\|_\infty \leqslant \|(I - \mathcal{K})^{-1}\| \|f\|_\infty \leqslant \frac{\|f\|_\infty}{1 - \rho}.$$

Nótese que la condición $\rho < 1$ está garantizada en cuanto $\|K\|_\infty < 1$, siendo $K \in \mathcal{C}([a,b] \times [a,b])$. □

2.3. Generalización del teorema

En principio, a la hora de probar que la ecuación integral de Volterra de segunda clase

$$u(x) = f(x) + \int_0^x K(x,t)u(t)dt, \qquad x \in [0,B],$$

tiene solución única, y a parte de suponer $B > 0$, que el núcleo $K(x,t) \in \mathcal{C}([0,B] \times [0,B])$ y $f \in \mathcal{C}[0,B]$, si queremos aplicar el teorema de la serie geométrica, tal y como acabamos de comprobar, tendremos que suponer una condición extra sobre el núcleo:

$$\max_{x \in [0,B]} \int_0^x |K(x,t)|dt < 1,$$

o, de forma más restrictiva, $\|K\|_\infty < \dfrac{1}{B}$. Sin embargo, comprobar esta restricción puede obviarse, gracias a una generalización del teorema de la serie geométrica. Simbólicamente, escribiremos la ecuación integral como $(I - L)u = f$.

Corolario 2.2. *Sea X un espacio de Banach, $L \in \mathcal{L}(X)$. Suponemos que para algún entero $n \geqslant 1$ se*

cumple

$$\|L^n\| < 1.$$

Entonces $I - L$ es una biyección en X, su inversa es un operador lineal y continuo,

$$(I-L)^{-1} = \sum_{j=0}^{\infty} L^j,$$

y

$$\|(I-L)^{-1}\| \leqslant \frac{1}{1-\|L^n\|} \sum_{i=0}^{n-1} \|L^i\|. \tag{2.3}$$

Demostración. Gracias al teorema de la serie geométrica, sabemos que $(I - L^n)^{-1}$ existe como un operador biyectivo lineal y continuo de X en sí mismo,

$$(I-L^n)^{-1} = \sum_{j=0}^{\infty} L^{nj}$$

en $\mathcal{L}(X)$, y

$$\|(I-L^n)^{-1}\| \leqslant \frac{1}{1-\|L^n\|}.$$

De las igualdades

$$(I-L)(\sum_{i=0}^{n-1} L^i) = (\sum_{i=0}^{n-1} L^i)(I-L) = I - L^n,$$

concluimos que $(I - L)$ es una biyección,

$$(I-L)^{-1} = (\sum_{i=0}^{n-1} L^i)(I-L^n)^{-1}$$

que es continuo, gracias al Teorema de los isomorfismos de Banach, de donde se deduce la desigualdad (2.3) y la igualdad

$$(I-L)^{-1} = \sum_{j=0}^{\infty} L^j.$$

□

Vamos a apoyarnos en esta generalización para ilustrar la demostración del siguiente resultado:

Corolario 2.3. *Toda ecuación integral lineal de Volterra de segunda clase tiene solución única, cuando trabajamos en contexto continuo.*

Demostración. Sea la ecuación integral de Volterra de segunda clase:

$$u(x) = f(x) + \int_0^x K(x,t)u(t)dt, \qquad x \in [0,B],$$

donde $f \in \mathcal{C}[0,B]$, $K \in \mathcal{C}([0,B] \times [0,B])$ y $u \in \mathcal{C}[0,B]$ es la función incógnita. Definimos el

operador lineal L, que aparece dentro de la ecuación de Volterra:

$$\begin{aligned} L &: \mathcal{C}[0,B] \to \mathcal{C}[0,B] \\ Lu(x) &\mapsto \int_0^x K(x,t)u(t)dt, \qquad x \in [0,B]. \end{aligned} \tag{2.4}$$

Así, podemos escribir la ecuación como $(I-L)u = f$. Vamos a comprobar que verifica las hipótesis de la generalización del teorema de la serie geométrica, es decir, que para algún entero $n \geqslant 1$ se cumple $\|L^n\| < 1$.
En primer lugar, establecemos la desigualdad

$$n \geqslant 1 \Rightarrow \|L^n\| \leqslant \frac{\|K\|_\infty^n B^n}{n!},$$

equivalentemente,

$$n \geqslant 1, u \in \mathcal{C}[0,B] \Rightarrow \|L^n u\|_\infty \leqslant \frac{\|K\|_\infty^n B^n}{n!}\|u\|_\infty. \tag{2.5}$$

En efecto, para $n = 1$, si $0 \leqslant x \leqslant B$, entonces

$$\begin{aligned} |Lu(x)| &\leqslant \int_0^x |K(x,t)u(t)|dt \\ &\leqslant \int_0^x \|K\|_\infty \|u\|_\infty dt \\ &= \|K\|_\infty \|u\|_\infty \int_0^x dt \\ &= \|K\|_\infty \|u\|_\infty x. \end{aligned}$$

Así pues,

$$|Lu(x)| \leqslant \|K\|_\infty \|u\|_\infty x. \tag{2.6}$$

Para $n = 2$, si $0 \leqslant x \leqslant B$, tenemos

$$\begin{aligned} |L^2 u(x)| &\leqslant \int_0^x |K(x,t)Lu(t)|tdt \\ &\leqslant \int_0^x \|K\|_\infty \|K\|_\infty \|u\|_\infty t dt \\ &= \|K\|_\infty^2 \|u\|_\infty \int_0^x t dt \\ &= \|K\|_\infty^2 \|u\|_\infty \frac{x^2}{2}, \end{aligned}$$

donde en la segunda desigualdad hemos utilizado (2.6). Inductivamente deducimos

$$\left.\begin{matrix} n \geqslant 1, \qquad u \in \mathcal{C}[0,B] \\ x \in [0,B] \end{matrix}\right\} \Rightarrow |L^n u(x)| \leqslant \frac{\|K\|_\infty^n x^n}{n!}\|u\|_\infty,$$

de donde se tiene (2.5) y, por tanto,

$$\|L^n\| \leqslant \frac{\|K\|_\infty^n B^n}{n!}. \tag{2.7}$$

Debido a que la sucesión $\{\frac{\|K\|_\infty^n B^n}{n!}\}$ converge a 0 (la serie $\sum_{n=0}^{\infty} \frac{\|K\|_\infty^n B^n}{n!}$ es convergente, $\sum_{n=0}^{\infty} \frac{\|K\|_\infty^n B^n}{n!} = e^{\|K\|_\infty B}$), la desigualdad (2.7) da $\|L^n\| \to 0$ y, por tanto, a partir de un n en adelante, $\|L^n\| < 1$. Luego podemos aplicar la generalización del teorema de la serie geométrica (Corolario 2.2) y tenemos que existe el operador inverso $(I-L)^{-1}$ en $\mathcal{C}([0,B])$, y su inverso se puede expresar como la suma de una serie geométrica convergente, por tanto, la solución única u viene dada por:

$$u = (I-L)^{-1} f.$$

□

Como $u = \sum_{j=0}^{\infty} L^j f$, una aproximación de dicha solución viene dada por una suma parcial $u_n = \sum_{j=0}^{n} L^j f$ y gracias a (2.7) tenemos

$$\|u - u_n\| \leqslant \sum_{j=n+1}^{\infty} \frac{\|K\|_\infty^j B^j}{j!}.$$

2.4. Un resultado de perturbación

Vamos a presentar otra aplicación del teorema de la serie geométrica, distinta a lo hecho con las ecuaciones integrales previamente. La *perturbación* es una estrategia en matemática aplicada que se enfoca en el estudio de una ecuación al relacionarla con otra ecuación "afín", de la cual sabemos que existe un resultado que nos da una solución, esto nos ayuda a encontrar soluciones para problemas más complicados utilizando otros más simples. El siguiente teorema representa una de las herramientas de uso más frecuente en este contexto.

Teorema 2.3. *Sean X e Y espacios normados, siendo al menos uno de ellos completo. Suponemos que $L \in \mathcal{L}(X,Y)$ posee un inverso lineal y continuo $L^{-1} : Y \to X$. Además, $M \in \mathcal{L}(X,Y)$ satisface*

$$\|M - L\| < \frac{1}{\|L^{-1}\|}. \tag{2.8}$$

Entonces $M : X \to Y$ es una biyección, $M^{-1} \in \mathcal{L}(Y,X)$ y

$$\|M^{-1}\| \leqslant \frac{\|L^{-1}\|}{1 - \|L^{-1}\|\|L-M\|}. \tag{2.9}$$

Además,

$$\|L^{-1} - M^{-1}\| \leqslant \frac{\|L^{-1}\|^2 \|L-M\|}{1 - \|L^{-1}\|\|L-M\|}. \tag{2.10}$$

Para soluciones de las ecuaciones $Lx_1 = y$ y $Mx_2 = y$, tenemos la acotación

$$\|x_1 - x_2\| \leqslant \|M^{-1}\|\|(L-M)x_1\|. \tag{2.11}$$

Demostración. Si Y es completo, podemos escribir

$$M = [I - (L - M)L^{-1}]L;$$

mientras que si X es completo, escribiremos

$$M = L[I - L^{-1}(L - M)].$$

Vamos a hacer la demostración en el caso de que Y sea completo. El operador $(L - M)L^{-1} \in \mathcal{L}(Y)$ satisface

$$\|(L - M)L^{-1}\| \leqslant \|L - M\|\|L^{-1}\| < 1.$$

Luego por el teorema de la serie geométrica, $[I - (L - M)L^{-1}]^{-1}$ existe, es lineal y continuo y además,

$$\|[I - (L - M)L^{-1}]^{-1}\| \leqslant \frac{1}{1 - \|(L - M)L^{-1}\|} \leqslant \frac{1}{1 - \|L^{-1}\|\|L - M\|}.$$

Entonces M^{-1} existe y es lineal y continuo, con

$$M^{-1} = L^{-1}[I - (L - M)L^{-1}]^{-1}$$

y

$$\|M^{-1}\| \leqslant \|L^{-1}\|\|[I - (L - M)L^{-1}]^{-1}\| \leqslant \frac{\|L^{-1}\|}{1 - \|L^{-1}\|\|L - M\|}.$$

Para probar (2.10), escribimos

$$L^{-1} - M^{-1} = M^{-1}(M - L)L^{-1},$$

tomamos normas y usamos (2.9).

Para (2.11),

$$x_1 - x_2 = (L^{-1} - M^{-1})y = M^{-1}(M - L)L^{-1}y = M^{-1}(M - L)x_1$$

y aplicamos normas. □

Podemos resumir el teorema anterior en una frase: *Un operador cercano a otro operador biyectivo, lineal y continuo, también será biyectivo, lineal y continuo.* Según se ilustra en muchos textos, como [HA09], la aplicación de este resultado es útil para una amplia gama de resultados de existencia de soluciones para ecuaciones integrales y diferenciales.

Ejemplo 2.1. Vamos a examinar si la ecuación integral

$$u(x) - \frac{1}{\lambda}\int_0^1 \frac{\cos(xt)}{2}u(t)dt = \frac{f(x)}{\lambda}, \qquad 0 \leqslant x \leqslant 1 \tag{2.12}$$

tiene solución, con $\lambda \neq 0$. De la discusión del corolario (2.1), si

$$1 > \|K\| = \int_0^1 \frac{\cos(t)}{2}dt = \sin(1)/2 \approx 0.4207, \tag{2.13}$$

entonces para cada $f \in \mathcal{C}[0,1]$, la ecuación admite una solución única $u \in \mathcal{C}[0,1]$.

Para obtener más valores de λ para los cuales nuestra ecuación tiene solución única, aplicamos el teorema de perturbación. Ya que $\cos(xt) \approx xt$ para valores pequeños de $|xt|$, comparamos la ecuación con

$$v(x) - \frac{1}{\lambda}\int_0^1 \frac{xt}{2} v(t)dt = \frac{f(x)}{\lambda}, \qquad 0 \leqslant x \leqslant 1. \tag{2.14}$$

Siguiendo la notación el teorema de perturbación, la ecuación (2.12) sería $Mu = f$, y (2.14) sería $Lv = f$. El espacio de Banach es $X = \mathcal{C}[0,1]$ con la norma $\|\cdot\|_\infty$, y $L, M \in \mathcal{L}(X)$.

Podemos resolver la ecuación integral (2.14) de forma explícita. Suponiendo $\lambda \neq 0$, tenemos que cada solución v toma la forma

$$v(x) = \frac{1}{\lambda}[f(x) + cx]$$

para alguna constante c. Sustituyéndolo en la ecuación nos lleva a una fórmula para c, y entonces

$$v(x) = \frac{1}{\lambda}[f(x) + \frac{1}{\lambda - 1/3}\int_0^1 \frac{xt}{2} f(t)dt], \qquad \lambda \neq 0, \frac{1}{3}.$$

Esta relación define $L^{-1}f$ para toda $f \in \mathcal{C}[0,1]$.

Para usar el teorema de perturbación, necesitamos medir algunas cantidades. Se puede calcular que

$$\|L^{-1}\| \leqslant \frac{1}{|\lambda|}(1 + \frac{1}{2|\lambda - 1/3|})$$

y

$$\|L - M\| = \int_0^1 (t - \frac{\cos t}{2})dt = \frac{1}{2} - \sin(1)/2 \approx 0.0793.$$

La condición (2.8) viene de

$$\frac{1}{|\lambda|}(1 + \frac{1}{2|\lambda - 1/3|}) < \frac{1}{1/2 - \sin(1)/2}.$$

Si λ es un número real, entonces hay tres casos a considerar: $\lambda > 1/3$, $0 < \lambda < 1/3$, y $\lambda < 0$. Para el caso $\lambda < 0$, la desigualdad es cierta si y sólo si $\lambda < \lambda_0 \approx -0.1596$.

Como consecuencia del teorema de perturbación, tenemos que si $\lambda < \lambda_0$, entonces nuestra ecuación integral tiene solución única para cualquier $f \in \mathcal{C}[0,1]$. Esto es una mejora significativa en comparación con el punto de partida que teníamos (2.13).

2.5. Existencia y unicidad de solución para sistemas de ecuaciones integrales lineales de Volterra de segunda clase

Avanzamos ahora un poco más en el estudio de las ecuaciones integrales de Volterra, ocupándonos de un sistema de ecuaciones integrales lineales de Volterra de segunda clase, principalmente porque aparece en el modelo temperatura interna de un edificio que veremos más adelante, trabajaremos en un contexto continuo que, vectorialmente, adopta la expresión

$$\mathbf{u}(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}(t)dt, \qquad x \in [0, B],$$

donde $\mathbf{u}$ es el vector formado por todas las funciones continuas que queremos determinar, $\mathbf{f}$ es el vector formado por las funciones continuas independientes, y $\mathbf{K}$ es el vector formado por los núcleos continuos de cada una de las ecuaciones que forman el sistema:

$$\mathbf{u}(x) = \begin{pmatrix} u_1(x) \\ u_2(x) \\ \vdots \\ u_n(x) \end{pmatrix}, \qquad \mathbf{f}(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{pmatrix}, \qquad \mathbf{K}(x,t) = \begin{pmatrix} K_1(x,t) \\ K_2(x,t) \\ \vdots \\ K_n(x,t) \end{pmatrix}.$$

Cuando hablamos de hacer una integral compuesta por vectores, la haremos componente a componente, es decir,

$$\int_0^t \mathbf{y}(s)ds = \begin{pmatrix} \int_0^t y_1(s)ds \\ \int_0^t y_2(s)ds \\ \vdots \\ \int_0^t y_n(s)ds \end{pmatrix},$$

donde $\mathbf{y}(s) = (y_1(s), y_2(s), ..., y_n(s)) \in \mathcal{C}([0,B], \mathbb{R}^n)$.

Corolario 2.4. *Todo sistema de ecuaciones integrales lineales de Volterra de segunda clase en ambiente continuo tiene solución única.*

Demostración. Partimos de la ecuación de Volterra:

$$\mathbf{u}(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}(t)dt, \qquad x \in [0,B].$$

Definimos el operador lineal L, que aparece dentro de la ecuación de Volterra:

$$\begin{aligned} \mathbf{L} : \mathcal{C}([0,B], \mathbb{R}^n) &\to \mathcal{C}([0,B], \mathbb{R}^n) \\ \mathbf{L}\mathbf{u}(x) &\mapsto \int_0^x \mathbf{K}(x,t)\mathbf{u}(t)dt, \qquad x \in [0,B]. \end{aligned}$$

Ahora podemos escribir la ecuación como $(\mathbf{I} - \mathbf{L})\mathbf{u} = \mathbf{f}$. Vamos a comprobar que para algún entero $n \geqslant 1$ se cumple $\|\mathbf{L}^n\| < 1$. Es decir, verifica las hipótesis de la generalización del teorema de la serie geométrica. Establecemos la desigualdad

$$n \geqslant 1 \Rightarrow \|\mathbf{L}^n\|_\infty \leqslant \frac{\|\mathbf{K}\|_\infty^n B^n}{n!},$$

equivalentemente,

$$n \geqslant 1, \mathbf{u} \in \mathcal{C}([0,B], \mathbb{R}^n) \Rightarrow \|\mathbf{L}^n\mathbf{u}\|_\infty \leqslant \frac{\|\mathbf{K}\|_\infty^n B^n}{n!}\|\mathbf{u}\|_\infty.$$

Los cálculos para $n = 1$ y $n = 2$ son análogos al caso escalar cambiando la notación.
Como la sucesión $\{\frac{\|\mathbf{K}\|_\infty^n B^n}{n!}\}$ converge a 0 (la serie $\sum_{n=0}^{\infty} \frac{\|\mathbf{K}\|_\infty^n B^n}{n!}$ es convergente, $\sum_{n=0}^{\infty} \frac{\|\mathbf{K}\|_\infty^n B^n}{n!} = e^{\|\mathbf{K}\|_\infty B}$), obtenemos $\|\mathbf{L}^n\| \to 0$ y, por tanto, a partir de un n en adelante, $\|\mathbf{L}^n\| < 1$. Ahora podemos aplicar la generalización del teorema de la serie geométrica que vimos anterior-

mente, aún estando en caso vectorial, el resultado es análogo y por tanto sabemos que existe el operador inverso $(\mathbf{I}-\mathbf{L})^{-1}$ en $\mathcal{C}([0,B],\mathbb{R}^n)$, y además, podemos expresar su inverso como la suma de una serie geométrica convergente, por tanto, la solución única $\mathbf{u}$ viene dada por:

$$\mathbf{u} = (\mathbf{I}-\mathbf{L})^{-1}\mathbf{f}.$$

□

Las aclaraciones hechas en caso escalar relativas a la aproximación $\sum_{j=0}^{n} \mathbf{L}^j\mathbf{f}$ de la solución $\sum_{j=0}^{\infty} \mathbf{L}^j\mathbf{f}$ son igualmente válidas aquí.

3. Métodos de resolución de ecuaciones integrales de Volterra de segunda clase

> Volterra empezó a trabajar en las ecuaciones integrales en 1884, pero el nombre de *ecuación integral* se lo dio Bois-Reymond en 1888. Sin embargo, el término *ecuación integral de Volterra* se utilizó por primera vez en 1908 por el matemático rumano Traian Lalesco.

3.1. Introducción

El diseño de métodos numéricos para ecuaciones integrales en general, y en particular de Volterra, es un campo de investigación vigente actualmente. Existen una gran variedad de métodos numéricos y analíticos, tales como el método de aproximaciones sucesivas, la transformada de Laplace, colocación con splines, Runge-Kutta, y otros muchos que han sido utilizados para manejar las ecuaciones integrales de Volterra. Todos estos métodos nos proporcionan como solución una función exacta o aproximada, sin embargo, al final de la sección veremos, además, los métodos de cuadratura, cuya particularidad es que en vez de dar como resultado una función, proporcionan valores cercanos a la solución en una cantidad finita de puntos.

A parte de estudiar algunos de estos métodos tradicionales, veremos otros métodos algo más recientes tales como:

- Método de descomposición de Adomian (ADM)
- Método de descomposición modificado (mADM)
- Método de iteración variacional (VIM)

Nos vamos a centrar principalmente en cómo se aplican estos métodos con un objetivo principal, encontrar una solución $u(x)$ para la ecuación integral de Volterra de segunda clase. Daremos una visión general de esta amplia familia de métodos de resolución, sin entrar en detalles como la convergencia o el error.

Para diferenciar mejor en qué se basan cada uno de los métodos, vamos a distinguir entre los métodos que utilizan series, los métodos iterativos, y otros métodos especiales que no se engloban en los anteriores.

Para recopilar información sobre los métodos, hemos tomado como referencia básica [Waz11].

3.2. Métodos basados en series

3.2.1. Método de descomposición de Adomian (ADM)

Fue desarrollado por George Adomian en [AR92] y [Ado94] y está muy bien abordado en muchas referencias. Se ha investigado mucho sobre este método para poder aplicarlo a una

amplia clase de ecuaciones diferenciales ordinarias, en derivadas parciales, y también en ecuaciones integrales (véase [HA09]).

Consiste en expresar una función $u(x)$ como una serie de descomposición definida por

$$u(x) = \sum_{n=0}^{\infty} u_n(x), \tag{3.1}$$

convergente en algún sentido, donde las componentes $u_n(x), n \geqslant 0$, tienen que ser determinadas de una forma recursiva. El método se ocupa de encontrar las componentes $u_0, u_1, u_2, ...$, individualmente. Como ya veremos más adelante, se pueden hallar estas componentes de una forma sencilla a través de una relación de recurrencia que normalmente implica integrales simples que pueden ser fácilmente evaluadas.

Para establecer la relación de recurrencia, sustituimos (3.1) en la ecuación integral de Volterra (1.2) para obtener

$$\sum_{n=0}^{\infty} u_n(x) = f(x) + \int_0^x K(x,t)(\sum_{n=0}^{\infty} u_n(t))dt,$$

o equivalentemente

$$u_0(x) + u_1(x) + u_2(x) + \cdots = f(x) + \int_0^x K(x,t)[u_0(t) + u_1(t) + \cdots]dt.$$

La primera componente $u_0(x)$ se identifica con todos los términos que no están incluidos dentro de la integral. Por lo tanto, las componentes $u_j(x), j \geqslant 1$, de la función desconocida $u(x)$, están completamente determinadas a través de la siguiente relación de recurrencia:

$$\begin{aligned} u_0(x) &= f(x), \\ u_{n+1}(x) &= \int_0^x K(x,t)u_n(t)dt, \qquad n \geqslant 0, \end{aligned}$$

que es equivalente a

$$\begin{aligned} u_0(x) &= f(x), & u_1(x) &= \int_0^x K(x,t)u_0(t)dt, \\ u_2(x) &= \int_0^x K(x,t)u_1(t)dt, & u_3(x) &= \int_0^x K(x,t)u_2(t)dt, \end{aligned} \tag{3.2}$$

y análogamente para las demás componentes. Como podemos ver en (3.2), las componentes $u_0(x), u_1(x), u_2(x), u_3(x), ..., u_n(x)$ están completamente determinadas. Como resultado, la solución $u(x)$ de la ecuación integral de Volterra (1.2) en forma de serie se obtiene fácilmente utilizando (3.1).

Observación 3.1. Hemos visto cómo el método de descomposición ha convertido una ecuación integral en una determinación de componentes calculables. Muchos investigadores formalizaron que si existe una solución exacta para el problema, entonces la serie obtenida converge muy rápidamente a esa solución (véase [Waz11]). De hecho, la serie obtenida coincide con la de la serie geométrica.

Sin embargo, para problemas concretos donde no se puede obtener una solución, generalmente se utiliza un número truncado de términos con fines numéricos. Podemos observar

que cuantos más componentes usemos, mayor precisión obtendremos, ya que la sucesión formada por los componentes converge a la solución.

Ejemplo 3.1. Resolveremos la siguiente ecuación integral de Volterra utilizando el ADM:

$$u(x) = 1 - \int_0^x u(t)dt.$$

En este caso, $f(x) = 1, \lambda = -1, K(x,t) = 1$. Se asume que la solución $u(x)$ tiene una forma en serie como la dada en (3.1). Sustituyendo la serie en ambos lados de nuestra ecuación, obtenemos:

$$\sum_{n=0}^{\infty} u_n(x) = 1 - \int_0^x \sum_{n=0}^{\infty} u_n(t)dt.$$

La primera componente corresponde con todos los términos que no están incluidos dentro de la integral, por tanto, obtenemos la siguiente recurrencia:

$$\begin{aligned} u_0(x) &= 1, \\ u_{k+1}(x) &= -\int_0^x u_k(t)dt, \qquad k \geqslant 0, \end{aligned}$$

por tanto, tenemos:

$$\begin{aligned} u_0(x) &= 1, \\ u_1(x) &= -\int_0^x u_0(t)dt = -\int_0^x 1dt = -x, \\ u_2(x) &= -\int_0^x u_1(t)dt = -\int_0^x (-t)dt = \frac{1}{2!}x^2, \\ u_3(x) &= -\int_0^x u_2(t)dt = -\int_0^x \frac{1}{2!}t^2dt = \frac{1}{3!}x^3. \end{aligned}$$

Así, obtenemos la solución en serie:

$$u(x) = 1 - x + \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \cdots,$$

que converge a la solución:

$$u(x) = e^{-x}.$$

3.2.2. Método de descomposición de Adomian modificado (mADM)

Wazwaz presenta una modificación importante del ADM en sus libros [Waz09] y [Waz99]. Este método facilitará el proceso de cálculo y acelerará la convergencia de la solución en series. Se aplicará, siempre que se pueda, a cualquier ecuación integral y diferencial de cualquier orden.

Observación 3.2. Este método se basa principalmente en dividir la función $f(x)$ en dos partes, por tanto no se puede usar si la función $f(x)$ está formada por un sólo término.

El método de descomposición modificado introduce una pequeña variación a la relación de recurrencia que vimos en el ADM, y esto nos llevará a la determinación de las componentes de $u(x)$ de una forma más fácil y rápida.

En muchos casos, la función $f(x)$ se puede escribir como una suma de dos funciones parciales, llamadas $f_1(x)$ y $f_2(x)$:

$$f(x) = f_1(x) + f_2(x).$$

Gracias a esto, se introducirá un cambio importante a la hora de formar la relación de recurrencia en el método de Adomian. Para minimizar el tamaño de los cálculos, identificamos la primera componente $u_0(x)$ como una de las partes de $f(x)$, que será $f_1(x)$ o $f_2(x)$. La otra parte se puede añadir a la componente $u_1(x)$ junto a los demás términos. En resumen, obtenemos la siguiente relación de recurrencia:

$$\begin{aligned} u_0(x) &= f_1(x), \\ u_1(x) &= f_2(x) + \int_0^x K(x,t)u_0(t)dt, \\ u_{k+1}(x) &= \int_0^x K(x,t)u_k(t)dt, \qquad k \geqslant 1. \end{aligned}$$

Observación 3.3. Esto muestra que la diferencia entre la relación de recurrencia estándar y la modificada se basa únicamente en la formación de las dos primeras componentes $u_0(x)$ y $u_1(x)$. Las demás componentes se mantienen igual.

Aunque esta variación en las dos primeras componentes es pequeña, WazWaz afirma en su libro [Waz11] que juega un papel muy importante en acelerar la convergencia de la solución y en minimizar la cantidad de trabajo computacional. Además, varios trabajos de investigación han confirmado que reducir el número de componentes en $f_1(x)$ afecta a todas las componentes, no sólo a $u_1(x)$.

A continuación vamos a ver dos observaciones importantes a cerca de este método:

Observación 3.4. Si elegimos correctamente las funciones $f_1(x)$ y $f_2(x)$, podremos obtener la solución exacta $u(x)$ utilizando muy pocas iteraciones, incluso algunas veces evaluando sólo dos componentes. De hecho, el éxito de esta modificación depende de hacer una buena elección de las funciones $f_1(x)$ y $f_2(x)$, y la forma de obtenerlas adecuadamente es a través de prueba y error, ya que no se ha encontrado todavía una regla para facilitar esta elección.

Observación 3.5. Si $f(x)$ está formada sólo por un término, el método recae en el método de descomposición de Adomian.

Podemos utilizar este método para las ecuaciones integrales tanto de Volterra como de Fredholm, lineales en nuestro caso. Vamos a ver un ejemplo utilizando este método, y compararemos la solución con la del método de Adomian para 5 iteraciones y así podremos apreciar la diferencia en cuanto a la rapidez de convergencia:

Ejemplo 3.2. Resolver la ecuación integral de Volterra:

$$u(x) = 1 - x - \frac{1}{2}x^2 - \int_0^x (t-x)u(t)dt.$$

La función $f(x)$ está compuesta por 3 términos, que están fuera de la integral, y vamos a dividirlos en dos partes de la siguiente forma:

$$\begin{aligned} f_1(x) &= 1 - x, \\ f_2(x) &= -\frac{1}{2}x^2. \end{aligned}$$

Y ahora utilizamos la fórmula de recurrencia modificada, obteniendo:

$$\begin{aligned} u_0(x) &= 1 - x, \\ u_1(x) &= -\frac{1}{2}x^2 - \int_0^x (t-x)u_0(t)dt = 0, \\ u_{k+1}(x) &= -\int_0^x (t-x)u_k(t)dt, \qquad k \geqslant 1. \end{aligned}$$

En la quinta iteración obtenemos:

$$u_5(x) = -\frac{x^9}{120960}.$$

Hemos obtenido la solución exacta para esta ecuación gracias a *Maxima*:

$$u(x) = 1 - \sinh(x).$$

Ahora vamos a mostrar gráficamente en la Figura 3.1 la solución que nos da el método de Adomian y el modificado para 5 iteraciones comparándolos con la solución exacta:

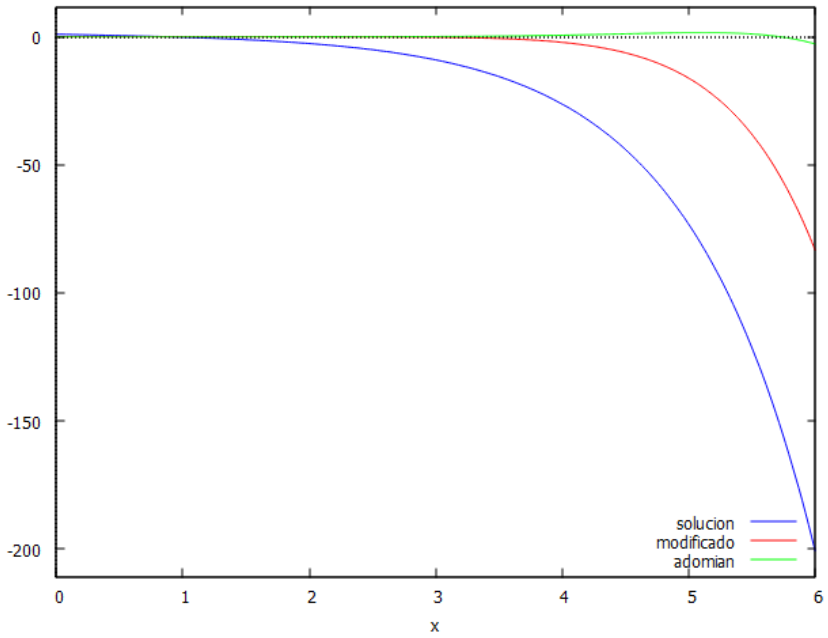

Figura 3.1.: Comparación de la solución para 5 iteraciones con el método de Adomian y el método modificado

Observación 3.6. Podemos ver como efectivamente la convergencia en el método modificado es mucho más rápida que en el método de Adomian ya que con sólo 5 iteraciones ya se aproxima mucho mejor a la solución.

3.2.3. Método de la solución en series

Definición 3.1. Una función real $u(x)$ se llama *analítica* si existe derivada de cualquier orden de forma que la serie de Taylor en cualquier punto b de su dominio

$$u(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(b)}{k!}(x-b)^k,$$

converge a $f(x)$ en un entorno de b.

Por simplicidad, escribiremos la forma genérica de la serie de Taylor en $x = 0$ como

$$u(x) = \sum_{n=0}^{\infty} a_n x^n. \tag{3.3}$$

Observación 3.7. Si f es analítica en todo punto de un intervalo I, entonces, la propia definición nos garantiza que $f \in \mathcal{C}^\infty(I)$. Sin embargo, el recíproco no es cierto, como prueba el siguiente ejemplo bien conocido (véase en [Rev15]):

Ejemplo 3.3. Sea la función

$$f(x) = \begin{cases} e^{-1/x^2} & x > 0 \\ 0 & x \leqslant 0 \end{cases}$$

Esta función es $\mathcal{C}^\infty(\mathbb{R})$, ya que todas sus derivadas existen y son continuas en cualquier punto. Sin embargo, no es analítica en $x = 0$. La serie de Taylor alrededor de $x = 0$ se reduce a una serie de potencias de ceros. Por lo tanto, no se puede representar como una serie de potencias convergente en un entorno de $x = 0$.

Vamos a presentar un método muy útil, que se basa principalmente en la serie de Taylor para funciones analíticas, y resolverá ecuaciones integrales de Volterra. Podemos apreciar que guarda cierta similitud con Adomian, ya que de igual forma expresamos u como la suma de cierta serie.

Asumimos que la solución $u(x)$ de la ecuación integral de Volterra

$$u(x) = f(x) + \int_0^x K(x,t)u(t)dt,$$

es analítica, es decir, posee una solución en forma de serie de Taylor, donde los coeficientes a_n se determinarán de forma recursiva. Sustituimos (3.3) en ambos miembros y obtenemos

$$\sum_{n=0}^{\infty} a_n x^n = T(f(x)) + \int_0^x K(x,t)(\sum_{n=0}^{\infty} a_n t^n)dt, \tag{3.4}$$

o por simplicidad usamos

$$a_o + a_1 x + a_2 x^2 + \cdots = T(f(x)) + \int_0^x K(x,t)(a_0 + a_1 t + a_2 t^2 + \cdots)dt,$$

donde $T(f(x))$ es la serie de Taylor para $f(x)$. La ecuación integral se convertirá en una integral tradicional donde en vez de integrar la función desconocida $u(x)$, se integrarán términos de la forma $t^n, n \geqslant 0$. Al estar en búsqueda de una solución en series, si la función $f(x)$ incluye funciones elementales, deberemos incluir sus desarrollos en forma de Taylor.

Primero integramos la parte derecha de la integral en (3.4) y tomamos los coeficientes de las potencias de x. Después igualamos los coeficientes de ambos lados para obtener una relación de recurrencia con $a_j, j \geqslant 0$. Resolver la recurrencia nos llevará a determinar completamente los coeficientes $a_j, j \geqslant 0$.

Ahora la solución en series se sigue inmediatamente de sustituir estos coeficientes. La solución exacta se obtendrá si existe, y si no, la serie obtenida se podrá utilizar con objetivos numéricos. En este caso, a más términos evaluemos, mayor precisión obtendremos.

Veamos un ejemplo para dejar claro este método:

Ejemplo 3.4. Vamos a resolver la ecuación integral de Volterra

$$u(x) = 1 - x\sin x + \int_0^x tu(t)dt.$$

Sustituyendo la serie en ambos lados y utilizando la expansión en serie de Taylor para el seno obtenemos:

$$a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \cdots = 1 - x(x - \frac{x^3}{3!} + \cdots) + \int_0^x t(a_0 + a_1t + a_2t^2 + \cdots)dt.$$

Integrando el lado derecho y quedándonos con los coeficientes encontramos

$$a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \cdots = 1 + (\frac{1}{2}a_0 - 1)x^2 + \frac{1}{3}a_1x^3 + (\frac{1}{6} + \frac{1}{4}a_2)x^4 + \cdots$$

Igualando los coeficientes de las potencias de x,

$$\begin{aligned} a_0 &= 1, & a_1 &= 0, \\ a_2 &= \frac{1}{2}a_0 - 1, & a_3 &= \frac{1}{3}a_1 = 0, \\ a_4 &= \frac{1}{6} + \frac{1}{4}a_2 = \frac{1}{4!}, \end{aligned}$$

y para $n \geqslant 0$,

$$a_{2n+1} = 0, \qquad a_{2n} = \frac{(-1)^n}{(2n)!}, \qquad n \geqslant 0.$$

La solución en forma de serie viene dada por

$$u(x) = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \cdots$$

que nos da la solución exacta

$$u(x) = \cos x.$$

En la Figura 3.2 mostramos la comparación de la suma parcial $v(x) = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \frac{1}{8!}x^8$ con la solución exacta $u(x)$, se aprecia que la aproximación es bastante buena.

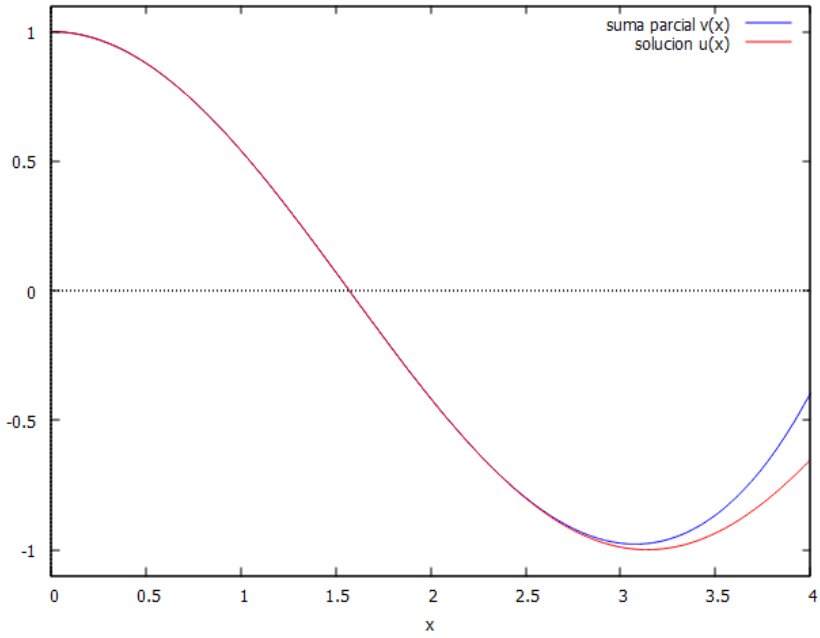

Figura 3.2.: Comparación de la solución exacta con la suma parcial $v(x)$

3.3. Métodos iterativos

3.3.1. Método de iteración variacional (VIM)

Este método ha sido desarrollado por Ji-Huan He en sus libros [Heo6] y [Heoo], y ha demostrado ser eficaz y seguro para estudios numéricos y analíticos. El método proporciona aproximaciones sucesivas que convergen rápidamente a la solución exacta en forma de serie, siempre que exista. Sin embargo, en el caso de que no se pueda obtener la solución exacta, la serie resultante se puede utilizar para fines numéricos. Asimismo, una de las ventajas que tiene este método es que puede abordar problemas tanto lineales como no lineales de la misma forma sin necesidad de añadir más restricciones. Lo vamos a incluir para ilustrar un método que es esencialmente no lineal, aunque no sea el objeto de estudio de esta memoria. Vamos a presentar los pasos principales del método:

Consideramos la ecuación diferencial:

$$Lu + Nu = g(t), \tag{3.5}$$

donde L y N son operadores lineales y no lineales respectivamente , y $g(t)$ es el término no homogéneo.

El método de iteración variacional presenta un funcional de corrección para la ecuación (3.5) de la forma:

$$u_{n+1}(x) = u_n(x) + \int_0^x \lambda(\xi)(Lu_n(\xi) + N\tilde{u}_n(\xi) - g(\xi))d\xi, \tag{3.6}$$

donde λ es un multiplicador de Lagrange general.

Observación 3.8. Nótese que en este método, λ puede ser una constante o una función, y $\tilde{u}_n$ es un valor restringido, por tanto se comporta como una constante, luego $\tilde{u}_n' = 0$.

Para un uso completo de este método, deberíamos seguir dos pasos:

1. Determinar el multiplicador de Lagrange $\lambda(\xi)$ que será identificado de forma óptima.

2. Una vez determinado λ, sustituimos el resultado en (3.6) donde se deberían omitir las restricciones.

Derivando en (3.6) con respecto a la variable independiente u_n, obtenemos

$$\frac{du_{n+1}}{du_n} = 1 + \frac{d}{du_n}(\int_0^x \lambda(\xi)(Lu_n(\xi) + N\tilde{u}_n(\xi) - g(\xi))d\xi),$$

o equivalentemente si multiplicamos por du_n:

$$du_{n+1} = du_n + (\int_0^x \lambda(\xi)(Lu_n(\xi))d\xi). \tag{3.7}$$

Para determinar el multiplicador de Lagrange $\lambda(\xi)$ normalmente se utiliza la integración por partes. Por ejemplo, si tenemos $Lu_n(\xi) = u_n'(\xi)$ en (3.7), entonces se convierte en

$$du_{n+1} = du_n + (\int_0^x \lambda(\xi)(u_n'(\xi))d\xi).$$

Integrando por partes obtenemos

$$du_{n+1} = du_n + \lambda(\xi)u_n(\xi) - \int_0^x \lambda'(\xi)du_n(\xi)d\xi.$$

o equivalentemente

$$du_{n+1} = du_n(\xi)(1 + \lambda|_{\xi=x}) - \int_0^x \lambda' du_n d\xi.$$

La condición final de u_{n+1} requiere que $du_{n+1} = 0$. Esto significa que la parte izquierda vale cero, y por tanto la parte derecha también debería valer cero, lo que nos lleva a las condiciones:

$$1 + \lambda|_{\xi=x} = 0, \qquad \lambda'|_{\xi=x} = 0.$$

Lo que al final nos da

$$\lambda = -1.$$

Una vez hemos determinado el multiplicador de Lagrange $\lambda(\xi)$, las sucesivas aproximaciones $u_{n+1}, n \geqslant 0$, de la solución $u(x)$ se obtendrán fácilmente al usar la función selectiva $u_0(x)$. Sin embargo, para una rápida convergencia, la función $u_0(x)$ se debe seleccionar utilizando las primeras condiciones como siguen:

$$\begin{aligned}
&u_0(x) = u(0), && \text{para la primera derivada } u_n' \\
&u_0(x) = u(0) + xu'(0), && \text{para la segunda derivada } u_n'' \\
&u_0(x) = u(0) + xu'(0) + \frac{1}{2!}x^2u''(0), && \text{para la tercera derivada } u_n'''
\end{aligned}$$

y así sucesivamente. Por tanto, tenemos la solución

$$u(x) = \lim_{n\to\infty} u_n(x).$$

Es decir, el funcional de corrección nos dará varias aproximaciones, y por tanto, la solución exacta se obtiene como el límite de todas estas sucesivas aproximaciones.

La determinación del multiplicador de Lagrange juega un papel muy importante para llegar a la solución del problema. A continuación, mostramos un esquema correspondiente al multiplicador de Lagrange y su funcional de corrección para un caso general de orden n:

$$u^{(n)} + f(u(\xi), u'(\xi), ..., u^{(n)}(\xi)) = 0, \lambda = (-1)^n \frac{1}{(n-1)!}(\xi - x)^{(n-1)},$$
$$u_{n+1} = u_n + (-1)^n \int_0^x \frac{1}{(n-1)!}(\xi - x)^{(n-1)}[u_n''' + f(u_n, ..., u_n^{(n)})]d\xi,$$

para todo $n \geqslant 1$.

Para utilizar el método de iteración variacional y resolver ecuaciones integrales de Volterra, es necesario convertir la ecuación integral a un problema de valores iniciales equivalente o a una ecuación integro-diferencial.

Vamos a examinar el problema de valores iniciales obtenido utilizando el método de iteración variacional como veremos en el siguiente ejemplo:

Ejemplo 3.5. Resolveremos la siguiente ecuación integral de Volterra usando el método de iteración variacional:

$$u(x) = 1 + \int_0^x u(t)dt.$$

Usando el Teorema Fundamental del Cálculo para derivar ambos lados de la ecuación, obtenemos

$$u'(x) - u(x) = 0. \tag{3.8}$$

Sustituyendo $x = 0$, tenemos la condición inicial $u(0) = 1$. Ahora, utilizando el método de iteración variacional, el funcional de corrección para la ecuación (3.8) es

$$u_{n+1}(x) = u_n(x) + \int_0^x \lambda(\xi)(u_n'(\xi) - \tilde{u}_n(\xi))d\xi. \tag{3.9}$$

Usando la fórmula del esquema visto anteriormente para el caso $n = 1$ llegamos a que

$$\lambda = -1.$$

Sustituyendo el valor del multiplicador de Lagrange $\lambda = -1$ en el funcional nos da la fórmula iterativa:

$$u_{n+1}(x) = u_n(x) - \int_0^x (u_n'(\xi) - u_n(\xi))d\xi.$$

Como dijimos anteriormente, podemos usar la condición inicial $u_0(x) = u(0) = 1$. Usando

esta selección en (3.9) obtenemos las siguientes aproximaciones sucesivas:

$$
\begin{aligned}
u_0(x) &= 1, \\
u_1(x) &= 1 - \int_0^x (u_0'(\xi) - u_0(\xi))d\xi = 1 + x, \\
u_2(x) &= 1 + x - \int_0^x (u_1'(\xi) - u_1(\xi))d\xi = 1 + x + \frac{1}{2!}x^2, \\
u_3(x) &= 1 + x + \frac{1}{2!}x^2 - \int_0^x (u_2'(\xi) - u_2(\xi))d\xi = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3,
\end{aligned}
$$

y así sucesivamente. El VIM admite el uso de

$$
\begin{aligned}
u(x) &= \lim_{n\to\infty} u_n(x), \\
&= \lim_{n\to\infty} (1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + ... + \frac{1}{n!}x^n)
\end{aligned}
$$

con lo que obtenemos la solución exacta

$$u(x) = e^x.$$

En la Figura 3.3 mostramos la comparación de la suma parcial $v(x) = 1 + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4$ con la solución exacta $u(x)$, se aprecia que la aproximación es bastante buena.

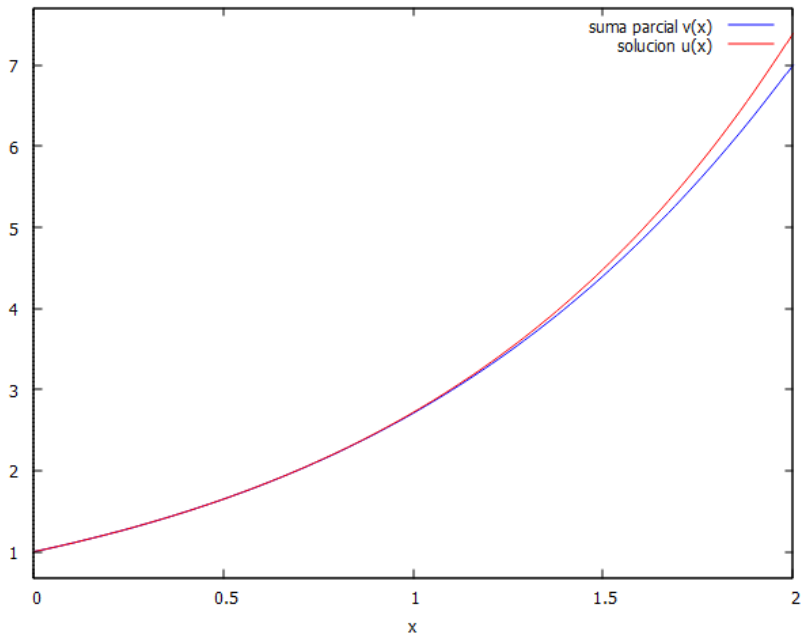

Figura 3.3.: Comparación de la solución exacta con la suma parcial $v(x)$

3.3.2. Método de aproximaciones sucesivas

El método de aproximaciones sucesivas, también conocido como el *método de iteración de Picard*, nos proporciona un esquema que puede ser utilizado para resolver problemas de valores iniciales o ecuaciones integrales. Este método encuentra aproximaciones sucesivas que convergen a la solución a partir de una inicial, llamada la *aproximación inicial*. Como veremos, la aproximación inicial puede ser cualquier función real que se utilizará en una relación recurrente para determinar las otras aproximaciones.

Dada la ecuación integral lineal de Volterra de segunda clase

$$u(x) = f(x) + \int_0^x K(x,t)u(t)dt,$$

donde $u(x)$ es la función desconocida a determinar, $K(x,t)$ es el núcleo, y λ un parámetro. El método de las aproximaciones sucesivas introduce la siguiente relación de recurrencia

$$u_n(x) = f(x) + \int_0^x K(x,t)u_{n-1}(t)dt, \qquad n \geqslant 1,$$

donde la aproximación inicial $u_0(x)$ puede ser cualquier función real, a diferencia de Adomian. Siempre empezamos con una suposición inicial para $u_0(x)$, normalmente se suele elegir 0 ó $f(x)$.

La convergencia de $u_n(x)$ está justificada con el estudio del capítulo 2, teniendo en cuenta que

$$u = \sum_{j=0}^{\infty} L^j f,$$

siendo L el operador lineal visto en (2.4), que

$$u_n = \sum_{j=0}^{n} L^j f,$$

y que el papel de f $(= u_0)$ lo puede jugar cualquier otra función continua g, ya que se tendría:

$$\begin{aligned} u_0 &= g \\ u_1 &= f + Lg \\ &\vdots \\ u_n &= (\sum_{j=0}^{n-1} L^j f) + L^n g. \end{aligned}$$

donde $L^n g \to 0$. Es más, en el mencionado capítulo 2 obtuvimos cotas del error para este método. En la siguiente observación vamos a ver algunas diferencias entre los métodos iterativos que hemos visto.

Observación 3.9. Es interesante ver que mientras que el método de iteración variacional utiliza la siguiente fórmula iterativa

$$u_{n+1}(x) = u_n(x) + \int_0^x \lambda(\xi)(\frac{\partial u_n(\xi)}{\partial \xi} - \tilde{u}_n(\xi))d\xi,$$

el método de aproximaciones sucesivas utiliza la siguiente:

$$u_n(x) = f(x) + \int_0^x K(x,t)u_{n-1}(t)dt, \qquad n \geqslant 1,$$

Podemos resumir las diferencias entre ambas fórmulas de la siguiente forma:

1. La primera fórmula contiene el multiplicador de Lagrange λ que debería ser determinado antes de aplicar la fórmula. Sin embargo, la fórmula de aproximaciones sucesivas no requiere el uso de λ.

2. La fórmula de iteración variacional permite el uso de la restricción $\tilde{u}_n(\xi)$ donde $\tilde{u}_n'(\xi) = 0$. La segunda fórmula no requiere esta restricción.

3. La primera fórmula se aplica a un ODE equivalente a la ecuación integral, mientras que la segunda fórmula se aplica directamente a la fórmula iterativa de la propia ecuación integral.

Vamos a ilustrar este método con un ejemplo para que se vea más claro.

Ejemplo 3.6. Vamos a resolver la siguiente ecuación integral de Volterra:

$$u(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{2}\int_0^x (x-t)^2 u(t)dt.$$

Para la aproximación inicial $u_0(x)$, seleccionamos

$$u_0(x) = 0.$$

El método de las aproximaciones sucesivas nos permite el uso de la siguiente fórmula iterativa

$$u_{n+1}(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{2}\int_0^x (x-t)^2 u_n(t)dt, \qquad n \geqslant 0.$$

Sustituyendo la aproximación inicial obtenemos:

$$\begin{aligned}
u_1(x) &= 1 + x + \frac{1}{2}x^2 + \frac{1}{2}\int_0^x (x-t)^2 u_0(t)dt = 1 + x + \frac{1}{2!}x^2, \\
u_2(x) &= 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4, + \frac{1}{5!}x^5, \\
u_3(x) &= 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4, + \frac{1}{5!}x^5 + \frac{1}{6!}x^6 + \frac{1}{7!}x^7 + \frac{1}{8!}x^8,
\end{aligned}$$

y así sucesivamente. La solución $u(x)$ viene dada por

$$u(x) = \lim_{n\to\infty} u_{n+1}(x) = e^x.$$

La solución de este ejemplo es idéntica a la que podemos ver en la Figura 3.3 del método de iteración variacional.

3.4. Otros métodos

3.4.1. Fenómeno de los términos de ruido

Esta nueva técnica depende principalmente de los llamados *términos de ruido*, y ha demostrado una rápida convergencia hacia la solución, se puede utilizar tanto para ecuaciones integrales como para ecuaciones diferenciales (véase [Waz11]).

Los términos con ruido, si existen entre las componentes $u_0(x)$ y $u_1(x)$, nos proporcionarán la solución exacta utilizando sólo las dos primeras iteraciones. Vamos a destacar los conceptos principales de estos términos:

1. Los *términos de ruido* se definen como los mismos términos con signos opuestos en las componentes $u_0(x)$ y $u_1(x)$. Pueden aparecer otros términos de ruido entre otras componentes, pero a nosotros nos interesarán los que aparezcan en las dos primeras.

2. Al cancelar los términos de ruido entre $u_0(x)$ y $u_1(x)$, aunque $u_1(x)$ contenga términos adicionales, los términos restantes que no han sido cancelados de $u_0(x)$ podrían proporcionar la solución exacta de la ecuación integral. Sin embargo, esto no siempre va a ser así, la aparición de términos de ruido entre $u_0(x)$ y $u_1(x)$ no tiene por qué garantizar la obtención de la solución exacta. Por lo tanto, es necesario demostrar que los términos no cancelados de $u_0(x)$ satisfacen la ecuación integral dada.

   Por otro lado, si los términos no cancelados de $u_0(x)$ no cumplieran con la ecuación integral dada, o los términos de ruido no aparecieran entre $u_0(x)$ y $u_1(x)$, entonces sería necesario determinar más componentes de $u(x)$ para obtener la solución en forma de serie.

3. Los términos de ruido aparecen en tipos específicos de ecuaciones no homogéneas, sin embargo, no aparecen en ecuaciones homogéneas.

4. Hay una condición necesaria para que se produzca la aparición de los términos de ruido, y es que la primera componente $u_0(x)$ debe contener la solución exacta $u(x)$, entre otros términos. Además, se demostró que la condición de no homogeneidad de la ecuación no siempre garantiza la aparición de los términos de ruido. Para más información acerca de los términos de ruido o la demostración de esta condición véase [Waz97], [Waz09] y [Waz99].

Vamos a ilustrar la utilidad de los términos de ruido con un ejemplo:

Ejemplo 3.7. Resolveremos la siguiente ecuación integral de Volterra:

$$u(x) = 8x + x^3 - \frac{3}{8}\int_0^x tu(t)dt.$$

Establecemos la relación de recurrencia siguiendo el método estándar de Adomian:

$$\begin{aligned} u_0(x) &= 8x + x^3, \\ u_1(x) &= -\frac{3}{8}\int_0^x tu(t)dt = -\frac{3}{40}x^5 - x^3. \end{aligned}$$

Podemos ver que $\pm x^3$ aparecen en $u_0(x)$ y $u_1(x)$, además con signos opuestos, por tanto es un término de ruido. Cancelando este término de la primera componente $u_0(x)$ obtenemos

la solución exacta:

$$u(x) = 8x,$$

que satisface la ecuación integral.

Observación 3.10. Si hubiéramos elegido el método modificado, seleccionamos $u_0(x) = 8x$, luego tenemos que $u_1(x) = 0$. Por tanto, obtenemos el mismo resultado.

3.4.2. Método de la Transformada de Laplace

Es una técnica muy potente que es capaz de resolver problemas de valores iniciales y ecuaciones integrales. Antes de aplicar el método, vamos a ver algunos conceptos importantes.

Si el núcleo $K(x,t)$ de la ecuación integral depende de la diferencia $x-t$, entonces se llama *núcleo de diferencia*. Y podemos expresar la ecuación integral de la siguiente forma:

$$u(x) = f(x) + \int_0^x K(x-t)u(t)dt. \tag{3.10}$$

Considerando dos funciones $f_1(x)$ y $f_2(x)$ que poseen las condiciones necesarias para la existencia de la transformada de Laplace, sean las transformadas de Laplace para las funciones $f_1(x)$ y $f_2(x)$ dadas por:

$$\mathcal{L}\{f_1(x)\} = F_1(s), \qquad \mathcal{L}\{f_2(x)\} = F_2(s).$$

El *producto de convolución de Laplace* de estas dos funciones se define como

$$(f_1 * f_2)(x) = \int_0^x f_1(x-t)f_2(t)dt,$$

ó

$$(f_2 * f_1)(x) = \int_0^x f_2(x-t)f_1(t)dt,$$

Observación 3.11. Recalcamos que

$$(f_1 * f_2)(x) = (f_2 * f_1)(x).$$

Podemos fácilmente ver que la transformada de Laplace del producto de convolución viene dada por:

$$\mathcal{L}\{(f_1 * f_2)(x)\} = \mathcal{L}\{\int_0^x f_1(x-t)f_2(t)dt\} = F_1(s)F_2(s).$$

Basándonos en este resumen de hechos bien conocidos, vamos a examinar ecuaciones integrales de Volterra específicas donde el núcleo es un núcleo de diferencia. Utilizaremos tanto la transformada de Laplace como su inversa.

Tomando la transformada de Laplace de ambos lados en (3.10) tenemos

$$U(s) = F(s) + \mathcal{K}(s)U(s),$$

donde

$$U(s) = \mathcal{L}\{u(x)\}, \qquad \mathcal{K}(s) = \mathcal{L}\{K(x)\}, \qquad F(s) = \mathcal{L}\{f(x)\}.$$

Resolviendo la ecuación para $U(s)$ obtenemos

$$U(s) = \frac{F(s)}{1 - \mathcal{K}(s)}, \qquad \mathcal{K}(s) \neq 1.$$

La solución $u(x)$ se obtiene tomando la inversa de la transformada de Laplace en la ecuación anterior donde obtenemos:

$$u(x) = \mathcal{L}^{-1}\{\frac{F(s)}{1 - \mathcal{K}(s)}\}.$$

Ejemplo 3.8. Resolveremos la siguiente ecuación integral de Volterra utilizando el nuevo método:

$$u(x) = \sin x + \cos x + 2\int_0^x \sin(x-t)u(t)dt.$$

Vamos a aplicar la transformada de Laplace, y nos apoyaremos en su linealidad para obtener

$$\mathcal{L}\{u(x)\} = \mathcal{L}\{\sin x + \cos x\} + 2\mathcal{L}\{\sin(x-t) * u(x)\},$$

y por tanto

$$U(s) = \frac{1}{s^2+1} + \frac{s}{s^2+1} + \frac{2}{s^2+1}U(s),$$

o equivalentemente

$$U(s) = \frac{1}{s-1}.$$

Tomando la inversa de Laplace en ambos lados obtenemos la solución exacta

$$u(x) = e^x.$$

3.4.3. Métodos de cuadratura

Son bien conocidos y de uso extendido. Nosotros hemos usado como referencia [KP02] para desarrollarlos.

Sea $h > 0$ un tamaño fijo y sean m, N tales que $Nh = B$, $m \leqslant N$. Entonces

$$u(mh) - \int_0^{mh} K(mh, s)u(s)ds = f(mh), \qquad mh \in [0, B].$$

Ahora, si usamos una fórmula de cuadratura de la forma $\int_0^{mh} F(s)ds = \sum_{j=0}^{m} w_{mj}F(jh)$, entonces la ecuación se aproxima por

$$u(mh) - \sum_{j=0}^{m} w_{mj}K(mh, jh)u(jh) = f(mh), \qquad mh \in [0, B],$$

donde $m = 1, 2, ..., N$ y tomando $x = 0$ tenemos el valor inicial $u(0) = f(0)$. Esto nos da el sistema de ecuaciones lineales (por tanto, resoluble sin problema, al menos para órdenes

bajos)

$$\begin{cases} u(0) = f(0), \\ w_{10}K(h,0)u(0) + (1 - w_{11}K(h,h))u(h) = f(h), \\ w_{20}K(2h,0)u(0) + w_{21}K(2h,h)u(h) + (1 - w_{22}K(2h,2h))u(2h) = f(2h), \\ \vdots \\ \sum_{j=0}^{N-1} w_{Nj}K(Nh,jh)u(jh) + (1 - w_{NN}K(Nh,Nh))u(Nh) = f(Nh). \end{cases}$$

La precisión del resultado depende de la elección de h, la suavidad de $K(x,s)$ y $f(x)$, y la regla de cuadratura elegida. Por ejemplo, si elegimos la regla trapezoidal, el sistema se convierte en

$$\begin{cases} u(mh) - h\sum_{j=0}^{m} \text{“}K(mh,jh)u(jh) = f(mh), \qquad m = 1,2,..., \\ u(0) = f(0), \end{cases} \tag{3.11}$$

que tiene un error de orden $O(h^2)$ cuando $h \to 0$.

Ejemplo 3.9. Consideramos $K(x,s) = 12$, y tenemos

$$u(x) - \int_0^x 12u(s)ds = f(x), \qquad x \in [0,B].$$

Ahora, usando la regla de cuadratura del trapecio (3.11) y escribimos la diferencia para $m = r$ y $m = r+1$ y restamos una de la otra, obteniendo la ecuación

$$(1 - 12h/2)u((r+1)h) - (1 + 12h/2)u(rh) = f((r+1)h) - f(rh) \equiv \Delta f(rh),$$

para $r = 0,1,2,....$ Luego la solución de la ecuación de Volterra viene dada por la recurrencia

$$u(rh) = \gamma^m f(0) + (1 - 12h/2)^{-1} \sum_{s=0}^{m-1} \gamma^{m-s-1} s\Delta f(sh),$$

donde $\gamma = (1 + 12h/2)/(1 - 12h/2)$. Tendremos que elegir h de forma que $h \neq 2/12$. El

x	h =1/20
0.0	1.0
0.2	0.820860
0.4	0.697396
0.6	0.872328
0.8	4.298881
1.0	46.161041

Figura 3.4.: Resultados del método para $h = 1/20$

error es de orden $O(h^2)$, se puede ver más detalladamente en [KP02].

Vamos a tomar como ejemplo $f(x) = 13e^{-x} - 12$, la solución exacta es $u(x) = e^{-x}$. Los resultados obtenidos con el método vienen dados en la Figura 3.4.

Aunque no sea objeto de este trabajo, se pueden ver otros métodos interesantes para la resolución de las ecuaciones integrales de Fredholm de segunda clase, como métodos de proyección, el método de Nyström u otros métodos iterativos, para entrar más en detalle y profundizar un poco más, véase [HA09] cap 12.

4. Algunos métodos para resolver sistemas de ecuaciones integrales de Volterra de segunda clase

4.1. Introducción

Los sistemas de ecuaciones integrales, lineales y no lineales, aparecen en muchas aplicaciones de ingeniería, física, química y modelos de crecimiento de poblaciones (véase en [Jer99] y [Lin85]). Las ideas generales y las características esenciales de estos sistemas se pueden aplicar en muchos ámbitos.

Una gran variedad de métodos numéricos y analíticos se usan para abordar estos sistemas, pero la mayoría encuentran dificultades en términos del gran trabajo computacional, sobre todo cuando el sistema incluye varias ecuaciones integrales. Para evitar estas dificultades que normalmente se ven en los métodos tradicionales, vamos a utilizar algunos de los métodos introducidos en el capítulo anterior, adaptándolos a este contexto más general. Los tres métodos que vamos a estudiar en este capitulo son los siguientes:

- Método de descomposición de Adomian
- Método de la transformada de Laplace
- Método de las aproximaciones sucesivas

En consonancia con el estudio analítico realizado en el capítulo 2, los sistemas de ecuaciones integrales lineales de Volterra de segunda clase, vienen dados por

$$\mathbf{u}(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}(t)dt, \qquad x \in [0, B].$$

Las funciones desconocidas $\mathbf{u}(x)$ que se determinarán son continuas, aparecen dentro y fuera de la integral. Los núcleos $\mathbf{K}(x,t)$ y las funciones $\mathbf{f}(x)$ son funciones continuas reales dadas. A continuación veremos los métodos para resolver estos sistemas.

Al igual que en la sección anterior, la información acerca de algunos de estos métodos ha sido extraída de [Waz11].

4.2. Método de descomposición de Adomian

Como ya vimos en caso escalar, este método descompone cada solución en la suma de una serie, donde cada componente se determina de forma recursiva. Ahora, el planteamiento es totalmente análogo:

$$\begin{aligned} \mathbf{u}_0(x) &= \mathbf{f}(x), \\ \mathbf{u}_{n+1}(x) &= \int_0^x \mathbf{K}(x,t)\mathbf{u}_n(t)dt, \qquad n \geqslant 0. \end{aligned}$$

Este método puede utilizarse en su forma estándar o combinando los términos de ruido. Además, el método de descomposición modificado se utilizará donde sea apropiado. Veamos un ejemplo para resolver un sistema de ecuaciones integrales de Volterra utilizando este método.

Ejemplo 4.1. Partimos del siguiente sistema:

$$\begin{cases} u(x) = x - \frac{1}{6}x^4 + \int_0^x ((x-t)^2 u(t) + (x-t)v(t))dt \\ v(x) = x^2 - \frac{1}{12}x^5 + \int_0^x ((x-t)^3 u(t) + (x-t)^2 v(t))dt. \end{cases}$$

El método nos sugiere que los términos lineales $u(x)$ y $v(x)$ se descompongan como una serie

$$u(x) = \sum_{n=0}^{\infty} u_n(x), \qquad v(x) = \sum_{n=0}^{\infty} v_n(x),$$

donde $u_n(x)$ y $v_n(x)$, $n \geqslant 0$ son los términos de $u(x)$ y $v(x)$ que encontraremos de forma recursiva.

Sustituyendo las series en el sistema obtenemos

$$\sum_{n=0}^{\infty} u_n(x) = x - \frac{1}{6}x^4 + \int_0^x ((x-t)^2 \sum_{n=0}^{\infty} u_n(t) + (x-t) \sum_{n=0}^{\infty} v_n(t))dt,$$

$$\sum_{n=0}^{\infty} v_n(x) = x^2 - \frac{1}{12}x^5 + \int_0^x ((x-t)^3 \sum_{n=0}^{\infty} u_n(t) + (x-t)^2 \sum_{n=0}^{\infty} v_n(t))dt.$$

Las primeras componentes $u_0(x)$ y $v_0(x)$ se definen como todos los términos que no están dentro de la integral, luego transformamos el sistema en un conjunto de relaciones recursivas dadas por

$$\begin{aligned} u_0(x) &= x - \frac{1}{6}x^4, \\ u_{k+1}(x) &= \int_0^x ((x-t)^2 u_k(t) + (x-t)v_k(t))dt, \qquad k \geqslant 0, \end{aligned}$$

y

$$\begin{aligned} v_0(x) &= x^2 - \frac{1}{12}x^5, \\ v_{k+1}(x) &= \int_0^x ((x-t)^3 u_k(t) + (x-t)^2 v_k(t))dt, \qquad k \geqslant 0. \end{aligned}$$

Si hacemos la primera iteración obtenemos

$$u_0(x) = x - \frac{1}{6}x^4, \qquad u_1(x) = \frac{1}{6}x^4 - \frac{1}{280}x^7,$$

y

$$v_0(x) = x^2 - \frac{1}{12}x^5, \qquad v_1(x) = \frac{1}{12}x^5 - \frac{11}{10080}x^8.$$

Es obvio que los términos de ruido $\pm\frac{1}{6}x^4$ aparecen entre $u_0(x)$ y $u_1(x)$. Además, los términos

de ruido $\pm\frac{1}{12}x^5$ aparecen entre $v_0(x)$ y $v_1(x)$. Si cancelamos estos términos de ruido en $u_0(x)$ y $v_0(x)$, el resto de términos restantes nos dan la solución exacta

$$(u(x), v(x)) = (x, x^2).$$

4.3. Método de la transformada de Laplace

Aunque ya hemos explicado este método en profundidad en el capítulo anterior en caso escalar, vamos a describir cómo sería para sistemas, y posteriormente mostraremos un ejemplo en el que veremos cómo se aplica a un sistema de ecuaciones integrales de Volterra.

Partimos de nuestro sistema

$$\mathbf{u}(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}(t)dt, \qquad x \in [0, B].$$

Tomando la transformada de Laplace de ambos lados tenemos

$$\mathbf{U}(s) = \mathbf{F}(s) + \mathbf{K}(s)\mathbf{U}(s),$$

donde

$$\mathbf{U}(s) = \mathcal{L}\{\mathbf{u}(x)\}, \qquad \mathbf{K}(s) = \mathcal{L}\{\mathbf{K}(x)\}, \qquad \mathbf{F}(s) = \mathcal{L}\{\mathbf{f}(x)\}.$$

Resolviendo la ecuación para $\mathbf{U}(s)$ obtenemos

$$\mathbf{U}(s) = \frac{\mathbf{F}(s)}{1 - \mathbf{K}(s)}, \qquad \mathbf{K}(s) \neq 1.$$

La solución $\mathbf{u}(x)$ se obtiene tomando la inversa de la transformada de Laplace en la ecuación anterior donde obtenemos:

$$\mathbf{u}(x) = \mathcal{L}^{-1}\{\frac{\mathbf{F}(s)}{1 - \mathbf{K}(s)}\}.$$

Ejemplo 4.2. Sea el sistema

$$\begin{cases} u(x) = 1 - x^2 + x^3 + \int_0^x ((x-t)u(t) + (x-t)v(t))dt \\ v(x) = 1 - x^3 - \frac{1}{10}x^5 + \int_0^x ((x-t)u(t) - (x-t)v(t))dt. \end{cases}$$

Es importante darse cuenta que ambos núcleos son iguales, es decir, $K_1(x-t) = K_2(x-t) = x - t$. Tomando la transformada de Laplace en ambos lados de cada ecuación obtenemos

$$U(s) = \mathcal{L}\{u(x)\} = \mathcal{L}\{1 - x^2 + x^3\} + \mathcal{L}\{(x-t) * u(x) + (x-t) * v(x)\},$$

$$V(s) = \mathcal{L}\{v(x)\} = \mathcal{L}\{1 - x^3 - \frac{1}{10}x^5\} + \mathcal{L}\{(x-t) * u(x) - (x-t) * v(x)\}.$$

Esto al mismo tiempo nos da

$$U(s) = \frac{1}{s} - \frac{2}{s^3} + \frac{6}{s^4} + \frac{1}{s^2}U(s) + \frac{1}{s^2}V(s),$$

$$V(s) = \frac{1}{s} - \frac{6}{s^4} - \frac{12}{s^6} + \frac{1}{s^2}U(s) - \frac{1}{s^2}V(s),$$

Reorganizando un poco el sistema obtenemos

$$(1 - \frac{1}{s^2})U(s) - \frac{1}{s^2}V(s) = \frac{1}{s} - \frac{2}{s^3} + \frac{6}{s^4},$$

$$(1 + \frac{1}{s^2})V(s) - \frac{1}{s^2}U(s) = \frac{1}{s} - \frac{6}{s^4} - \frac{12}{s^6}.$$

Resolviendo el sistema para $U(s)$ y $V(s)$ tenemos como resultado

$$U(s) = \frac{1}{s} + \frac{3!}{s^4},$$

$$V(s) = \frac{1}{s} - \frac{3!}{s^4}.$$

Tomando la inversa de la transformada de Laplace en ambos lados de cada ecuación, la solución exacta viene dada por

$$(u(x), v(x)) = (1 + x^3, 1 - x^3).$$

4.4. Método de aproximaciones sucesivas

El método de aproximaciones sucesivas también se puede aplicar a sistemas de ecuaciones integrales de Volterra en el caso vectorial. Cuando tenemos un sistema de ecuaciones integrales de Volterra en forma vectorial, el procedimiento es similar al caso escalar, pero trabajaremos con vectores en lugar de con funciones escalares, es decir, tenemos que

$$\mathbf{u}(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}(t)dt$$

donde $\mathbf{u}(x)$ es un vector de funciones que deseas encontrar, $\mathbf{f}(x)$ es un vector de funciones conocidas, y $\mathbf{K}(x,t)$ es la matriz de funciones núcleo, todos ellos continuos. Además, necesitaremos una función vectorial con las aproximaciones iniciales de cada ecuación, que denotaremos como $\mathbf{u}_0$. Veamos un ejemplo:

Ejemplo 4.3. Consideramos el siguiente sistema:

$$\begin{cases} y(x) = 1 + \int_0^x (\frac{1}{8}y(t) - \frac{1}{6}v(t))dt \\ v(x) = 3 + \int_0^x (\frac{1}{7}y(t) - \frac{1}{8}v(t))dt \end{cases}$$

donde tenemos los siguientes vectores:

$$\mathbf{u}(x) = \begin{pmatrix} y(x) \\ v(x) \end{pmatrix}, \qquad \mathbf{f}(x) = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \qquad \mathbf{K}(x,t) = \begin{pmatrix} \frac{1}{8} & -\frac{1}{6} \\ \frac{1}{7} & -\frac{1}{8} \end{pmatrix},$$

Además tenemos el vector de aproximaciones iniciales para ambas ecuaciones:

$$\mathbf{u}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Por tanto la fórmula iterativa que vamos a utilizar para calcular las aproximaciones sucesivas es

$$\mathbf{u}_{n+1}(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}_n(t)dt, \qquad n \geqslant 0.$$

Sustituyendo $\mathbf{u}_0$ obtenemos:

$$\mathbf{u}_1(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}_0(t)dt = \begin{pmatrix} 1 - \dfrac{t}{6} \\ 3 - \dfrac{t}{8} \end{pmatrix}$$

$$\mathbf{u}_2(x) = \mathbf{f}(x) + \int_0^x \mathbf{K}(x,t)\mathbf{u}_1(t)dt = \begin{pmatrix} 1 - \dfrac{3t}{8} \\ -\dfrac{11t^2}{1344} - \dfrac{13t}{56} + 3 \end{pmatrix}$$

Una solución más aproximada con la quinta iteración sería

$$\mathbf{u}_5(x) = \begin{pmatrix} -\dfrac{121t^5}{10838016} + \dfrac{121t^4}{1806336} + \dfrac{11t^3}{3584} - \dfrac{11t^2}{1344} - \dfrac{3t}{8} + 1 \\ -\dfrac{121t^5}{14450688} + \dfrac{121t^4}{602112} + \dfrac{143t^3}{75264} - \dfrac{11t^2}{448} - \dfrac{13t}{56} + 3 \end{pmatrix}.$$

Comparamos en la Figura 4.1 la aproximación con la solución y observamos que la aproximación ya es aceptable con la quinta iteración.

Observación 4.1. Todos los resultados de convergencia y error que vimos en el capítulo 2 para el caso escalar y que aplicamos en este método, se utilizan de forma análoga en el caso vectorial.

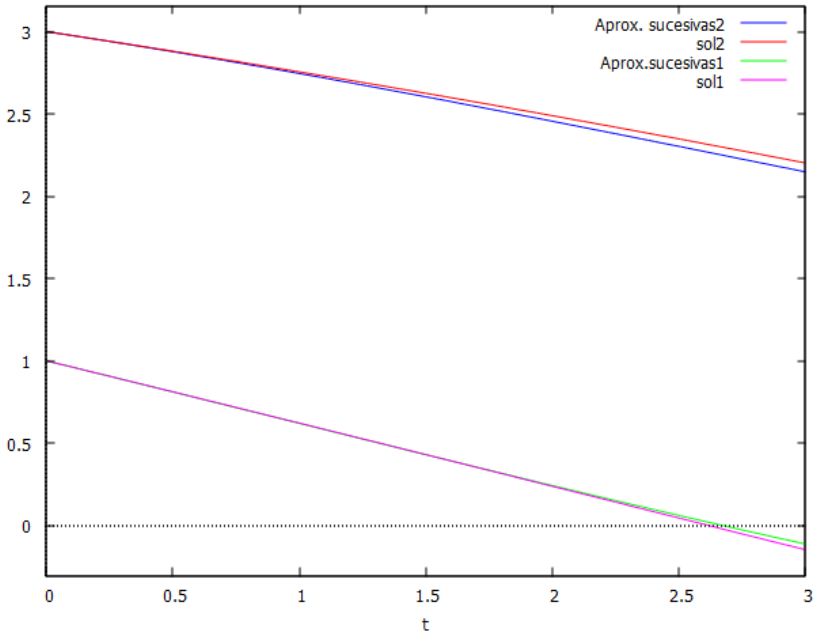

Figura 4.1.: Solución exacta y con aproximaciones sucesivas

5. Relación entre ecuaciones diferenciales y ecuaciones de Volterra

A continuación veremos la relación que existe entre ambos tipos de ecuaciones, más concretamente, veremos las ecuaciones diferenciales lineales como un caso particular de las ecuaciones de Volterra (lineales) de segunda clase. Primero vamos a ver el caso escalar, y después lo haremos vectorialmente.

5.1. Ecuación escalar

Consideramos el PVI:

$$x \in \mathcal{C}^1(0,B) : \begin{cases} x'(t) = a(t)x(t) + b(t) \\ x(0) = x_0 \end{cases}, \qquad t \in (0,B),$$

donde $a, b \in \mathcal{C}[0,B]$ y $x_0 \in \mathbb{R}$. La solución del PVI es el único punto fijo del operador lineal y continuo T, definido como

$$\begin{aligned} T : \mathcal{C}[0,B] &\longrightarrow \mathcal{C}[0,B] \\ x &\longmapsto T(x) : [0,B] \to \mathbb{R} \\ &\qquad (T(x))(t) = x_0 + \int_0^t (a(s)x(s) + b(s))ds. \end{aligned}$$

Por tanto, vemos que la solución del PVI es un caso particular de la ecuación integral lineal de Volterra de segunda clase:

$$x(t) = f(t) + \int_0^t K(t,s)x(s)ds,$$

donde

$$f(t) = x_0 + \int_0^t b(s)ds, \qquad K(t,s) = a(s).$$

Para ver la equivalencia, basta aplicar el Teorema Fundamental del Cálculo a la siguiente ecuación integral:

$$x(t) = x_0 + \int_0^t (a(s)x(s) + b(s))ds,$$

donde $x \in \mathcal{C}[0,B]$, y obtenemos, equivalentemente,

$$\begin{cases} x'(t) = a(t)x(t) + b(t) \\ x(0) = x_0 \end{cases}, \qquad t \in (0,B),$$

siendo $x \in \mathcal{C}^1[0,B]$. En definitiva, la solución del PVI inicial coincide con la solución de la ecuación de Volterra lineal de segunda clase.

5.2. Ecuación vectorial

En este caso, el PVI sería:

$$\begin{array}{l}\mathbf{x}\in\mathcal{C}^1([0,B],\mathbb{R}^n)\\ \mathbf{x}:[0,B]\longrightarrow\mathbb{R}^n\\ \quad t\longmapsto\mathbf{x}(t)=(x_1(t),...,x_n(t))\end{array}:\begin{cases}x_1'(t)=a_{11}(t)x_1(t)+\cdots+a_{1n}(t)x_n(t)+b_1(t)\\ \vdots\\ x_n'(t)=a_{n1}(t)x_1(t)+\cdots+a_{nn}(t)x_n(t)+b_n(t)\\ x_1(0)=x_1,...,x_n(0)=x_n,\end{cases}\tag{5.1}$$

donde $a_{ij}\in\mathcal{C}([0,B])$ y $b_i\in\mathcal{C}([0,B])$ para todo $i,j\in\{1,...,n\}$. Utilizando notación matricial para simplificar las fórmulas, nos quedaría de la siguiente forma:

$$\begin{array}{c}\mathbf{x}'(t)=\mathbf{A}(t)\mathbf{x}(t)+\mathbf{b}(t)\\ \mathbf{x}(0)=\mathbf{x_0}\end{array}$$

donde

$$\mathbf{A}(t)=\begin{pmatrix}a_{11}(t)&\cdots&a_{1n}(t)\\ \vdots&&\vdots\\ a_{n1}(t)&\cdots&a_{nn}(t)\end{pmatrix},\qquad\mathbf{x}(t)=\begin{pmatrix}x_1(t)\\ \vdots\\ x_n(t)\end{pmatrix},\qquad\mathbf{b}(t)=\begin{pmatrix}b_1(t)\\ \vdots\\ b_n(t)\end{pmatrix}$$

y el vector $\mathbf{x_0}$ está formado por los valores iniciales de cada una de las ecuaciones.

La solución de este PVI viene dada por el punto fijo del operador lineal y continuo

$$\begin{array}{c}\mathbf{T}:(\mathcal{C}[0,B],\mathbb{R}^n)\longrightarrow\mathcal{C}([0,B],\mathbb{R}^n)\\ \mathbf{x}\longmapsto\mathbf{Tx}:[0,B]\longrightarrow\mathbb{R}^n\\ \mathbf{Tx}(t)=\mathbf{x_0}+\int_0^t(\mathbf{A}(s)\mathbf{x}(s)+\mathbf{b}(s))ds.\end{array}$$

Luego al igual que en el caso escalar, la solución del PVI es un caso particular de la ecuación de Volterra de segunda clase

$$\mathbf{x}(t)=\mathbf{f}(t)+\int_0^t\mathbf{K}(t,s)\mathbf{x}(s)ds.$$

Ahora vemos la equivalencia aplicando el Teorema Fundamental del Cálculo a cada uno de los elementos de la matriz y de los vectores de la siguiente ecuación:

$$\mathbf{x}(t)=\mathbf{x_0}+\int_0^t(\mathbf{A}(s)\mathbf{x}(s)+\mathbf{b}(s))ds,$$

obteniendo:

$$\begin{array}{c}\mathbf{x}'(t)=\mathbf{A}(t)\mathbf{x}(t)+\mathbf{b}(t)\\ \mathbf{x}(0)=\mathbf{x_0}.\end{array}$$

Es decir, tanto de forma escalar como vectorial, tenemos el siguiente resultado.

Corolario 5.1. *La solución del PVI (5.1) coincide con la solución de la ecuación integral lineal de Volterra de segunda clase dada por*

$$\boldsymbol{x}(t)=\boldsymbol{x_0}+\int_0^t(\boldsymbol{A}(s)\boldsymbol{x}(s)+\boldsymbol{b}(s))ds.$$

Vamos a ver un ejemplo en el cual, partiendo de un PVI formado por un sistema de ecuaciones, lo transformaremos en un sistema de ecuaciones integrales de Volterra, y posteriormente, lo resolveremos mediante el método de aproximaciones sucesivas, luego lo volveremos a resolver utilizando el programa *Maxima*, y por último, compararemos los resultados obtenidos.

Ejemplo 5.1. Consideramos el PVI

$$\begin{cases} x'(t) = \dfrac{x(t)}{7} - \dfrac{y(t)}{10} + t \\ y'(t) = -\dfrac{x(t)}{8} + \dfrac{y(t)}{9} + 1 \\ x(0) = 0, \qquad y(0) = 0 \end{cases}, \qquad t \in [0,1],$$

que, como hemos visto anteriormente, es equivalente a la siguiente ecuación de Volterra:

$$\mathbf{x}(t) = \mathbf{x_0} + \int_0^t (\mathbf{A}(s)\mathbf{x}(s) + \mathbf{b}(s))ds.$$

donde

$$\mathbf{A}(t) = \begin{pmatrix} 1/7 & -1/10 \\ -1/8 & 1/9 \end{pmatrix}, \qquad \mathbf{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \qquad \mathbf{b}(t) = \begin{pmatrix} t \\ 1 \end{pmatrix}, \qquad \mathbf{x_0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Así, aplicaremos la siguiente fórmula iterativa para calcular las aproximaciones sucesivas:

$$\mathbf{x}_{n+1}(t) = \mathbf{f}(t) + \int_0^t \mathbf{K}(s,t)\mathbf{x}_n(s)ds, \qquad n \geqslant 0.$$

donde

$$\mathbf{f}(t) = \mathbf{x_0} + \int_0^t \mathbf{b}(s)ds, \qquad \mathbf{K}(t,s) = \mathbf{A}(s),$$

sustituimos $\mathbf{x_0}$ y obtenemos:

$$\mathbf{x}_1(t) = \mathbf{f}(t) + \int_0^t \mathbf{K}(s,t)\mathbf{x}_0(s)ds = \begin{pmatrix} \dfrac{t^2}{2} \\ t \end{pmatrix},$$

en la segunda iteración tenemos

$$\mathbf{x}_2(t) = \mathbf{f}(t) + \int_0^t \mathbf{K}(s,t)\mathbf{x}_1(s)ds = \begin{pmatrix} \dfrac{t^3}{14} + \dfrac{2t^2}{5} \\ -\dfrac{t^3}{16} + \dfrac{t^2}{9} + t \end{pmatrix},$$

Así, llegamos a que la iteración número cinco es:

$$\mathbf{x}_5(t) = \begin{pmatrix} \frac{2351441t^6}{2489356800} + \frac{494317t^5}{200037600} + \frac{8209t^4}{793800} + \frac{29t^3}{630} + \frac{2t^2}{5} \\ -\frac{9169t^6}{10001880} - \frac{1232909t^5}{514382400} - \frac{4057t^4}{408240} - \frac{61t^3}{1620} + \frac{t^2}{9} + t \end{pmatrix}$$

Ahora vamos a obtener la solución exacta gracias a Maxima, mediante los comandos que vemos en la Figura 5.1 , finalmente comparamos los resultados para $x(t)$ (Figura 5.2), en

```
(%i1)  eq1:'diff(x(t),t) = 1/7·x(t) −1/10·y(t) +t;
eq1    d/dt x(t) = −(y(t)/10) + x(t)/7 + t

(%i2)  eq2:'diff(y(t),t) = −1/8·x(t) +1/9·y(t) +1;
eq2    d/dt y(t) = y(t)/9 − x(t)/8 + 1

(%i3)  atvalue ( x ( t ) , t = 0 , 0 ) ;
(%o3)  0

(%i4)  atvalue ( y ( t ) , t = 0 ,0 ) ;
(%o4)  0

(%i5)  desolve ( [ eq1 , eq2 ] , [ x ( t ) ,y ( t ) ] ) ;
(%o5)  [x(t) = (%e^(8t/63) (3224027520 cosh(sqrt(20245) t/1260)/289 − 455388595200 sinh(sqrt(20245) t/1260)/(289 sqrt(20245))))/5040 − 560 t/17 − 639688/289,
        y(t) = (%e^(8t/63) (4125945600 cosh(sqrt(20245) t/1260)/289 − 590303246400 sinh(sqrt(20245) t/1260)/(289 sqrt(20245))))/5040 − 630 t/17 − 818640/289]
```

Figura 5.1.: Comandos de Maxima para obtener la solución exacta

azul vemos la solución del método de las aproximaciones sucesivas, y en rojo la solución de Maxima, de la misma forma para $y(t)$ (Figura 5.3). Vemos claramente cómo con 5 iteraciones, la convergencia es bastante buena y obtenemos una solución muy cercana a la exacta.

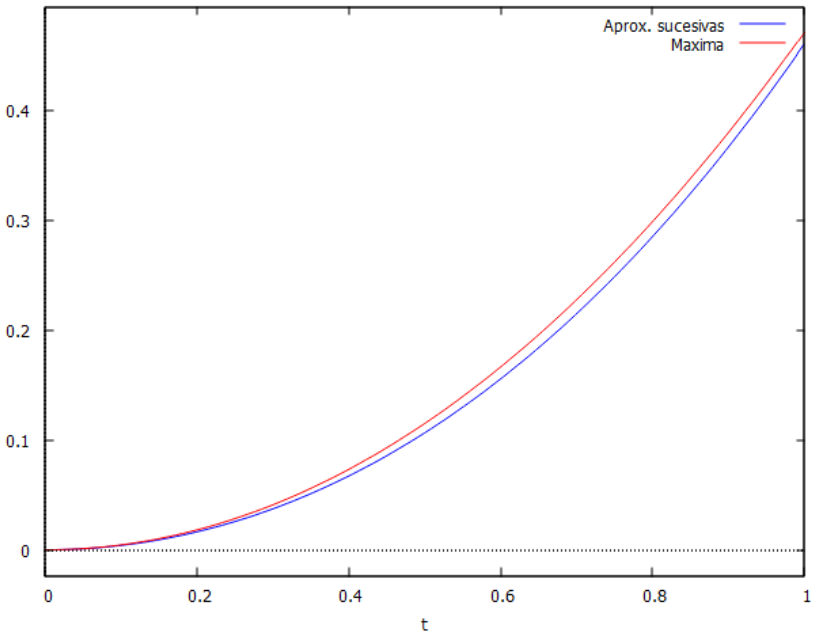

Figura 5.2.: Solución $x(t)$

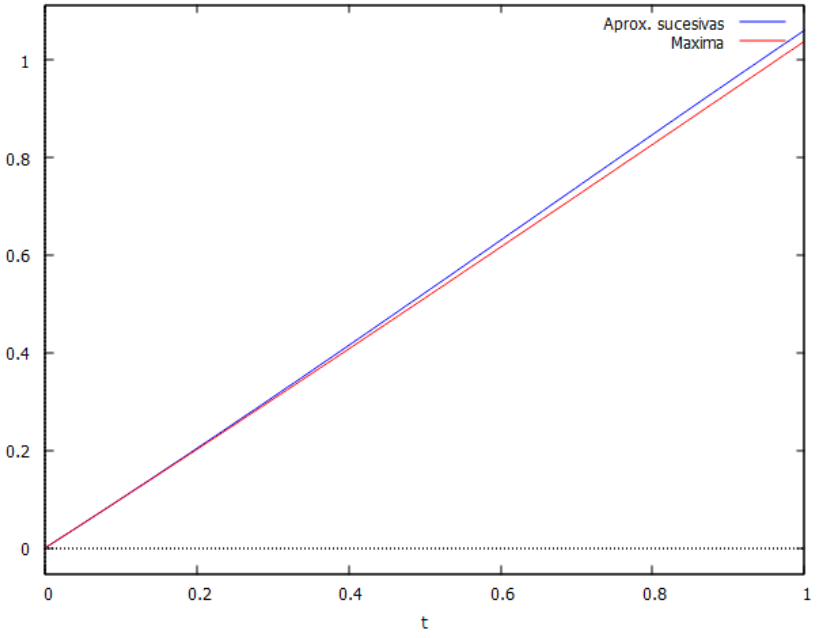

Figura 5.3.: Solución $y(t)$

6. Caso de uso: Calentamiento y enfriamiento de edificios

Como aplicación sobre lo estudiado anteriormente, vamos a formular un modelo matemático que describa la temperatura dentro de un edificio, como función de la temperatura exterior, el calor generado dentro del edificio y el sistema de aire acondicionado o calefacción.

Un enfoque natural para modelar la temperatura dentro de un edificio es el uso del análisis por compartimentos. A continuación vamos a ver diferentes modelos de edificios con distintos tipos y número de habitaciones, pondremos ejemplos y explicaremos cómo resolverlos, además observaremos que las soluciones son como deberíamos esperar en la vida real, pero eso sí, las cuantificamos de forma precisa.

En esta sección hemos recopilado información de [SM11], [NSS05] y [San23].

6.1. Edificio como habitación única

Sea $T(t)$ la temperatura dentro del edificio en el instante t y veamos al edificio como un único compartimento, es decir, sin estar dividido en varias habitaciones. Entonces la variación en la temperatura queda determinada por todos los factores que generan o disipan calor. Tomaremos en cuenta tres factores principales que afectan la temperatura dentro del edificio:

- En primer lugar está el calor generado dentro del propio edificio, por factores como pueden ser las personas o la maquinaria interna. A esta razón de incremento en la temperatura la denotaremos por $H(t)$.

- En segundo lugar está el calentamiento (o enfriamiento) proporcionado por la calefacción (o el aire acondicionado). Esta variación en la temperatura será representada como $U(t)$.

- El tercer factor es el efecto de la temperatura exterior, $M(t)$, sobre la temperatura interna del edificio. Este factor se puede modelizar mediante la **ley de enfriamiento de Newton**.

  Esta ley establece que hay una variación de la temperatura $T(t)$ que es proporcional a la diferencia entre la temperatura exterior $M(t)$ y la temperatura interior $T(t)$. Es decir, la variación en temperatura del edificio debida a $M(t)$ es

  $$K[M(t) - T(t)].$$

  La constante positiva K depende de las propiedades físicas del edificio, como la cantidad de puertas y ventanas o el tipo de aislamiento, pero K no depende de M, T o t. Por lo tanto, cuando la temperatura exterior es mayor que la temperatura interior, $M(t) - T(t) > 0$ y hay un incremento en la temperatura del edificio debido a $M(t)$.

  Por otro lado, cuando la temperatura exterior es menor que la temperatura interior, entonces $M(t) - T(t) < 0$ y la temperatura del edificio disminuye.

En general, la variación de calentamiento adicional $H(t)$ y la razón de calefacción (ó enfriamiento) $U(t)$ quedan descritas en términos de energía por unidad de tiempo. Sin embargo, al multiplicar por la capacidad calórica del edificio (en unidades de cambio de temperatura por unidad de energía calórica), podemos expresar las dos cantidades $H(t)$ y $U(t)$ en términos de temperatura por unidad de tiempo.

En definitiva, vemos que

$$\frac{dT}{dt} = K[M(t) - T(t)] + H(t) + U(t),$$

cuando la razón de calentamiento adicional $H(t)$ es siempre no negativa y $U(t)$ es positiva para la calefacción y negativa para el aire acondicionado. Como la ecuación es lineal, podemos resolverla fácilmente obteniendo como solución:

$$T(t) = e^{-Kt} \int e^{Kt} Q(t) dt + Ce^{-Kt} \tag{6.1}$$

$$= e^{-Kt} \{ \int e^{Kt} [KM(t) + H(t) + U(t)] dt + C \}.$$

A continuación vamos a resolver unos ejemplos sobre calentamiento y enfriamiento de edificios en el que tenemos solamente una habitación. Vamos a ver primero un ejemplo sencillo en el que suponemos la temperatura exterior constante:

Ejemplo 6.1. Suponemos que al final del día (en el instante t_0), cuando las personas salen del edificio, la temperatura exterior permanece constante e igual a M_0, la razón de calentamiento adicional H dentro del edificio se anula y la razón de uso del calefactor o el aire acondicionado U también se anula. Determinar $T(t)$, dada la condición inicial $T(t_0) = T_0$.
Solución. Con $M = M_0$, $H = 0$ y $U = 0$, la ecuación (6.1) se convierte en

$$T(t) = e^{-Kt} \{ \int e^{Kt} K M_0 dt + C \} = e^{-Kt} [M_0 e^{Kt} + C] = M_0 + Ce^{-Kt}.$$

Al hacer $t = t_0$ y usar el valor inicial T_0 de la temperatura, vemos que la constante C es $(T_0 - M_0)e^{Kt_0}$. Por lo tanto,

$$T(t) = M_0 + (T_0 - M_0)e^{-K(t-t_0)}.$$

Cuando $M_0 < T_0$, es decir, la temperatura exterior es menor que la interior, la solución de la ecuación decrece de manera exponencial a partir de la temperatura inicial T_0 hasta la temperatura final M_0.

Observación 6.1. Para determinar el tiempo que tarda la temperatura en cambiar, consideramos la sencilla ecuación lineal $dA/dt = -\alpha A$, cuyas soluciones son de la forma $A(t) = A(0)e^{-\alpha t}$. Cuando $t \to \infty$, la función $A(t)$ decae en forma exponencial ($\alpha > 0$) o crece de manera exponencial ($\alpha < 0$). En cualquier caso, el tiempo que tarda $A(t)$ en cambiar de $A(0)$ a $A(0)/e$ es justamente $1/\alpha$, ya que

$$A(\frac{1}{\alpha}) = A(0)e^{-\alpha(1/\alpha)} = \frac{A(0)}{e}.$$

La cantidad $1/|\alpha|$, que no depende de $A(0)$, es la **constante de tiempo**.

Volviendo al ejemplo 1, vemos que la temperatura $T(t)$ satisface la ecuación

$$\frac{dT}{dt}(t) = -KT(t) + KM_0, \qquad \frac{d(T - M_0)}{dt}(t) = -K[T(t) - M_0],$$

donde M_0 es una constante, en cualquier caso, la constante de tiempo es justamente $1/K$, lo que representa el tiempo que tarda la diferencia de temperaturas $T - M_0$ en cambiar de $T - M_0$ a $(T_0 - m_0)/e$. También diremos que $1/K$ es la **constante de tiempo para el edificio** (sin calefacción o aire acondicionado). A mayor sea el valor de esta constante, mayor aislamiento habrá, y por tanto, más tiempo tardará en transferirse el calor entre el exterior y el interior del edificio. Un valor típico para la constante de tiempo de un edificio es de 2 a 4 horas, pero puede ser menor si las ventanas están abiertas o existe algún ventilador, ya que obviamente, tendremos menos aislamiento. También puede ser mayor si el edificio está aislado de otros edificios. El valor de esta constante depende también de los materiales que tiene el edificio, todo esto se puede ver con más detalle en [San23].

Vamos a ver un caso concreto, en el que tendremos como datos:

$$M_0 = 30°C, \qquad T_0 = 20°C, \qquad K = 3,$$

entonces, sustituyendo en la fórmula de la temperatura exterior, obtenemos

$$T(t) = 30 + (20 - 30)e^{-3t}.$$

En la Figura 6.1, podemos ver cómo la temperatura exterior se mantiene constante, y la interior va creciendo exponencialmente, sin embargo, cuanto más cerca se encuentra de la temperatura exterior, la variación es menor, hasta volverse también constante en los 30°C.

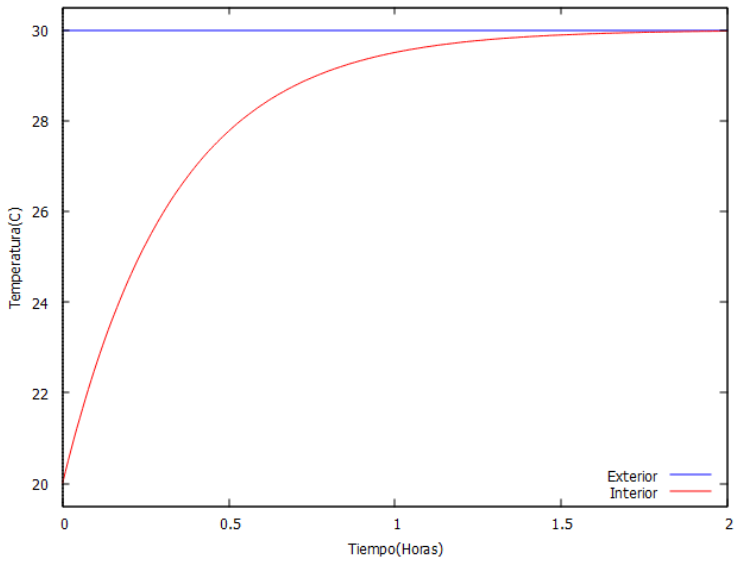

Figura 6.1.: Variación de la temperatura manteniendo la temperatura exterior constante

Ejemplo 6.2. Determinar la temperatura del edificio $T(t)$ si la razón de calentamiento adicional $H(t)$ es igual a la constante H_0, no hay calentamiento ni enfriamiento $(U(t) = 0)$ y la

temperatura exterior M varía como una onda senoidal en un periodo de 24 horas, con un mínimo en $t = 0$ (medianoche) y un máximo en $t = 12$ (mediodía). Es decir,

$$M(t) = M_0 - B\cos\omega t,$$

donde B es una constante positiva, M_0 es la temperatura exterior promedio y $\omega = 2\pi/24 = \pi/12$ radianes/hora. (Esto podría ocurrir durante la primavera o el otoño cuando no hay calefactor ni aire acondicionado).
Solución. Definimos la función $Q(t)$ como

$$Q(t) = K(M_0 - B\cos\omega t) + H_0.$$

Al hacer $B_0 := M_0 + H_0/K$, podemos escribir Q como

$$Q(t) = K(B_0 - B\cos\omega t),$$

donde KB_0 representa el valor promedio diario de $Q(t)$, es decir,

$$KB_0 = \frac{1}{24}\int_0^{24} Q(t)dt.$$

Cuando la función de forzamiento $Q(t)$ que hemos obtenido se sustituye en la expresión para la temperatura en la ecuación (6.1), el resultado, después de integrar por partes, es

$$T(t) = e^{-Kt}[\int e^{Kt}(KB_0 - KB\cos\omega t)dt + C]$$

$$T(t) = B_0 - BF(t) + Ce^{-Kt}, \tag{6.2}$$

donde

$$F(t) := \frac{\cos\omega t + (\omega/K)\sin\omega t}{1 + (\omega/K)^2}$$

Elegimos la constante C de modo que en medianoche $(t = 0)$, el valor de temperatura T sea igual a cierta temperatura inicial T_0. Así,

$$C = T_0 - B_0 + BF(0) = T_0 - B_0 + \frac{B}{1 + (\omega/K)^2}.$$

Observamos que el tercer término de la solución (6.2) que multiplica a la constante C tiende a cero exponencialmente. El término constante B_0 es igual a $M_0 + H_0/K$y representa la temperatura promedio diaria dentro del edificio (despreciando el término exponencial). Cuando no hay una razón de calentamiento adicional dentro del edificio $(H_0 = 0)$, esta temperatura promedio es igual a la temperatura exterior promedio M_0. El término $BF(t)$ representa la variación senoidal de la temperatura dentro del edificio correspondiente a la variación de la temperatura en el exterior.

Como $F(t)$ se puede escribir en la forma

$$F(t) = [1 + (\omega/K)^2]^{-1/2}\cos(\omega t - \phi),$$

donde $\tan\phi = \omega/K$, la variación senoidal dentro del edificio se retrasa con respecto de la variación en el exterior por ϕ/ω

horas. Además, la magnitud de la variación dentro del edificio es ligeramente menor, por un factor de $[1 + (\omega/K)^2]^{-1/2}$, que la variación en el exterior. La frecuencia angular de variación ω es $2\pi/24$ radianes/hora (aproximadamente $1/4$). Los valores usuales para la razón adimensional ω/K están entre $1/2$ y 1. Para este rango, el retraso entre la temperatura interior y la exterior es aproximadamente de 1.8 a 3 horas y la magnitud de la variación interior está entre 89 y 71 % de la variación en el exterior. En la Figura 6.2 podemos ver la

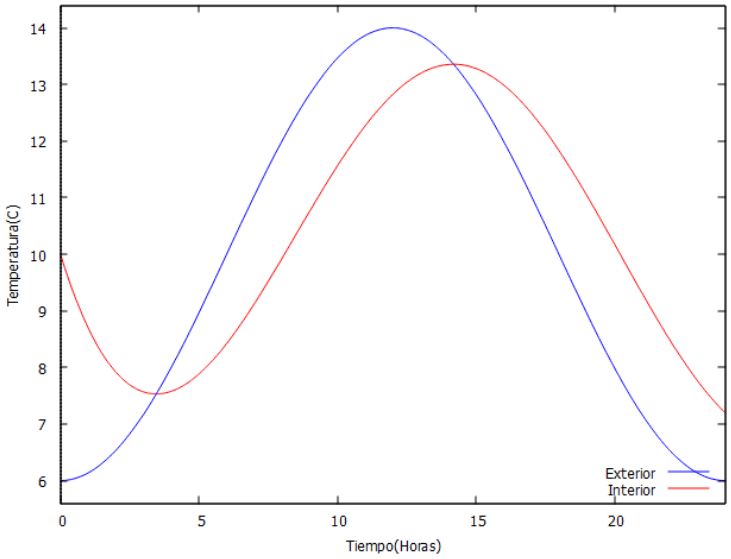

Figura 6.2.: Variación de la temperatura dentro y fuera de un edificio sin calefacción

variación senoidal de 24 horas de la temperatura exterior para un día moderado típico así como las variaciones de temperatura dentro del edificio para una razón adimensional ω/K de la unidad, que corresponde a una constante de tiempo $1/K$ de aproximadamente 4 horas. Los valores que hemos tomado para realizar este ejemplo concreto son:

- $M_0 = 10$°C (Temperatura exterior promedio)
- $T_0 = 10$°C (Temperatura interior inicial)
- $K = 0.4$
- $B = 4$

Podemos ver cómo la temperatura interior del edificio sigue el comportamiento esperado, es decir, disminuye cuando es mayor que la exterior, y viceversa, y además, a mayor es la distancia entre ambas temperaturas, mayor es la variación de la temperatura interior del edificio, como es natural. Al trazar esta última curva, hemos supuesto que el término exponencial ha desaparecido.

6.2. Ejemplo con dos habitaciones

Ahora consideremos el mismo problema con dos zonas, de modo que el calor se transfiere de una a otra en función de la diferencia de temperatura. Suponemos además que alguna

de las zonas posee una fuente de calor (o de enfriamiento) que hará que ésta se caliente (o enfríe) en función de su capacidad calorífica. La variación de temperatura en cada zona será la suma del calor (o frío) generado por dicha fuente, si existe en esa zona, y la pérdida o ganancia de calor generada por el contacto con otras zonas o con el exterior. Para calcular las ecuaciones aplicaremos la *ley de Newton del enfriamiento* vista anteriormente.

Ejemplo 6.3. Un estudio consta de dos zonas: la zona A de la planta alta y la zona B de la planta baja como podemos ver en la Figura 6.3. La planta baja, que tiene una capacidad calorífica de $(1/5)°C/1000$ btu (btu: unidades térmicas británicas), es calentada por un calefactor que genera 90000 btu por hora. Las constantes de tiempo de transferencia de calor son: 3 horas entre la planta baja y el exterior, 1/2 hora entre la planta alta y el exterior y 1/2 hora entre las dos plantas. Si la temperatura en el exterior permanece constante a $2°C$ e inicialmente ambas zonas estaban a $22°C$, calculemos la temperatura en la planta baja al cabo de 1 hora.

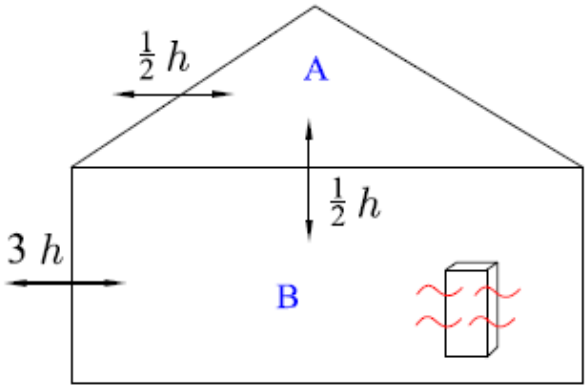

Figura 6.3.: Calentamiento de edificio compuesto por dos zonas.

Solución. En este caso, tenemos tres regiones, la zona A, la zona B y el exterior, luego tendremos que tener en cuenta la transferencia de calor entre las tres.

Sea $x(t)$ la temperatura en la zona A en un instante t y sea $y(t)$ la temperatura en la zona B en un instante t.

La zona B recibe el calor generado por el calefactor a razón de $90000btu/h$, puesto que su capacidad calorífica es de $(1/5)°C/1000$ btu, tendremos que la temperatura que gana B es:

$$90000btu/h \times (1/5)°C/1000btu = 18°C/h.$$

La variación de temperatura en las zonas A y B vendrá dada por:

$$\frac{dx}{dt} = 2(2-x) + 2(y-x)$$

$$\frac{dy}{dt} = \frac{1}{3}(2-y) + 2(x-y) + 18.$$

Por tanto, tenemos el sistema no homogéneo:

$$\frac{dx}{dt} = -4x + 2y - 4$$

$$\frac{dy}{dt} = 2x - \frac{7}{3}y + \frac{56}{3}$$

Resolvamos el sistema. La ecuación característica es:

$$\begin{vmatrix} -4-r & 2 \\ 2 & -\frac{7}{3}-r \end{vmatrix} = 0 \to 3r^2 + 19r + 16 = 0,$$

cuyas raíces son los valores propios: $r_1 = -1 y r_2 = -\frac{16}{3}$. Calculemos los vectores propios asociados a cada valor:

$$H_{-1} = \{(x,y) : \begin{pmatrix} -3 & 2 \\ 2 & -\frac{4}{3} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}\} = \{(x,y) : -3x + 2y = 0\},$$

$$H_{-16/3} = \{(x,y) : \begin{pmatrix} \frac{4}{3} & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}\} = \{(x,y) : 2x + 3y = 0\},$$

Por tanto, un vector propio asociado a $r_1 = -1$ es $\vec{u}_1 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ y un vector propio asociado a $r_2 = -\frac{16}{3}$ es $\vec{u}_2 = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$ y la solución de la parte homogénea resulta:

$$\vec{x}_h(t) = \begin{pmatrix} x_h(t) \\ y_h(t) \end{pmatrix} = C_1 e^{-t} \begin{pmatrix} 2 \\ 3 \end{pmatrix} + C_2 e^{-16t/3} \begin{pmatrix} 3 \\ -2 \end{pmatrix}.$$

Ahora buscamos una solución particular mediante el método de los coeficientes indeterminados. Puesto que el término no homogéneo es un polinomio de grado 0 y además 0 no es raíz de la ecuación característica, podemos tomar la solución particular de la forma:

$$\vec{x}_p = \vec{a} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}.$$

Derivando y sustituyendo en la ecuación, tenemos:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -4 & 2 \\ 2 & -\frac{7}{3} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} + \begin{pmatrix} 4 \\ 56/3 \end{pmatrix}$$

pasando el término independiente al otro lado de la igualdad,

$$\begin{pmatrix} -4 & 2 \\ 2 & -\frac{7}{3} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -4 \\ -56/3 \end{pmatrix}$$

de donde obtenemos: $a_1 = 35/4$ y $a_2 = 31/2$. La solución general es:

$$\vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = C_1 e^{-t} \begin{pmatrix} 2 \\ 3 \end{pmatrix} + C_2 e^{-16t/3} \begin{pmatrix} 3 \\ -2 \end{pmatrix} + \begin{pmatrix} 35/4 \\ 31/2 \end{pmatrix}$$

Considerando las condiciones iniciales: para $t = 0, x(0) = 22$ y $y(0) = 22$, se tiene:

$$\begin{pmatrix} 22 \\ 22 \end{pmatrix} = C_1 \begin{pmatrix} 2 \\ 3 \end{pmatrix} + C_2 \begin{pmatrix} 3 \\ -2 \end{pmatrix} + \begin{pmatrix} 35/4 \\ 31/2 \end{pmatrix}$$

Agrupando los términos independientes y reescribiendo los sumandos multiplicados por las constantes C_1 y C_2 en términos de la matriz fundamental, tenemos:

$$\begin{pmatrix} 2 & 3 \\ 3 & -2 \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 53/4 \\ 13/2 \end{pmatrix}$$

y resolviendo el sistema obtenemos:

$$C_1 = 46/13 \qquad y \qquad C_2 = 107/52.$$

La solución a este problema de valor inicial es:

$$\vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \frac{46}{13} e^{-t} \begin{pmatrix} 2 \\ 3 \end{pmatrix} + \frac{107}{52} e^{-16t/3} \begin{pmatrix} 3 \\ -2 \end{pmatrix} + \begin{pmatrix} 35/4 \\ 31/2 \end{pmatrix}.$$

Puesto que la temperatura en B era $y(t)$, ésta al cabo de $1h$ será:

$$y(1) = \frac{46}{13} e^{-t} 3 + \frac{107}{52} e^{-16t/3}(-2) + \frac{31}{2} \approx 19,405°C.$$

6.3. Modelo con 3 habitaciones

Vamos a estudiar ahora un caso un poco más complejo con 3 habitaciones, un par de aires acondicionados y hasta 10 personas en una sola habitación, podremos ver si la temperatura en cada habitación se comporta como debería, en la Figura 6.4 vemos el edificio con el que vamos a trabajar. Los datos que vamos a utilizar son:

- Temperatura en A $\rightarrow a(t)$
- Temperatura en B $\rightarrow b(t)$
- Temperatura en C $\rightarrow c(t)$
- Temperatura en el exterior F $\rightarrow M(t) = 5 - 10\cos(t \cdot (\pi/12))$
- Constante de transferencia entre cada habitación $K = 5$
- Constante de transferencia entre una habitación y el exterior $K_{xF} = 3$
- Calor generado por cada aire acondicionado U $\rightarrow U(t) = -8°\text{C}/5h$
- Calor generado en el interior H $\rightarrow H(t) = 0.018$ por cada persona

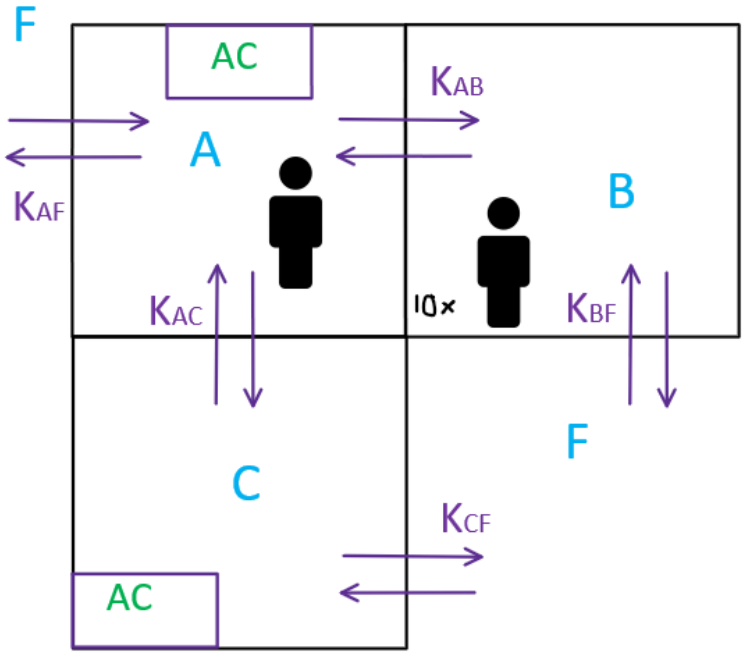

Figura 6.4.: Edificio con 3 habitaciones contiguas, personas y aires acondicionados

Con la estructura del edificio, obtenemos el sistema de ecuaciones:

$$\begin{cases} a'(t) = \dfrac{b(t)-a(t)}{K_{AB}} + \dfrac{c(t)-a(t)}{K_{AC}} + \dfrac{M(t)-a(t)}{K_{AF}} + U(t) + H(t) \\ b'(t) = \dfrac{a(t)-b(t)}{K_{AB}} + \dfrac{M(t)-b(t)}{K_{BF}} + 10H(t) \\ c'(t) = \dfrac{a(t)-c(t)}{K_{AC}} + \dfrac{M(t)-c(t)}{K_{CF}} + U(t) \end{cases}$$

Como valores iniciales para las temperaturas de cada habitación tomaremos:

$$a(t_0) = 20°\text{C}, \qquad b(t_0) = 10°\text{C}, \qquad c(t_0) = 15°\text{C}.$$

Resolvemos el sistema con *Maxima* y obtenemos la solución gráficamente en la Figura 6.5. Como conclusión podemos observar que los resultados son los naturales, ya que la habitación B es la que se mantiene más caliente de las tres, lo que es normal al no tener aire acondicionado y además tiene 10 personas generando calor en su interior, por otro lado, la habitación C es un poco más fría que la A debido a que no tiene ninguna persona dentro, y finalmente vemos como todas las temperaturas interiores van variando según la temperatura exterior, pero siempre de forma más suave que ésta, como era de esperar.

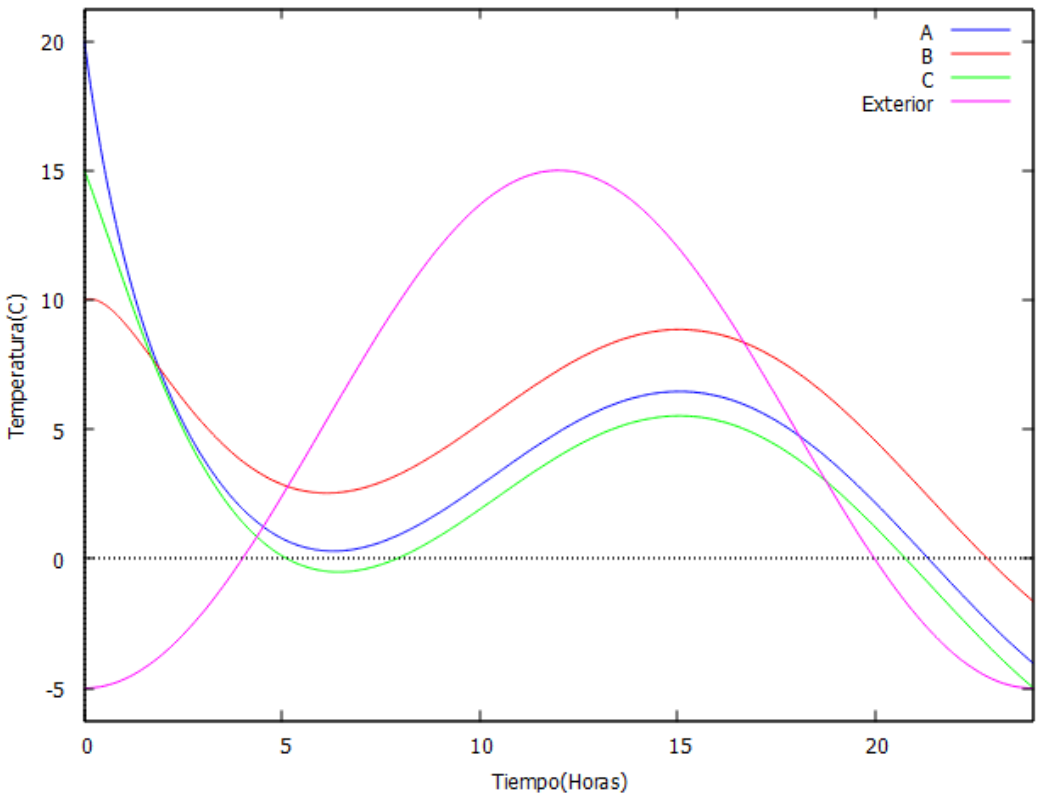

Figura 6.5.: Temperaturas de las 3 habitaciones con temperatura exterior variable

6.4. Modelos de un edificio con 5 habitaciones

6.4.1. Primer ejemplo (temperatura exterior constante)

Vamos a estudiar un par de casos de edificios con 5 habitaciones, donde incluiremos personas, y aires acondicionados para hacer el edificio más realista. El primer caso será el que tenemos en la Figura 6.6. Podemos ver 5 habitaciones contiguas y 2 aires acondicionados en las habitaciones B y D. Los datos que tomaremos son los siguientes:

- Temperatura en A $\rightarrow a(t)$
- Temperatura en B $\rightarrow b(t)$
- Temperatura en C $\rightarrow c(t)$
- Temperatura en D $\rightarrow d(t)$
- Temperatura en E $\rightarrow e(t)$
- Temperatura en el exterior F $\rightarrow M(t) = 35$°C
- Constante de transferencia entre cada habitación $K = 4$
- Constante de transferencia entre una habitación y el exterior $K_{xF} = 2$

- Calor generado por cada aire acondicionado U $\rightarrow U(t) = -4°C/5h$
- Calor generado en el interior H $\rightarrow H(t) = 0$

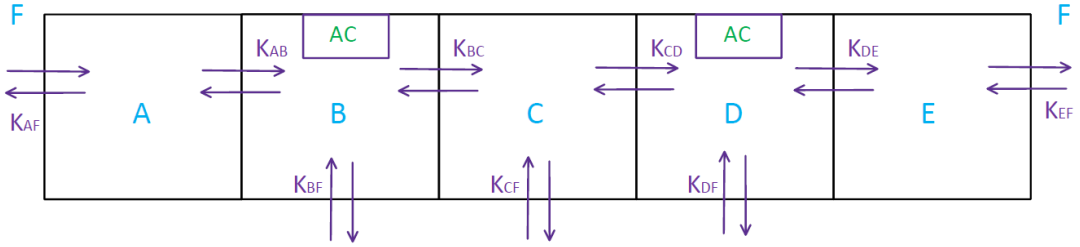

Figura 6.6.: Edificio con 5 habitaciones contiguas

Observando la estructura del edificio, obtenemos el siguiente sistema de ecuaciones:

$$\begin{cases} a'(t) = \dfrac{b(t)-a(t)}{K_{AB}} + \dfrac{M(t)-a(t)}{K_{AF}} \\ b'(t) = \dfrac{a(t)-b(t)}{K_{AB}} + \dfrac{c(t)-b(t)}{K_{BC}} + \dfrac{M(t)-b(t)}{K_{BF}} + U_B(t) \\ c'(t) = \dfrac{b(t)-c(t)}{K_{BC}} + \dfrac{d(t)-c(t)}{K_{CD}} + \dfrac{M(t)-c(t)}{K_{CF}} \\ d'(t) = \dfrac{c(t)-d(t)}{K_{CD}} + \dfrac{e(t)-d(t)}{K_{DE}} + \dfrac{M(t)-d(t)}{K_{DF}} + U_D(t) \\ e'(t) = \dfrac{d(t)-e(t)}{K_{DE}} + \dfrac{M(t)-e(t)}{K_{EF}} \end{cases}$$

Como valores iniciales para las temperaturas de cada habitación tomaremos:

$$a(t_0) = 25°C, \qquad b(t_0) = 25°C, \qquad c(t_0) = 28°C, \qquad d(t_0) = 20°C, \qquad e(t_0) = 20°C.$$

Resolvemos el sistema con *Maxima* y obtenemos la solución gráficamente en la Figura 6.7. Es importante observar varias cosas:

- Las habitaciones D y E comienzan con la misma temperatura, pero hasta la tercera hora hace más calor en la habitación D, ¿Cómo puede ser si esta habitación tiene aire acondicionado? Pues porque está pegada a la habitación C en la que hace mucho más calor, luego la variación de temperatura se ve más afectada por esto que por el aire acondicionado, pero vemos como cuando todas las habitaciones se van aproximando en cuanto a temperatura, el efecto del aire acondicionado hace que la habitación D sea la más fresquita, como era de esperar.
- Por un motivo similar, vemos que la habitación C incluso se enfría en la primera hora, pese a que fuera están a 35°C, esto es debido a que la constante de transferencia $1/K$ entre cada habitación es menor que hacia el exterior, y al estar las otras habitaciones más frías al principio, se enfría un poco, pero en cuanto todas las demás habitaciones se van calentando, deja de tener efecto sobre la habitación C y emprende su camino hacia los 35°C del exterior.

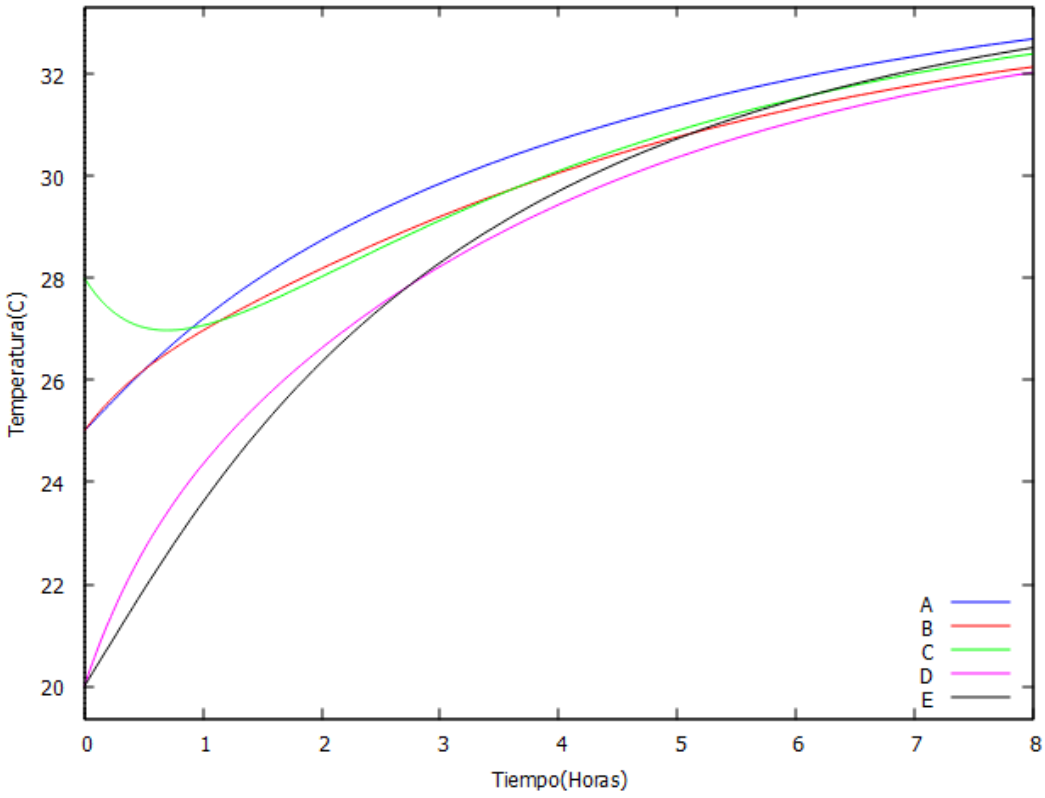

Figura 6.7.: Temperaturas de las 5 habitaciones con temperatura exterior constante de 35°C

- Como era de esperar, al principio la variación de temperatura es mayor por haber más diferencia entre las habitaciones y el exterior, y conforme va pasando el tiempo, se van aproximando a los 35°C y la variación va disminuyendo.
- En general, las habitaciones A y E son las que más variación de temperatura tienen, esto es naturalmente debido a que sólo tienen cerca una habitación, y se ven más influenciadas por el exterior, además de que no tienen aire acondicionado.

6.4.2. Segundo ejemplo (temperatura exterior variable)

En el siguiente caso vamos a colocar algunas personas en las habitaciones, y además vamos a cambiar un poco la estructura interna del edificio, tomaremos la Figura 6.8 como referencia.

Vamos a tomar como variables:

- Temperatura en el exterior F $\rightarrow M(t) = 18 - 12\cos(t \cdot (\pi/12))$
- Constante de transferencia entre cada habitación $K = 4$
- Constante de transferencia entre una habitación y el exterior $K_{xF} = 2$
- Calor generado por cada aire acondicionado U $\rightarrow U(t) = -2°C/h$
- Calor generado en el interior H $\rightarrow H(t) = 0.018$ por cada persona

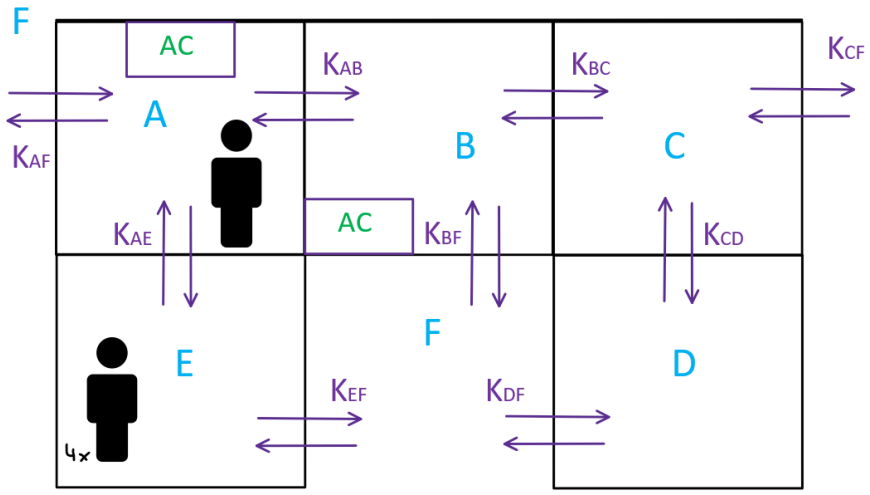

Figura 6.8.: Edificio con 5 habitaciones, 1 persona en A y 4 personas en E

Atendiendo a la estructura del edificio, obtenemos el siguiente sistema:

$$\begin{cases} a'(t) = \dfrac{b(t)-a(t)}{K_{AB}} + \dfrac{e(t)-a(t)}{K_{AE}} + \dfrac{M(t)-a(t)}{K_{AF}} + U_A(t) + H(t) \\ b'(t) = \dfrac{a(t)-b(t)}{K_{AB}} + \dfrac{c(t)-b(t)}{K_{BC}} + \dfrac{M(t)-b(t)}{K_{BF}} + U_B(t) \\ c'(t) = \dfrac{b(t)-c(t)}{K_{BC}} + \dfrac{d(t)-c(t)}{K_{BC}} + \dfrac{M(t)-c(t)}{K_{CF}} \\ d'(t) = \dfrac{c(t)-d(t)}{K_{BD}} + \dfrac{M(t)-d(t)}{K_{DF}} \\ e'(t) = \dfrac{a(t)-e(t)}{K_{AE}} + \dfrac{M(t)-e(t)}{K_{EF}} + 4H(t) \end{cases}$$

Como valores iniciales para las temperaturas de cada habitación tomaremos:

$$a(t_0) = 10°C, \qquad b(t_0) = 0°C, \qquad c(t_0) = 12°C, \qquad d(t_0) = 17°C, \qquad e(t_0) = 8°C.$$

Ahora resolvemos el sistema con *Maxima* y obtenemos las temperaturas de cada habitación gráficamente en la Figura 6.9. En este caso tenemos varios detalles que observar:

- Se aprecia perfectamente cómo la temperatura interior tiene una variación más suave que la exterior, algo natural debido a los aislantes térmicos que tiene el edificio.
- Vemos cómo las habitaciones A y B, que tienen un potente aire acondicionado, siempre se mantienen más frías que el resto durante todo el día.
- Pese a que al principio cada habitación tiene una temperatura bastante diferente, al final todas se van aproximando entre ellas, debido a que lo que más las influencia es la temperatura exterior, es decir, es el mayor factor que afecta a la variación de la temperatura interna de este edificio.

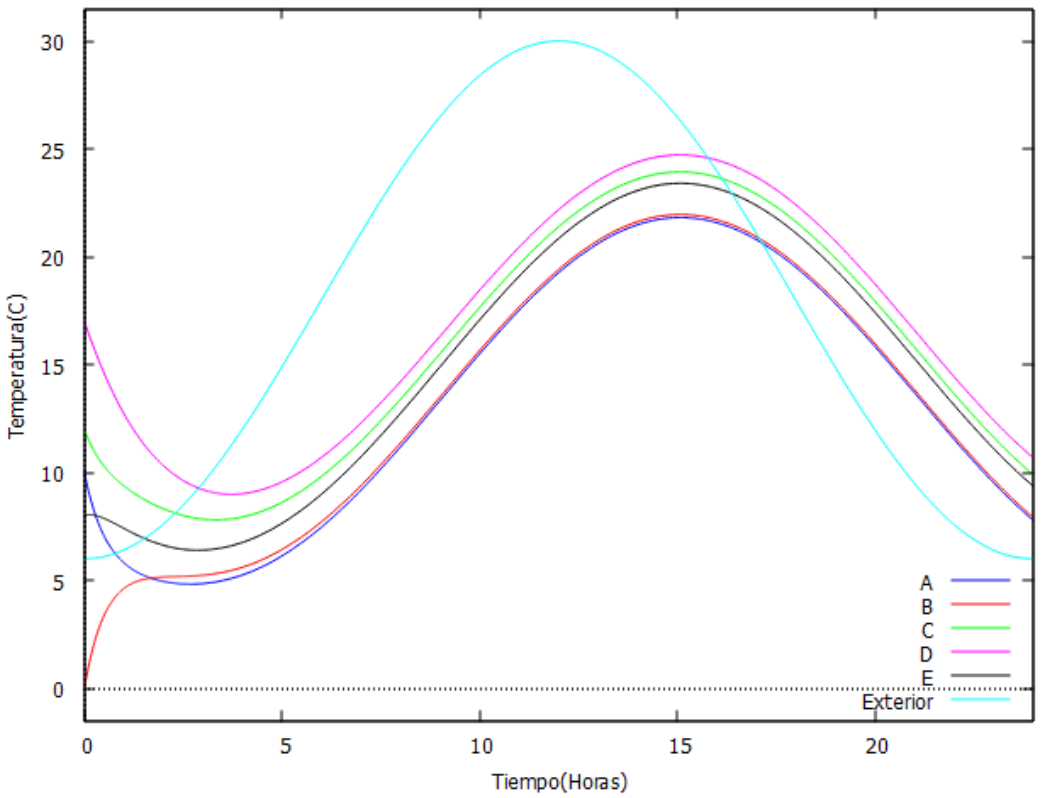

Figura 6.9.: Temperaturas de las 5 habitaciones con temperatura exterior variable

7. Arquitectura Software

7.1. Introducción

En el desarrollo de aplicaciones, la arquitectura subyacente puede definir tanto su funcionalidad como su escalabilidad. A continuación, veremos los principales tipos de arquitecturas que suelen utilizar las aplicaciones hoy en día. Mencionaremos las ventajas e inconvenientes que tienen cada una de ellas, con el fin de saber cuál es la que mejor se adapta a cada aplicación según las necesidades del proyecto.

Vamos a estudiar las arquitecturas monolíticas, cliente-servidor, basadas en microservicios, y posteriormente veremos la Arquitectura Orientada a Servicios (*Service Oriented Architectures, SOA*) junto con el modelo REST (Transferencia de Estado Representacional) y su implementación en las aplicaciones, conocida como RESTful.

7.2. Arquitectura de una aplicación

La arquitectura de una aplicación describe cómo encajan sus piezas a alto nivel. Los desarrolladores utilizan muchos tipos de arquitecturas. Muchas de estas abordan características particulares del problema que se está resolviendo.

Por ejemplo, los sistemas basados en reglas se suelen utilizar para manejar situaciones complejas, en las que resolver un problema particular se pueda reducir a seguir un conjunto de reglas. Algunos sistemas de resolución de problemas utilizan este enfoque. Algunas de las arquitecturas que veremos son:

- Monolítica. Todas las funciones de la aplicación se implementan en un solo programa, lo que puede simplificar el desarrollo pero limita la flexibilidad y escalabilidad a medida que la aplicación crece.
- La arquitectura cliente-servidor separa las funciones de la aplicación en dos partes distintas: el cliente, que es la interfaz de usuario, y el servidor, que maneja el procesamiento y el almacenamiento de datos. Esta separación permite una mayor escalabilidad y distribución de tareas, pero puede introducir complejidad adicional en la comunicación entre el cliente y el servidor.
- La arquitectura orientada a servicios (SOA) es un enfoque en el que las diferentes funciones de la aplicación se exponen como servicios independientes, lo que permite una mayor reutilización y flexibilidad en el desarrollo de software empresarial.
- Arquitectura de microservicios. No son sólo un tipo de arquitectura, sino también un modo de abordar la escritura del software. Con ellos, las aplicaciones se dividen en sus elementos más pequeños, que son independientes entre sí. Cada uno de dichos elementos o procesos es un microservicio.
- REST es un estilo arquitectónico para diseñar sistemas distribuidos que se basa en la idea de que cada recurso se identifica mediante un URI y se puede acceder y manipular

a través de operaciones estándar de HTTP como GET, POST, PUT y DELETE. Esta arquitectura es especialmente adecuada para aplicaciones web y servicios web que requieren una alta escalabilidad y rendimiento.

En resumen, cada tipo de arquitectura tiene sus propias ventajas y desventajas, y la elección de la arquitectura depende por tanto de los requisitos y las restricciones específicas de cada proyecto.

7.2.1. Arquitectura monolítica

En una arquitectura monolítica, un solo programa hace todo. Muestra la interfaz de usuario, accede a los datos, procesa cualquier petición del servidor y hace todo lo que la aplicación necesite.

Esta arquitectura tiene algunas desventajas significativas. En particular, las piezas del sistema están estrechamente vinculadas, por lo que no te ofrece mucha flexibilidad.

Una arquitectura monolítica también requiere comprender cómo encajan todas las piezas del sistema desde el principio del proyecto. Si se comete algún error en los detalles, el acoplamiento estrecho entre las piezas del sistema dificulta corregirlos más tarde. Sin embargo, las arquitecturas monolíticas tienen algunas ventajas, debido a que todo está integrado en un solo programa, no es necesario tener una comunicación complicada a través de redes. Las arquitecturas monolíticas también son útiles para aplicaciones pequeñas donde un solo programador o equipo está trabajando en el código.

Dos ejemplos de arquitecturas monolíticas muy popularizados son los ERP (Enterprise Resource Planning), son una forma de dar lógica a las enormes cantidades de datos dentro de una organización y permitir el flujo de información entre los diferentes equipos de manera automática y sin inconvenientes, y los CRM (Customer Relation Management).

La información que hemos obtenido para describir la arquitectura monolítica está basada en [Ste15].

7.2.2. Cliente-Servidor

Una arquitectura cliente-servidor separa las piezas del sistema que necesitan usar una función particular (clientes) de las partes del sistema que proporcionan esas funciones (servidores). Eso desacopla las piezas de cliente y servidor del sistema para que los desarrolladores puedan trabajar en ellas por separado. Por ejemplo, muchas aplicaciones dependen de una base de datos para mantener información, y la aplicación necesitará mostrar esa información en algún tipo de interfaz de usuario. Una forma de hacerlo sería integrar la base de datos directamente en la aplicación. Un problema con este diseño es que múltiples usuarios no pueden usar los mismos datos. Este problema se puede solucionar pasando a una arquitectura de dos niveles donde un cliente (la interfaz de usuario) está separado del servidor (la base de datos). Los clientes y el servidor se comunican a través de alguna red como una red de área local (LAN), red de área amplia (WAN) o Internet. La arquitectura de dos niveles facilita el soporte de múltiples clientes con el mismo servidor, pero vincula estrechamente a clientes y servidores. Los clientes deben saber qué formato utiliza el servidor, y si se cambia la forma en que el servidor presenta sus datos, se necesita también cambiar el cliente para que coincida. Eso puede suponer un trabajo extra, especialmente al principio de un proyecto cuando las

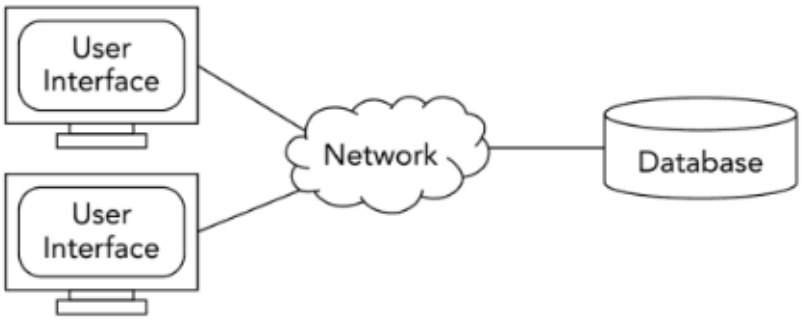

Figura 7.1.: Arquitectura de 2 niveles, donde el cliente está separado del servidor

necesidades del cliente y del servidor no están completamente definidas. Podemos ver una imagen de la arquitectura de dos niveles en la Figura 7.1.

Otra opción es aumentar la separación entre los clientes y el servidor introduciendo otra capa entre los dos para crear una arquitectura de tres niveles. En este caso, el nivel intermedio proporciona una separación entre los clientes y el servidor, la cual permite que diferentes equipos trabajen sin interferir demasiado entre sí. Además de proporcionar separación, un nivel intermedio puede realizar otras acciones que hagan que los datos sean más fáciles de usar tanto para el cliente como para el servidor.

Se pueden definir otras arquitecturas multinivel que utilicen más de tres niveles si fuera necesario. Por ejemplo, un nivel de datos podría almacenar los datos, un segundo nivel podría calcular agregaciones y realizar otros cálculos sobre los datos, un tercer nivel podría usar técnicas de inteligencia artificial para hacer recomendaciones basadas en los datos del segundo nivel, y un cuarto nivel sería de presentación que permitiría a los usuarios ver los resultados.

Algunos ejemplos de aplicaciones actuales que utilizan la arquitectura cliente-servidor son:

- *Outlook* es un cliente de correo que sirve para leer los correos del servicio *hotmail*.
- *WhatsApp* o *Telegram* son clientes de servicios de chat, video, audio...
- *Firefox* es un cliente web, *Chrome* también, son navegadores.

La información que hemos obtenido para describir la arquitectura cliente-servidor está basada en [Ste15].

7.2.3. Arquitectura Orientada a Servicios, SOA

En los últimos años, SOA ha recibido una atención creciente en línea con el movimiento hacia abordar los desafíos asociados con la mejora y mantenimiento de diversos entornos. Con el objetivo de modernizar el sistema de software de las organizaciones, la migración de un sistema heredado a un sistema basado en SOA se ha convertido en una tendencia predominante.
Son varios los beneficios de emplear SOA en el desarrollo de tecnologías líderes en el mundo

actual, como *Internet of Things* (IoT) y *Cloud Computing* (CC), además de microservicios. Esto se debe a que SOA ofrece integración flexible y reutilización de servicios debido a su arquitectura modular basada en servicios. SOA también ofrece transparencia porque encapsula varias aplicaciones y fuentes de datos en forma de caja negra. De esta manera, un conjunto integrado de recursos de Tecnologías de la Información (TI) aún podría ser accesible a pesar de la existencia de diversas tecnologías, códigos de lenguaje, funcionalidades y plataformas.

Aunque no hay una definición clara de SOA, en [NWI+20] se define como un concepto arquitectónico que promueve el acoplamiento flexible, la reutilización, la interoperabilidad, la agilidad, la eficiencia, con un enfoque en descomponer cada proceso empresarial en bloques más pequeños de tareas y funciones como servicios. Estos servicios están bien definidos y organizados y sirven como unidades independientes de funcionalidad empresarial estándar que están conectadas entre sí para crear un proceso empresarial unificado.

En [NWI+20] se indican varios estudios que afirman que la combinación de SOA con otras tecnologías podría proporcionar más beneficios para las organizaciones. Podría integrarse con nuevas tecnologías médicas, agentes inteligentes, tecnologías inalámbricas, identificación por radiofrecuencia (RFID) y procedimientos operativos de Internet para establecer nuevas automatizaciones para el proceso empresarial. Además, los requisitos de IoT podrían cumplirse mediante el enfoque SOA, ya que puede proporcionar medición de rendimiento, detección de ataques de seguridad e inteligencia empresarial.

Como un claro ejemplo del uso de SOA tenemos el de *Cisco*, la empresa adoptó SOA para lograr que su experiencia en la realización de pedidos sea coherente para todos los canales y productos. Podemos hacer referencia también al caso de la empresa *Independence Blue Cross* de Filadelfia. La misma implementó una SOA con el fin de lograr que los distintos integrantes que se encargaban de los datos de los pacientes pudieran trabajar utilizando el mismo origen de datos; una única fuente real de los mismos.

7.2.4. Microservicios

La arquitectura de microservicios funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, es un método de desarrollo de aplicaciones software. En ella, cada microservicio desempeña una función específica y se comunica con el resto a través de APIs, además cuentan con sistemas de almacenamiento propio, lo que evita la sobrecarga y caída de la aplicación.

Los microservicios se encuentran distribuidos y tienen un nivel de acoplamiento bajo, para no influir en los demás. El objetivo es distribuir un software de calidad con mayor rapidez. Puede desarrollar múltiples microservicios de forma simultánea, sin necesidad de volver a diseñar o implementar toda la aplicación después de realizar modificaciones, ya que los servicios se implementan de forma independiente.

Gracias a ello, varios desarrolladores pueden trabajar en sus servicios individuales al mismo tiempo, en lugar de actualizar toda la aplicación, lo que reduce el tiempo de desarrollo y permite lanzar características nuevas con mayor frecuencia.

Veamos qué ventajas y desventajas tiene la aplicación de una arquitectura de microservicios frente a otros tipos de arquitectura:

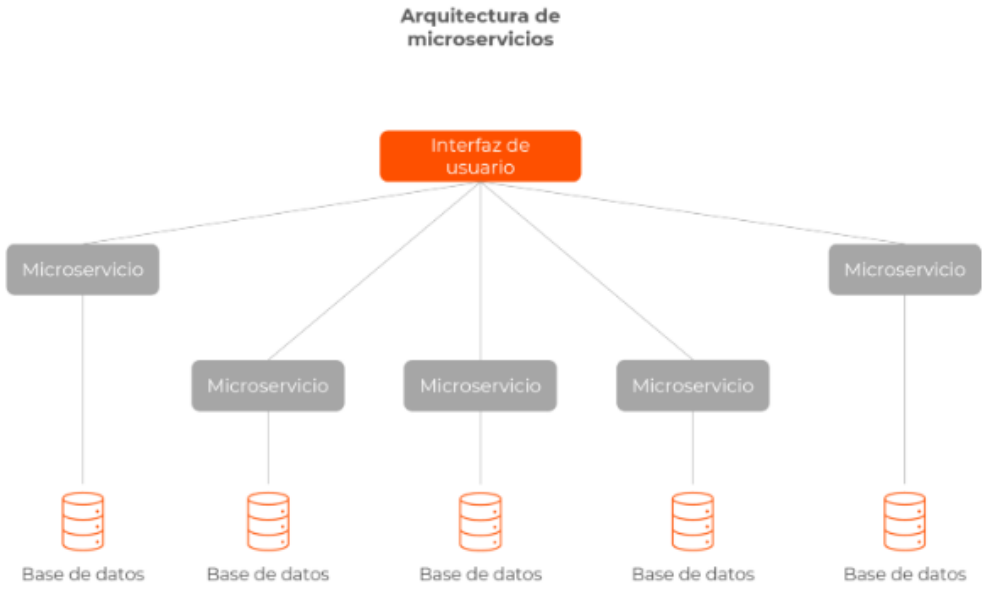

Figura 7.2.: Modelo de arquitectura basada en microservicios

7.2.4.1. Ventajas

- Modularidad: al tratarse de servicios autónomos, se pueden desarrollar y desplegar de forma independiente. Además un error en un servicio no debería afectar la capacidad de otros servicios para seguir trabajando según lo previsto.
- Escalabilidad: como es una aplicación modular, se puede escalar horizontalmente cada parte según sea necesario, aumentando el escalado de los módulos que tengan un procesamiento más intensivo.
- Versatilidad: se pueden usar diferentes tecnologías y lenguajes de programación. Lo que permite adaptar cada funcionalidad a la tecnología más adecuada y rentable.
- Mantenimiento simple y barato: al poder hacerse mejoras de un solo módulo y no tener que intervenir en toda la estructura, el mantenimiento es más sencillo y barato que en otras arquitecturas.
- Rapidez de actuación: el reducido tamaño de los microservicios permite un desarrollo menos costoso, así como el uso de Dockers permite que el despliegue de la aplicación se pueda llevar a cabo rápidamente.

7.2.4.2. Desventajas

- Coste de implantación alto: una arquitectura de microservicios puede suponer un alto coste de implantación debido a costes de infraestructura y pruebas distribuidas.

- Complejidad en la gestión: si contamos con un gran número de microservicios, será más complicado controlar la gestión e integración de los mismos. Es necesario disponer de una centralización de trazas y herramientas avanzadas de procesamiento de información que permitan tener una visión general de todos los microservicios.
- Inversión de tiempo inicial: al crear la arquitectura, se necesita más tiempo para poder fragmentar los distintos microservicios e implementar la comunicación entre ellos.
- Alto consumo de memoria: al tener cada microservicio sus propios recursos y bases de datos, consumen más memoria y CPU.

Podemos nombrar algunos ejemplos de aplicaciones actuales que utilizan este tipo de arquitectura:

- Netflix. Esta plataforma tiene una arquitectura generalizada que desde hace ya un par de años se pasó a los microservicios para el funcionamiento de sus productos.
- Amazon. No soporta tantos dispositivos como Netflix, pero tampoco es que sea fundamental para cubrir su sector. Migró hace tres años a la arquitectura de microservicios siendo una de las primeras grandes compañías que la implementaban en producción.

Tanto la Figura 7.2 como la información sobre las arquitecturas de microservicios está sacada de [mic19].

7.2.5. REST y RESTful

REST (Representational State Transfer) es un conjunto de criterios de diseño para desarrollar proyectos basados en recursos, que cambió por completo la ingeniería del software y es un concepto fundamental para el desarrollo de cualquier sistema (aplicaciones móviles, sistemas industriales, empresas, negocios, etc...). RESTful se suele utilizar para referirse a

Figura 7.3.: Ejemplo de las operaciones de transferencia de estado

los servicios web de ejecución de una arquitectura REST. El objetivo de REST es definir recursos, modificarlos e intercambiar la representación de estos en un formato concreto:

XML, JSON, HTML, texto plano, etc... Estos recursos se identifican de forma única mediante un URI *(Uniform Resource Identifier)* y son accesibles globalmente, además se construyen de forma dinámica. Estos recursos se pueden manipular a través de operaciones estándar de HTTP como pueden ser GET, POST, PUT o DELETE que sirven para leer, actualizar, crear y eliminar, respectivamente. En la Figura 7.3 podemos ver un ejemplo de uso de las distintas operaciones.

Los recursos se pueden clasificar en dos tipos principalmente:

- Entrypoint: URL de un sitio web para visualizar el contenido, están diseñados para interactuar con el usuario final.
- Endpoint: URL de una API que responde a una petición, a diferencia del entrypoint, no están diseñados para interactuar con el usuario final.

Las principales ventajas que nos ofrece el uso de REST son:

- Separación entre cliente y servidor que nos permite distinguir entre la interfaz de usuario del servidor y del almacenamiento de datos, además de mejorar la portabilidad con otros proyectos.
- Visibilidad, fiabilidad y escalabilidad gracias a la separación entre cliente y servidor añadiendo nuevos recursos.
- El desarrollo orientado a recursos es muy modular, al usar encapsulación y reusabilidad.
- La interfaz de acceso a los diferentes recursos es universal, debido al uso de HTTP, JSON/XML y ser independiente de cualquier plataforma.
- Es uno de los conceptos más ampliamente usados actualmente para el desarrollo, prácticamente cualquier sistema actual usa tecnología REST/RESTful en su backend.

Sin embargo, también existen algunas desventajas:

- Es necesario abordar problemas de una forma distinta, cambiando el modo de pensar.
- Al principio requiere un mayor tiempo de desarrollo y más conocimientos que otros modelos clásicos de servicios web.

La información acerca de la arquitectura REST está basada principalmente en [All10].

7.3. Tecnologías Backend

Hemos implementado la API utilizando Python y Flask, pero otra opción hubiera sido con Java y Spring, a continuación mostramos una comparación exhaustiva entre ambos:

Flask y Spring son ambos populares frameworks web que permiten construir aplicaciones web. Sin embargo, hay varias diferencias clave entre ellos en cuanto a su arquitectura, enfoque de desarrollo y uso.

- Estructura: Flask es un micro-framework, mientras que Spring es un framework completo. Flask permite a los desarrolladores tener más flexibilidad a la hora de elegir componentes y bibliotecas para construir su aplicación. Por otro lado, Spring viene con una estructura predefinida, proporcionando un enfoque más valorado para el desarrollo web.

- Lenguaje: Flask se usa principalmente con Python, mientras que Spring se usa comúnmente con Java. Python es conocido por su simplicidad y legibilidad, lo que hace de Flask una opción popular para proyectos pequeños a medianos. Java, por otro lado, es un lenguaje más versátil y ampliamente utilizado, lo que hace de Spring el framework preferido para aplicaciones a gran escala a nivel empresarial.
- Filosofía de Desarrollo: Flask sigue la filosofía de "Do-it-yourself", donde los desarrolladores tienen más control sobre el proceso de desarrollo. Permite a los desarrolladores elegir e integrar varios componentes según sus requisitos. Spring, por otro lado, sigue la filosofía de Çonvention over Configuration", proporcionando un conjunto de configuraciones y convenciones predefinidas para simplificar el desarrollo. Esto puede acelerar el desarrollo pero puede limitar la flexibilidad.
- Comunidad y Ecosistema: Flask tiene una comunidad más pequeña en comparación con Spring, pero es altamente activa y está creciendo rápidamente, además, cuenta con una amplia gama de extensiones y bibliotecas de terceros, lo que facilita encontrar soluciones para necesidades específicas. Spring, por otro lado, tiene una comunidad más grande y madura con una extensa documentación y soporte.
- Curva de Aprendizaje: Flask tiene una curva de aprendizaje relativamente baja debido a su simplicidad y enfoque minimalista. Es fácil comenzar con Flask y construir una aplicación básica rápidamente. Spring, por otro lado, tiene una curva de aprendizaje más pronunciada debido a sus características extensas y complejidad. Requiere una comprensión más profunda de Java y Spring.
- Adecuación: Flask es adecuado para aplicaciones pequeñas y medianas, prototipos y desarrollo rápido. Es ideal para proyectos que requieren flexibilidad y soluciones personalizadas. Spring, por otro lado, es más adecuado para aplicaciones a nivel empresarial a gran escala con requisitos complejos. Proporciona un amplio soporte para seguridad, escalabilidad e integración con otros sistemas empresariales.

En resumen, Flask y Spring difieren en cuanto a su estructura, lenguaje, filosofía de desarrollo, soporte comunitario, curva de aprendizaje y adecuación para diferentes tamaños de proyecto. Flask ofrece más flexibilidad y simplicidad, mientras que Spring proporciona un marco integral con amplias características a nivel empresarial.

7.4. Tecnologías Frontend

Para el desarrollo del Frontend de nuestro proyecto, nos hemos decidido por React, una biblioteca de JavaScript muy popular. Sin embargo, es importante mencionar que Angular también fue considerado como una opción viable debido a sus similitudes con React.

7.4.1. React vs. Angular

React se destaca por su enfoque minimalista, lo que lo hace relativamente fácil de entender para aquellos familiarizados con JavaScript. Sin embargo, configurar un proyecto en React puede llevar tiempo, y el dominio de librerías adicionales es necesario para gestionar el estado de la aplicación. Además, las actualizaciones frecuentes de React requieren un esfuerzo adicional por parte del desarrollador para mantenerse al día con las mejores prácticas y

cambios en la sintaxis.

Angular, por otro lado, presenta una curva de aprendizaje más pronunciada debido a su gran tamaño y complejidad. Aunque TypeScript, el lenguaje en el que está basado Angular, se asemeja a JavaScript, aprender todos los conceptos asociados con Angular lleva más tiempo. Además, las actualizaciones constantes de Angular requieren un esfuerzo adicional de aprendizaje por parte del desarrollador.

Si bien React cuenta con una gran comunidad de desarrolladores que respaldan su continuo desarrollo, Angular presenta cierto rechazo debido a su complejidad percibida, especialmente después de la versión 1.0, que no tuvo mucho éxito. Sin embargo, el respaldo de Google ha mejorado la credibilidad de Angular, y su uso por parte de grandes empresas como McDonald's y Apple subraya su relevancia en la industria.

React ofrece a los desarrolladores la libertad de elegir la estructura de su aplicación, lo que se convierte en una mayor flexibilidad, pero también en una mayor complejidad al inicio del proyecto. Por otro lado, Angular proporciona una estructura fija y compleja, más adecuada para desarrolladores experimentados. Su arquitectura basada en tres capas (modelo, controlador y vista), implica una mayor formalidad en la organización del código.

En resumen, la elección entre React y Angular depende de las necesidades y preferencias del proyecto, considerando factores como la curva de aprendizaje, la comunidad de soporte y la estructura deseada para la aplicación. Ambos tienen sus ventajas y desventajas, y la elección final dependerá de los objetivos y requisitos específicos de desarrollo.

8. Diseño, implementación y despliegue del simulador

8.1. Introducción

En este apartado vamos a detallar la implementación de un simulador mediante tecnología web para mostrar la aplicación y utilidad de este trabajo de una forma mucho más gráfica y dinámica, con la posibilidad de cambiar los parámetros del edificio a nuestro gusto y ver cómo afectan a las distintas temperaturas del edificio.

Veremos el proceso y desarrollo del simulador, realizando un análisis del diseño del proyecto, incluyendo los distintos tipos de requisitos junto con el proceso de verificación, para el desarrollo hemos utilizado tecnologías actuales que podremos encontrar en cualquier proyecto de desarrollo de las principales empresas hoy en día, además veremos los aspectos más importantes en el proceso de creación del backend y frontend, detallando todos los pasos que hemos seguido tanto en el diseño como en la implementación, mostraremos algunos ejemplos de casos de uso del simulador para comprobar su correcto funcionamiento comparándolo con la parte teórica, además, al final se discutirán posibles mejoras y actualizaciones que se podrían realizar sobre el proyecto a futuro.

El objetivo es que sea accesible para cualquier usuario, y para ello la interfaz será muy intuitiva y fácil de entender, además incluiremos en el Apéndice A una pequeña guía de instalación y acceso del simulador, de tal forma que cualquiera pueda utilizarlo y probarlo.

8.2. Tecnologías utilizadas

A continuación vamos a enumerar y explicar los principales motivos y características acerca de las tecnologías que hemos utilizado, incluyendo el enlace a sus páginas web oficiales. Empezamos por las que hemos requerido para desarrollar el backend:

- Python. Además de porque tiene una comunidad muy amplia, la cual nos permite encontrar feedback rápidamente sobre cualquier duda o problema que tengamos, es uno de los lenguajes más utilizados para ciencia de datos, IA y cálculo, por lo que existen muchas librerías para simplificar la implementación de nuestros sistemas de ecuaciones diferenciales. Aunque otra opción es utilizar Java o en C++, sería más complejo realizar el cálculo de las ecuaciones diferenciales además del manejo de los datos, así que por las necesidades del proyecto hemos decidido utilizar Python. Las principales bibliotecas que hemos utilizado para el cálculo matemático son Numpy y Scipy.

- Flask. Es un framework para Python muy popular utilizado comúnmente para crear aplicaciones web, y sobre todo, servicios web. Nos hemos decantado por su uso debido al hecho de ser gratuito y de código abierto, además de su sencillo uso.

- JSON. Notación estándar para el intercambio de información entre los servicios y las aplicaciones. Está basado en un subconjunto del lenguaje JavaScript, y una característi-

ca muy importante es que no depende del lenguaje que se utilice, por lo que es ideal para el intercambio de datos.

- OpenAPI. Es una notación estándar para describir y compartir con otros usuarios (desarrolladores, comunidad) nuestro servicio web (resolución de sistemas de ecuaciones diferenciales).

En cuanto al desarrollo del frontend hemos utilizado:

- React. Es una de las librerías más populares de JavaScript, está diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Tiene un alto rendimiento y permite infundir código HTML con JavaScript. Aplicaciones muy populares como Facebook, Netflix, DropBox, Instagram o Paypal usan React, y es por ello que es tan importante y nos hemos querido familiarizar con ella en este trabajo. Además, es mantenida por Facebook y la comunidad de software libre.

- Highcharts. Es una librería escrita en JavaScript para generar gráficos interactivos en aplicaciones web, soporta una gran variedad de gráficos y en nuestro caso la utilizaremos para mostrar el resultado final.

- Bootstrap. Es un framework que permite a los desarrolladores web darle forma a un sitio web y adaptarlo a las necesidades de los usuarios, nosotros lo hemos utilizado para mejorar el diseño de la página.

- Ant Design. Es una biblioteca React UI (User Interface), que como su nombre indica, permite construir interfaces de usuario con un elegante diseño, la hemos compaginado con Bootstrap para mejorar el diseño de la página.

En el diagrama de la Figura 8.1 podemos ver la arquitectura que sigue el simulador junto con los logos de las principales tecnologías que hemos utilizado tanto para el frontend como para el backend.

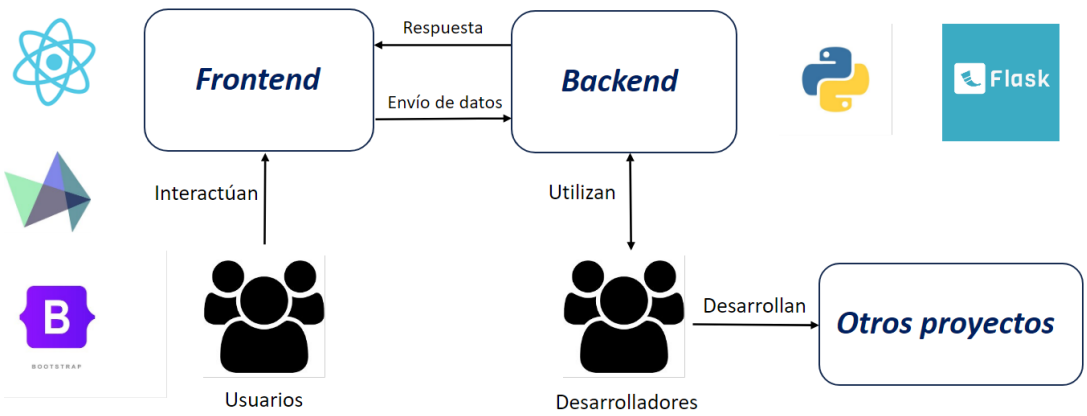

Figura 8.1.: Arquitectura del simulador

8.3. Diseño

8.3.1. Análisis de requisitos

8.3.1.1. Requisitos funcionales

Los requerimientos funcionales son aquellos que vienen establecidos por las funcionalidades del sistema, es decir, definen el comportamiento interno del software.

- **RF**1. El sistema deberá ser capaz de crear un nuevo edificio. El usuario deberá rellenar un formulario que debe ser comprobado por el software.
- **RF**2. El sistema deberá ser capaz de leer los datos del formulario y crear y resolver un sistema de ecuaciones diferenciales correspondiente, proporcionando la solución en el formato adecuado.
- **RF**3. El sistema deberá ser capaz de borrar un edificio que se esté generando para volver al estado inicial. El usuario podrá seleccionar un botón de Reset para ello.
- **RF**4. El sistema deberá poder generar un informe con los datos de las temperaturas de las habitaciones durante el día. Dicho informe se generará en formato JSON.
- **RF**5. El sistema deberá poder generar una gráfica con las temperaturas de las habitaciones durante el día además de la temperatura exterior.

8.3.1.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que imponen restricciones sobre el diseño o la implementación.

Usabilidad.

- **RN**1. La navegación por la web debe ser intuitiva y estar dotada de instrucciones para los usuarios y así facilitar la interacción de los usuarios con la interfaz.
- **RN**2. Se deberán comunicar avisos cuando, por ejemplo, se cometa un error al introducir datos en el formulario, indicando la forma correcta de introducir los datos.

Rendimiento.

- **RN**3. Se debe tener en cuenta el número de habitaciones que tenga el edificio, para no generar un sistema de ecuaciones excesivamente complejo y que dificulte la computación.
- **RN**4. El sistema ha de ser capaz de realizar todas las conexiones sin que esto repercuta en el tiempo de espera ni en el rendimiento.

Soporte.

- **RN**5. Cuando un usuario genere un edificio, debe facilitarnos toda la información necesaria para calcular las temperaturas internas. Deberemos adaptarla para que sea compatible con el formato del sistema de ecuaciones diferenciales correspondiente.

Organizativos.

- **RN**6. La API se desarrollará utilizando como lenguaje de programación Python y el framework Flask. Por otro lado, para el desarrollo del frontend se utilizará React, que permitirá aportar dinamicidad al sistema.

8.3.2. Verificación

Vamos a verificar que el sistema funciona correctamente, es decir, que cumple con los requisitos funcionales y no funcionales descritos anteriormente:

Figura 8.2.: Formulario intentando introducir 25 habitaciones

- En la Figura 8.2 podemos ver el formulario que debe rellenar el usuario para crear un nuevo edificio, cumpliendo así el primer requisito **RF**1, además aparece el botón para poder hacer Reset, de forma que podamos borrar el edificio actual y generar uno nuevo, como se indica en **RF**3, por otro lado vemos cómo tenemos avisos en el formulario que se generan de forma dinámica para ayudar al usuario a introducir los datos de forma correcta, lo que verifica los requisitos **RN**1 y **RN**2, en cuanto al apartado de rendimiento, hemos puesto un número máximo de 20 habitaciones en el edificio para no colapsar el sistema, y todos los cálculos y conexiones se hacen muy rápidamente, en cuestión de milisegundos, verificando **RN**3 y **RN**4, por último, también se verifica **RN**5, ya que como se ve en Figura 8.3, el sistema es capaz de generar el sistema a partir de los datos introducidos por el usuario.

- Una vez se han leído los datos del formulario, se crea el sistema de ecuaciones diferenciales asociado como podemos ver en la Figura 8.3, cumpliendo **RF**2.

- En la Figura 8.8, se puede ver que podemos descargar la solución en formato JSON, además de la gráfica con las temperaturas internas de las distintas habitaciones además de la externa, verificando los requisitos **RF**4 y **RF**5.
- Efectivamente, se ha utilizado Python y Flask para el desarrollo del backend, y React para el frontend, verificando así el requisito **RN**6. En la Figura 8.1 se pueden ver las principales tecnologías que se han utilizado.


```
▼ condiciones_iniciales:
    y_0: "12"
    y_1: "5"
    y_2: "0"
  ▶ [[Prototype]]: Object
▼ ecuaciones: Array(3)
    0: "(y[1] - y[0])/1+(10 - 5*np.cos(t*np.pi/12)-y[0])/3 + (-2) + 0*0.018"
    1: "(y[0] - y[1])/1+(y[2] - y[1])/1+(10 - 5*np.cos(t*np.pi/12)-y[1])/3 + 5*0.018"
    2: "(y[1] - y[2])/1+(10 - 5*np.cos(t*np.pi/12)-y[2])/3 + 2*0.018"
```


Figura 8.3.: Sistema de ecuaciones diferenciales creado a partir del edificio

8.4. Frontend

8.4.1. Idea inicial del diseño

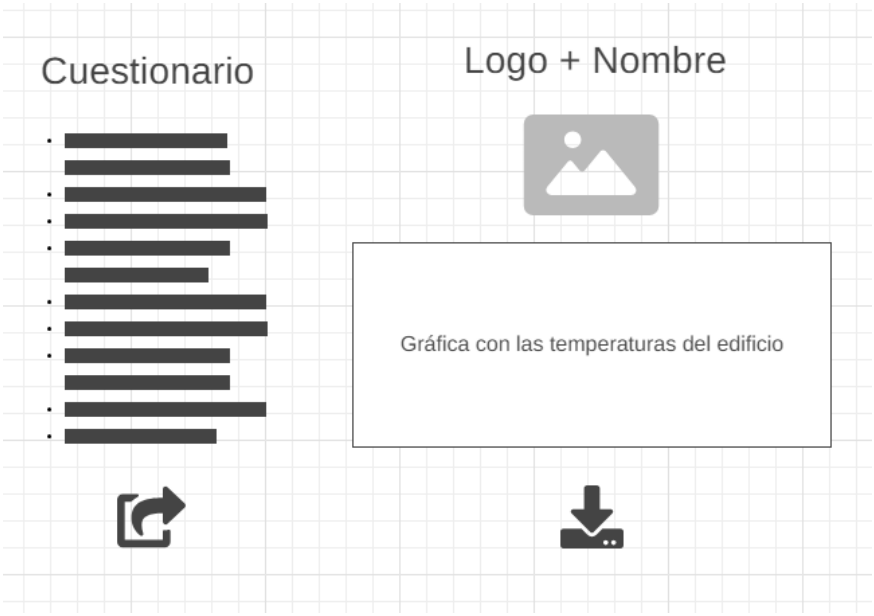

Figura 8.4.: Idea principal del diseño del simulador

En la Figura 8.4 podemos ver un wireframe del diseño propuesto para el frontend antes de realizar la implementación. Para recoger los datos y parámetros del edificio hemos decidido

utilizar un cuestionario sencillo, y justo debajo tendrá un botón para enviar los datos a la API.

En el lado derecho tendremos un logo o nombre del simulador con el objetivo de darle una seña de identidad al proyecto, y justo debajo una gráfica con la solución al problema, es decir, las temperaturas de las habitaciones internas del edificio, que se mostrarán como respuesta de la API cuando le demos al botón de enviar. Por último, un botón para descargar los resultados en formato JSON, por si algún desarrollador quiere utilizarlo en otro proyecto.

8.4.2. Desarrollo y diseño

El objetivo principal del frontend es tener una herramienta web, visual y atractiva, que mediante una serie de parámetros, nos permita obtener los valores correspondientes a la resolución de nuestros sistemas de ecuaciones diferenciales, para que cualquier tipo de usuario pueda hacer uso de esta herramienta sin tener un conocimiento experto relacionado con las matemáticas ni la informática.

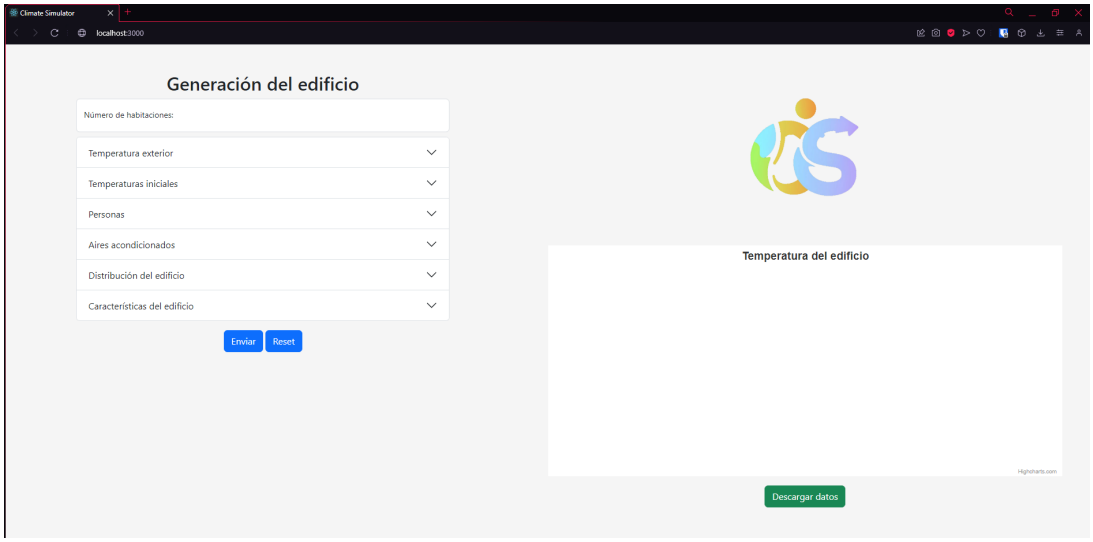

Figura 8.5.: Frontend del simulador

Nada más abrir la página nos encontramos con el aspecto que podemos ver en la Figura 8.5, donde a la izquierda tenemos un formulario dinámico que nos permite generar el edificio con todos los parámetros necesarios, y es que los campos del formulario varían en tiempo real según el número de habitaciones, esto es posible gracias a que estamos utilizando React. Para mejorar el diseño y estilo del formulario hemos utilizado tanto Ant Design como Bootstrap, con el objetivo de hacer una interfaz más agradable para el usuario. En la Figura 8.6 podemos ver cómo se incluyen 2 habitaciones en el desplegable al haber introducido que el edificio tendrá 2 habitaciones, y si cambiamos el número de habitaciones, el formulario cambia dinámicamente.

Para añadir la temperatura exterior, hemos incluido dos opciones, que podremos cambiar haciendo click en la barra superior del campo:

- Opción 1: Aquí podremos incluir directamente la función que tendrá la temperatura exterior durante las 24 horas, está pensada para usuarios expertos en el tema. La

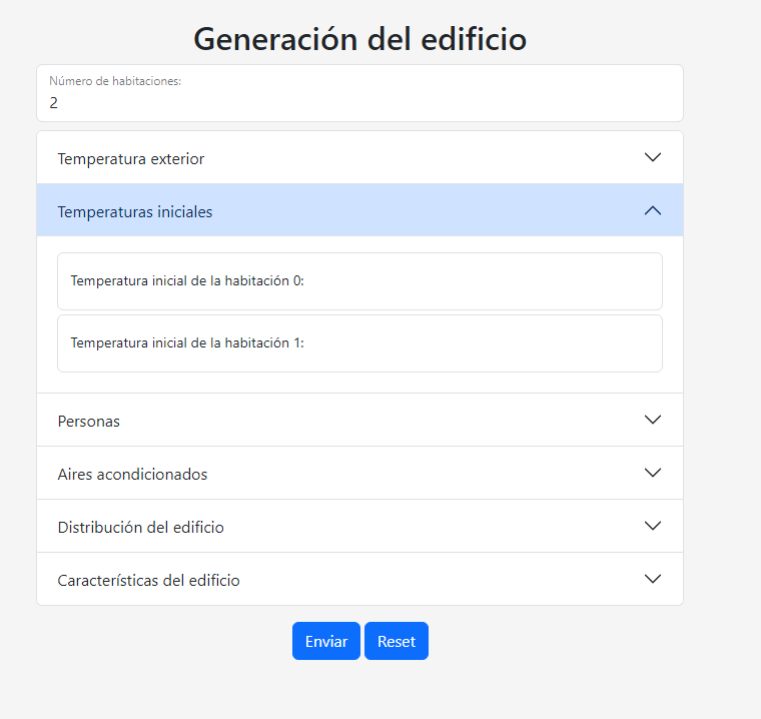

Figura 8.6.: Formulario introducir 2 habitaciones

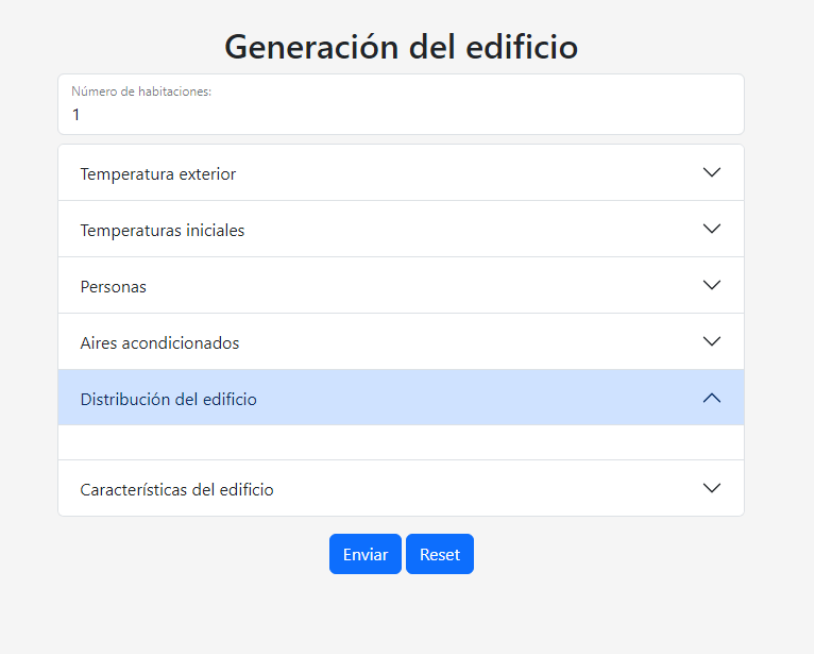

Figura 8.7.: Formulario con 1 habitación

función debe ser leída por la biblioteca *NumPy* de Python, luego tendremos que usar expresiones como por ejemplo: $5 - 10 * np.cos(t * np.pi/12)$

- Opción 2: Podremos indicar la temperatura media del día y además cuánta variación tendrá, por ejemplo, si ponemos temperatura media de 10°C, y variación de 5, entonces la temperatura variará entre 5°C – 15°C grados, esta opción está pensada para que cualquier tipo de usuario pueda introducir la temperatura externa, sin necesidad de tener un conocimiento experto.

Otro detalle a tener en cuenta es que si ponemos sólo una habitación, no nos aparecerá la distribución del edificio, ya que no tendría sentido, como podemos ver en la Figura 8.7.

Al final tenemos dos botones tanto para enviar el formulario y obtener inmediatamente la gráfica con las temperaturas a la derecha, o para resetear el formulario por si queremos empezar de nuevo a crear el edificio.

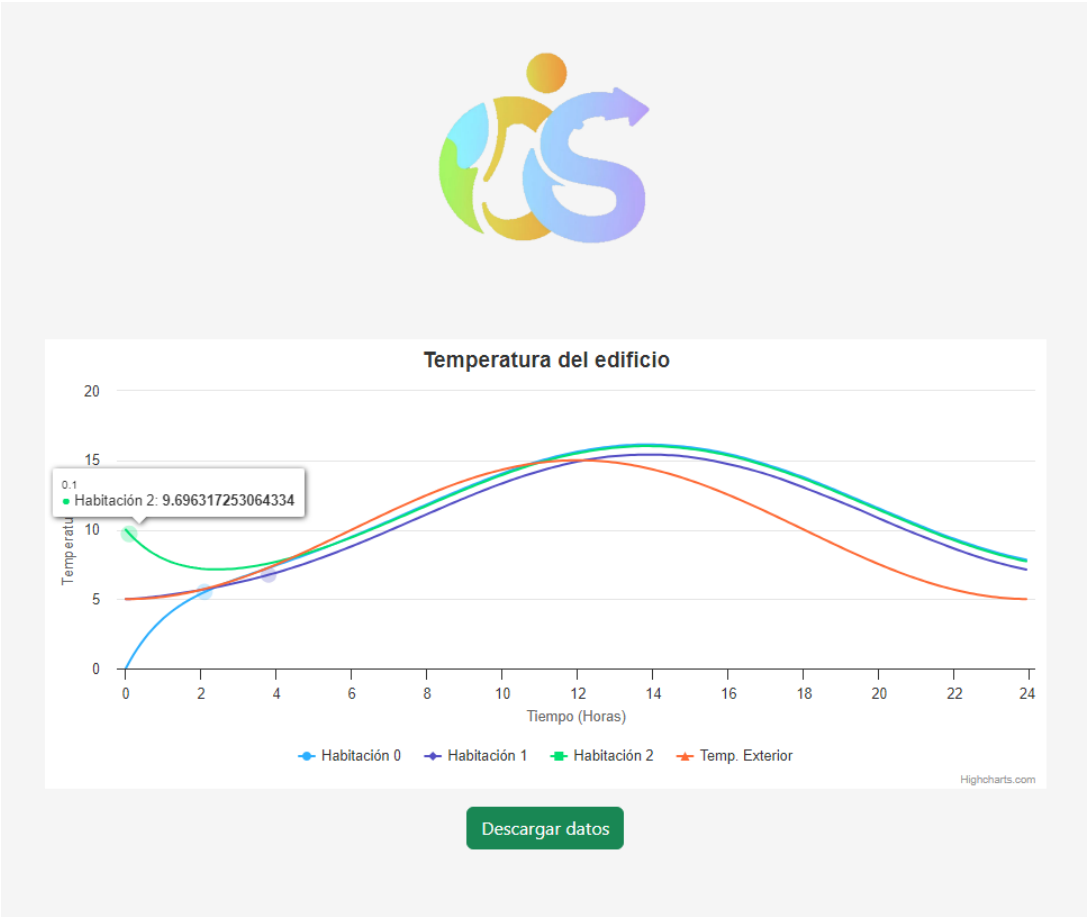

Figura 8.8.: Parte derecha de la página

Ahora pasando a la parte derecha, tenemos en primer lugar el logo que hemos decidido añadir a la página, esto le aporta una identidad al proyecto aparte de ser más atractivo, justo debajo tenemos la solución al problema, es decir, una gráfica que hemos incluido

gracias a Highcharts donde podemos ver las temperaturas de cada habitación, además de la temperatura externa para cada hora del día, y si pasamos el cursor por encima podemos ver el valor numérico de la temperatura en una hora determinada, todo esto se muestra en la Figura 8.8.

Por último, hemos incluido justo debajo un botón verde que nos permite descargar los datos de las temperaturas obtenidas en formato JSON, esta es una funcionalidad muy interesante que nos permite utilizar los resultados como posible entrada para otro programa por si fuesen necesarios.

8.5. Desarrollo del Backend

Por las ventajas ya mencionadas, para la implementación del backend de nuestra propuesta se va seguir la filosofía RESTful, ya que es la más utilizada hoy en día, esto quiere decir que nuestro servicio web implementa una arquitectura de REST, la cual sirve para estandarizar comunicaciones web entre sistemas, logrando que se entiendan mucho mejor entre ellos. Se basa en que el cliente envía peticiones para recuperar o modificar recursos, y el servidor responde con el resultado, que puede ser con los datos que hemos pedido o el estado de la petición.

Todo el desarrollo lo hemos realizado utilizando el lenguaje de programación Python, y nos hemos servido de Flask para crear el servicio web.

El objetivo principal de nuestra API es resolver sistemas de ecuaciones diferenciales, para ello recibimos del servidor web las ecuaciones, las condiciones iniciales y la función que representa la temperatura exterior del edificio, y devuelve un vector formado por:

- Un vector con los puntos que representarán el tiempo en el eje X (de 0 a 24 horas).
- Un vector en el que cada componente contiene otro vector con las temperaturas de cada habitación en cada instante de tiempo (los puntos del primer vector).
- Un último vector formado por la temperatura externa en cada instante de tiempo.

La información es devuelta por el backend como respuesta al frontend, donde se construirá la gráfica final con los resultados obtenidos.

Además, vamos a añadir el estándar de OpenAPI, que nos permite visualizar los métodos de la API para que cualquier programador o persona interesada pueda ver de forma esquemática el uso y funcionamiento detallado de cada uno de los métodos. Un detalle muy importante es que es independiente del lenguaje de programación con el que esté desarrollada la API. En la Figura 8.9 podemos ver cómo queda la documentación, la cual es accesible para todo el mundo. Podemos ver el archivo con la especificación OpenAPI completa en Apéndice B.

8.6. Casos de uso

8.6.1. Ejemplo con 3 habitaciones que vimos en la Figura 6.4

Siguiendo el modelo de edificio que tomamos en el ejemplo de la Figura 6.4, vamos a rellenar el formulario con los siguientes datos:

- Número de habitaciones: 3

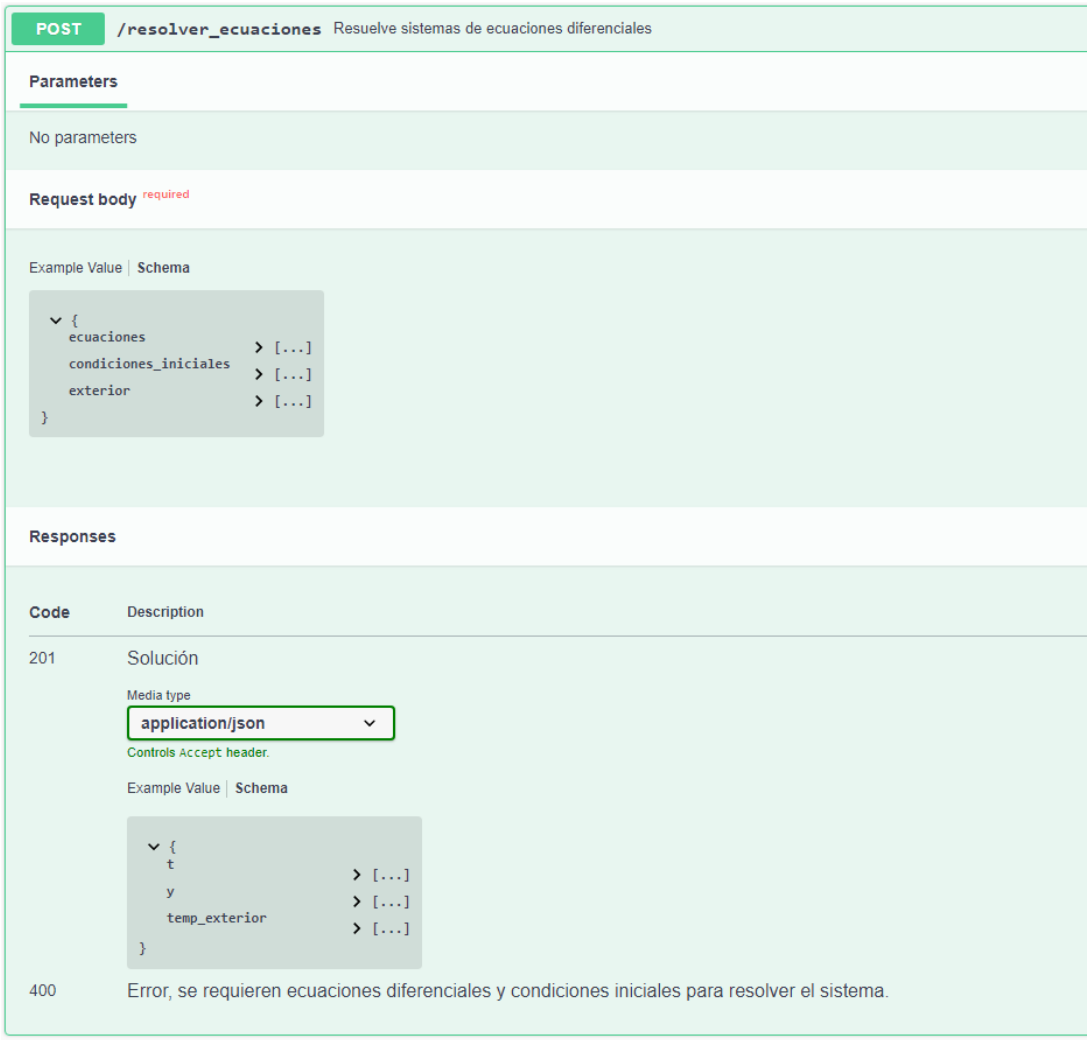

Figura 8.9.: Especificación OpenAPI

- Temperatura exterior: $5 - 10 \cdot cos(t \cdot \pi/12)$
- Temperatura inicial de las habitaciones: $20, 10, 15$
- Cantidad de personas en las habitaciones: $1, 10, 0$
- Aires acondicionados en las habitaciones 0 y 2
- Distribución del edificio:
  - Habitaciones contiguas a la habitación 0: $1, 2$
  - Habitaciones contiguas a la habitación 1: 0
  - Habitaciones contiguas a la habitación 2: 0
- Características del edificio:

- Constante de transferencia entre habitaciones: 5
- Constante de transferencia con el exterior: 3
- Calor generado por el aire acondicionado: $-8°C/5h$

Al introducir los datos en el formulario y hacer click en el botón Enviar, obtenemos la solución al problema, la cual coincide con la que vimos en Figura 6.5:

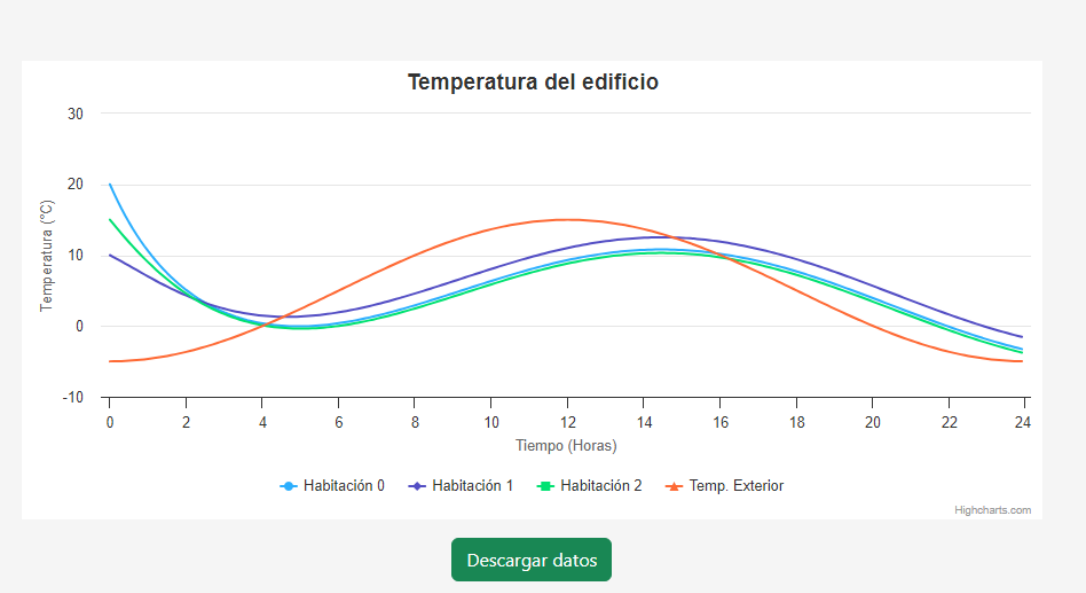

Figura 8.10.: Temperaturas internas del edificio

8.6.2. Ejemplo con 6 habitaciones

Vamos a diseñar un nuevo edificio algo más elaborado, que constará de 6 habitaciones y seguirá la estructura que podemos ver en la Figura 8.11. Vamos a introducir los siguientes datos en el formulario que corresponden con nuestro edificio:

- Número de habitaciones: 6
- Temperatura exterior: $20 - 15 \cdot cos(t \cdot \pi/12)$
- Temperatura inicial de las habitaciones: $20, 10, 15, 5, 5, 12$
- Cantidad de personas en las habitaciones: $1, 10, 0, 0, 5, 0$
- Aires acondicionados en las habitaciones 0, 2 y 5
- Distribución del edificio:
  - Habitaciones contiguas a la habitación 0: $1, 2$
  - Habitaciones contiguas a la habitación 1: $0, 3$
  - Habitaciones contiguas a la habitación 2: $0, 3, 5$

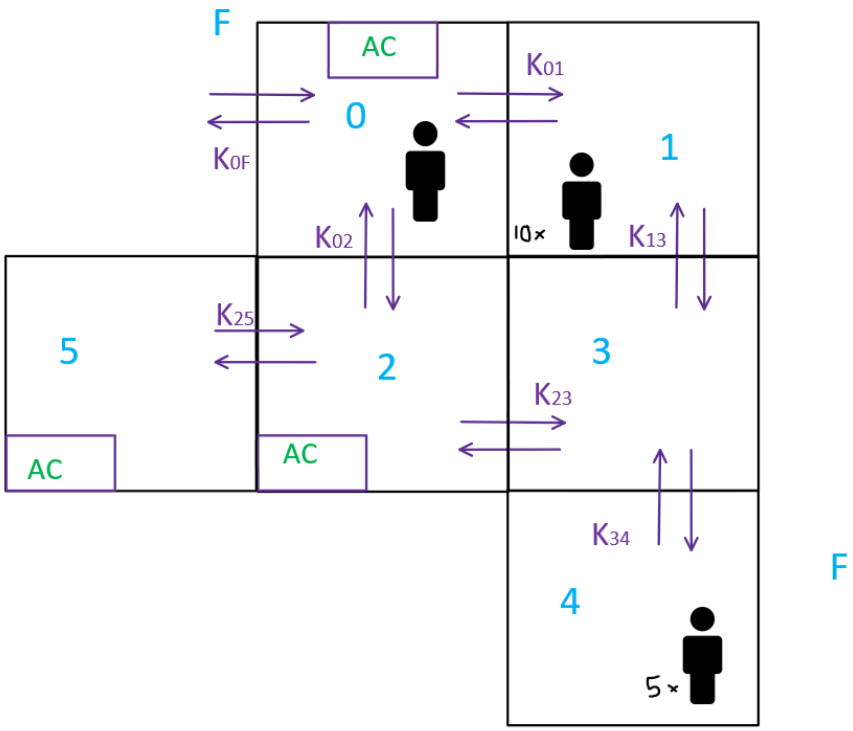

Figura 8.11.: Estructura del edificio con 6 habitaciones

- Habitaciones contiguas a la habitación 3: 1, 2, 4
- Habitaciones contiguas a la habitación 4: 3
- Habitaciones contiguas a la habitación 5: 2

- Características del edificio:
  - Constante de transferencia entre habitaciones: 3
  - Constante de transferencia con el exterior: 8
  - Calor generado por el aire acondicionado: $-4°C/h$

Al introducir los datos en el formulario y hacer click en el botón Enviar, obtenemos la solución al problema, como podemos ver en la Figura 8.12. Podemos observar que el edificio tiene un gran aislamiento con el exterior, y es por ello que las temperaturas de las habitaciones no se ven prácticamente influenciadas por la temperatura exterior, además de las altas temperaturas exteriores y tener un aire acondicionado que genera bastante frío, podríamos estar ante un escenario típico de verano.

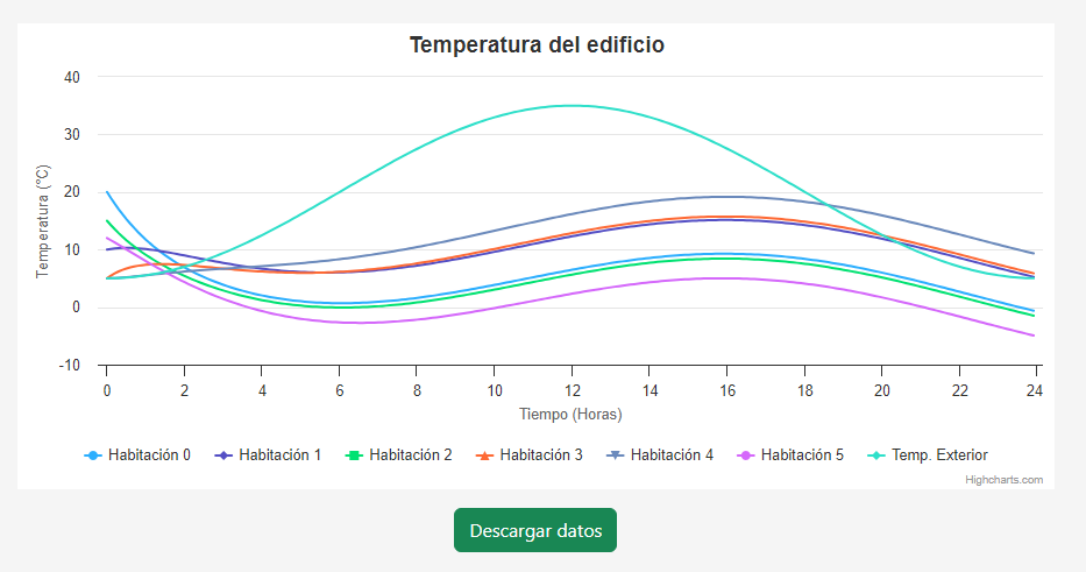

Figura 8.12.: Temperaturas internas del edificio con 6 habitaciones

9. Conclusiones y trabajo futuro

9.1. Conclusiones

El trabajo se enfoca en el desarrollo de un simulador que emplea tecnologías actuales para predecir las temperaturas internas de edificios en función de diversos factores externos. Este proyecto es dinámico gracias al uso de React en el frontend, y con el uso de Python y Flask para el desarrollo del backend, ofreciendo una solución para la simulación precisa del comportamiento térmico de estructuras edificadas. La aplicación está fundamentada en un riguroso estudio de sistemas de ecuaciones integrales junto con varios tipos de métodos para resolverlos, los cuales han sido analizados y aplicados para modelar de manera efectiva las temperaturas dentro de un edificio. En el trabajo hemos recopilado muchas técnicas para resolver ecuaciones integrales y sistemas de Volterra, además de un estudio amplio en el caso vectorial, que no suele realizarse en los libros de texto.

La motivación principal para el trabajo es crear una herramienta fundamental en el que profesionales en el ámbito de la arquitectura o la ingeniería civil puedan utilizarla sin necesidad de tener fundamentos matemáticos, además de ser útil y gráfica para estudiar el comportamiento de las distintas temperaturas que transcurren a lo largo de un día en un edificio, siendo capaces de modificar los parámetros y la estructura del edificio de forma cómoda y dinámica e ir comprobando cómo varían las temperaturas, aplicando para ello un modelo matemático y además, dar la posibilidad de que otras personas puedan utilizarlo en su beneficio.

En este trabajo se pretende diseñar, implementar y desplegar un sistema que, haciendo uso de los modelos matemáticos, permita al usuario simular la temperatura en el interior de un edificio de acuerdo a la configuración de ciertos parámetros. Además, se pretende realizar un estudio general de algunos tipos de ecuaciones integrales, así como métodos de resolución o aproximación de la solución para algunas de ellas, y la aplicación de un modelo matemático para la distribución de la temperatura en el interior de un edificio con varias estancias. Además, gracias a las simulaciones se podría analizar de manera realista la eficiencia energética de un edificio en términos de los parámetros y funciones que describen el modelo, pero este análisis se escapa del objeto del trabajo.

La parte matemática nos ha servido para sentar una base sobre las ecuaciones integrales, ser capaces de entender y aplicar varios métodos para su resolución escalar, también vectorial en el caso de Volterra, y poder aplicarlo en un contexto concreto como es el de simular las temperaturas internas de un edificio. En particular, nos hemos centrado en las ecuaciones lineales integrales de Volterra de segunda clase, estudiando y analizando la existencia y unicidad de sus soluciones. Hemos visto la relación entre las ecuaciones diferenciales y ecuaciones lineales integrales de Volterra, por último, hemos ejemplificado varios modelos de edificios y distintas formas de resolver sus sistemas asociados que nos servirán como guía en el desarrollo del simulador para resolver los sistemas de forma óptima.

En cuanto a la parte informática, la idea es crear una aplicación software que sirva como simulador y que permita representar una gran variedad de edificios de forma intuitiva para que cualquier usuario pueda hacer uso de esta herramienta, así como mostrar de forma gráfica y directa la simulación de las temperaturas en el edificio a lo largo del día.

Para ello hemos implementado una API como servicio web que nos permite resolver sistemas de ecuaciones diferenciales, usando tecnologías como Flask y Python, junto con un entorno web desarrollado con React con el que poder enviar y recibir peticiones a la API, además de mostrar todos los datos y soluciones de forma gráfica, permitiéndonos incluso descargar los resultados para poder utilizarlos con otro fin.

Es importante destacar la conexión entre las matemáticas y la informática en el desarrollo del trabajo, ya que ambas son esenciales e indispensables para el funcionamiento general del proyecto, gracias a las contribuciones que hemos realizado tanto por la parte matemática como informática, hemos conseguido cumplir con el objetivo general del trabajo, crear un simulador funcional que estudie el comportamiento de las temperaturas internas de un edificio basándose en un modelo matemático concreto.

9.2. Trabajo futuro

El simulador es plenamente funcional e ilustra la implementación y uso de los modelos matemáticos propuestos a través de un conjunto de servicios. No obstante, se plantean posibles mejoras que sin duda serían de gran utilidad para el usuario final:

- Cambiar la forma en la que se introducen los parámetros del edificio y hacer un panel en el que de forma gráfica y dinámica se vaya creando el edificio, y podamos ir añadiendo o quitando elementos tales como habitaciones, personas, etc... Esto sería una mejora en el diseño de la página además de permitir un uso más sencillo de la misma.
- Podríamos acceder a una API de tiempo meteorológico para no tener que introducir la temperatura externa a mano y obtenerla automáticamente. Además de trabajar con las próximas 24 horas, podríamos estudiar cualquier día o intervalo de tiempo en específico.
- Actualmente utilizamos una arquitectura monolítica para el desarrollo, la cual no permite mucha escalabilidad, así que podríamos migrarlo todo a microservicios y a la nube para que sea más escalable.

En cuanto a la posible extensión en el ámbito de las matemáticas podemos destacar:

- Diseñar algoritmos numéricos para la resolución de S.E.I.V (sistemas de ecuaciones integrales de Volterra), lineales o no.
- Utilizar otro modelo que involucre S.E.I.V, utilizando un desarrollo análogo resolviéndolo con un software si se trata de un sistema lineal, o con un algoritmo numérico si es no lineal, ya que los software son limitados a la hora de manejar sistemas no lineales.

A. Manual de instalación y acceso

El manual está pensado para Linux, pero la misma tecnología existe para Windows o Mac y el proceso de instalación es similar.

A.1. Requisitos previos

Antes de instalar y ejecutar el programa, debemos tener instalado lo siguiente en el sistema:

- Python (Version 3.6 en adelante)
- PIP (gestor de paquetes de Python)
- Flask, NumPy, SciPy, Flask-CORS (librerías de Python)
- npm y Node.js
- React

A.2. Instrucciones de instalación

A.2.1. Primera parte: API

1. Clonamos el repositorio del TFG desde GitHub y accedemos al de la API:

   ```
   git clone https://github.com/vaynilla/TFG.git
   cd API_TFG
   ```

2. Instalamos las dependencias de Python utilizando PIP:

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

3. Ejecutamos el servidor de la API:

   ```
   python app.py
   ```

   La API ahora debería estar en funcionamiento y accesible en la dirección URL especificada (por ejemplo, http://localhost:5000).

A.2.2. Segunda parte: Simulador

1. Accedemos al repositorio del simulador:

   ```
   cd simulador
   ```

2. Instalamos las dependencias del proyecto utilizando npm:

   ```
   npm install
   ```

3. Iniciamos la aplicación:

   ```
   npm start
   ```

A.2.3. Acceso a la aplicación

Una vez hemos completado los pasos anteriores, podremos acceder a la aplicación desde el navegador web:

- Backend (API): La API estará disponible en la URL especificada por Flask (por ejemplo, http://localhost:5000).
- Frontend (React): La aplicación frontend estará disponible en http://localhost:3000 por defecto, a menos que se especifique lo contrario.

B. Especificación OpenAPI

```yaml
openapi: "3.0.0"
info:
  title: Resuelve sistemas de ecuaciones diferenciales
  version: "1.0.0"
paths:
  /resolver_ecuaciones:
    post:
      summary: Resuelve sistemas de ecuaciones diferenciales
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                ecuaciones:
                  type: array
                  items:
                    type: string
                condiciones_iniciales:
                  type: array
                  items:
                    type: integer
                exterior:
                  type: string
      responses:
        '201':
          description: Solución
          content:
            application/json:
              schema:
                type: object
                properties:
                  t:
                    type: array
                  y:
                    type: array
                    items:
                      type: array
                  temp_exterior:
                    type: array
        '400':
          description: Error, se requieren ecuaciones diferenciales y condiciones iniciales
          para resolver el sistema.
```

Figura B.1.: Archivo openapi.yaml

Bibliografía

Las referencias se listan por orden alfabético. Aquellas referencias con más de un autor están ordenadas de acuerdo con el primer autor.

[Ado94] G. Adomian. *Solving Frontier Problems of Physics: The decomposition method*. Kluwer, 1994.

[All10] Subbu Allamaraju. *RESTful Web Servics Cookbook*. O'Reilly Media, Inc., 2010.

[AR92] G. Adomian and R. Rach. *Noise terms in decomposition series solution*. Comput. Math. Appl. 1992.

[Bre11] Brezis. *Haim Functional analysis, Sobolev spaces and partial differential equations*. Universitext. Springer, New York, 2011.

[HA09] Weimin Han and Kendall E. Atkinson. *Theoretical Numerical Analysis. A Functional Analysis Framework. 3rd Edition*. Texts in Applied Mathematics. Springer New York, NY, 2009.

[He00] J.H. He. *Variational iteration method for autonomous ordinary differential systems*. Comput. Math. Appl. 2000.

[He06] J.H. He. *Some asymptotic methods for strongly nonlinear equations*. International Journal of Modern Physics B, 2006.

[Jer99] A. Jerri. *Introduction to Integral Equations with Applications*. Wiley, New York, 1999.

[KP02] Prem K. Kythe and Pratap Puri. *Computational Methods for Linear Integral Equations*. Springer Science+, Business Media, LLC, 2002.

[Lin85] P. Linz. *Analytical and Numerical Methods for Volterra Equations*. SIAM, Philadelphia, 1985.

[mic19] Arquitectura de microservicios: qué es, ventajas y desventajas. https://decidesoluciones.es/arquitectura-de-microservicios/, septiembre 2019. Recurso online.

[NSS05] R. Kent Nagle, Edward B. Saff, and Arthur David Snider. *Ecuaciones diferenciales y problemas con valores en la frontera. Cuarta edición*. Pearson Educación, 2005.

[NWI+20] Niknejad N., Ismail W., Ghani I., Nazari B., and Bahari M. *Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation*. Information Systems, 91, 101491. 2020.

[Rev15] Fernando Revilla. Función suave pero no analítica. https://fernandorevilla.es/2015/12/28/funcion-suave-pero-no-analitica/, diciembre 2015. Recurso online.

[San23] Andrea Daniela Rincón Sandoval. *Análisis numérico de la temperatura interior de un edificio con varias estancias*. Trabajo Fin de Master en Rehabilitación Arquitectónica. Universidad de Granada, 2023.

[SM11] Beatriz Campos Sancho and Cristina Chiralt Monleón. *Ecuaciones diferenciales*. Universitat Jaume I, 2011.

[Ste15] R. Stephens. *Beginning Software Engineering*. Wiley, 2015.

[Waz97] Abdul-Majid Wazwaz. *Necessary conditions for the appearance of noise terms in decomposition solution series*. Comput. Math. Appl. 1997.

[Waz99] Abdul-Majid Wazwaz. *A reliable modification of the Adomian decomposition method*. Comput. Math. Appl. 1999.

[Waz09] Abdul-Majid Wazwaz. *Partial Differential Equations and Solitary Waves Theory*. HEP and Springer, 2009.

[Waz11] Abdul-Majid Wazwaz. *Linear and Nonlinear Integral Equations. Methods and Applications.* Springer Berlin, Heidelberg, 2011.