Estudios de técnicas de discretización en problemas de consenso distribuido

Pedro Pinedo Borobio

Trabajo de fin de grado en Matemáticas
Universidad de Zaragoza

Directores del trabajo : Juan Ignacio Montijano Torcal
y Eduardo Montijano Muñoz
14 de septiembre de 2018

Abstract

The innovation in sensors and communication technology has created a new investigation field: sensor networks. The idea comes to light with the need of setting up an appliance net with constant data capture that allows recording and storing relevant information, pass data along the different appliances and the central location where this data is stored.

One clear example, it could be the situation in which sensor array seeks to process a joint measurement of some certain of interest in the environment. This value changes depending on its location. Therefore, it is necessary to provide a communication and behavior mechanism distributely. In this point, the consensus problema arises, our aim will be that all the sensors reach the same value given different initial status.

In order to understand and analyze the consensus problem this project has required to study several concepts from graph theory and communication networks. Some of these concepts include the characterization of networks using graphs and several important matrices strongly coupled with the network topology such as the adjacency and the Laplacian matrices associated to a graph. Finally, we have seen how to characterize the consensus problem by means of a set of differential equations

$$\dot{x}_i(t) = \sum_{j \in N(i)} (x_j(t) - x_i(t)),$$

making use of all the aforementioned concepts, and we have analyzed and demonstrated the conditions under which these equations lead to a consensus value.

The difficulty of this problem will be defined by the sensor interaction, that will depend on the graph generated by the network connection . If the number of interactions between sensors is low, the convergence will be slow. On the other hand, if the number of sensors is high, the convergence of the consensus will be greater.

Due to the equations that appear in the consensus problem, the numerical methods are necessary even when the problem can be solved analytically. This is because the problem is resolved in each node but every sensor has local information of the differential equation, which means, each sensors know the information of each node is connected to.

The distributed solution to this problem is complex due to mixture of information based on local data and communications restrictions, all nodes are not connected with each other.

The usual method used to solve this problem is the Euler method

$$x_{i,k+1} = x_{i,k} + h\dot{x}_{i,k}(t) = x_{i,k} + h \sum_{j \in N(i)} (x_{j,k} - x_{i,k}), \ \ \forall i = 1, \ldots, n,$$

which is a one step method, because of its simplicity. One of the drawbacks is that the time between successive communications is small. This limitation comes from the fact that the stability interval of the method is not large. In addition, the number of communications is high. Therefore, in this assessment, a numeric method to improve these features will be studied, yet maintaining similar order features and easy implementation as with Euler method. Explicit multi-step methods are a good tool to reach our aim since the integration progresses step by step using the information from previous steps and requires only one evaluation of the vector field at each integration step. This can allow us to reduce the required number of necessary communications to solve the problem. A multistep method provides an

approximation to the solution of the differential equation y_{n+k} at the point t_{n+k} by

$$\sum_{j=0}^{k} \alpha_{k-j} y_{n-j} = \sum_{j=0}^{k} \beta_{k-j} f(t_{n-j}, y_{n-j}).$$

where α_j, β_j are the coefficients of the method, wich is usually represented by the two polynomials

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j \qquad \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j$$

A linear multistep method is convergent if and only if it is consistent (has order one)

$$\rho(1) = 0, \quad \rho'(1) = \sigma(1)$$

and it is stable, that is the roots of $\rho(\zeta)$ have modulus smaller or equal than one and the roots with modulus one are simple (root condition).

The stability polynomial of the method is defined by

$$\sum_{j=0}^{k} (\alpha_{k-j} - h\lambda\beta_{k-j}) y_{n-j}, \;\; n \geq k$$

and the stability interval is defined as the set of real values z for which the stability polynomial satisfies the root condition.

First of all, multi-step method theory has been studied. Some of the more important concepts are: the definition of numerical method to solve the problem, the required conditions to establish a certain order, domain and the absolute stability range and the criteria utilised to reach the domain of stability.

In this project, a two steps and order one numerical method has been developed. As it is a two step approach for the implementation of this method an algorithm of initiation is required, the Euler method for the discretization of the continuous problem is being used. Once the problem has been discretized and therefore resolved by means of the Routh-Hurwitz criterion, it gives necessary and sufficient conditions for the values that are in the interval of stability. It has been observed that these intervals depend on the free parameters method α_0 and β_0,

$$IEA = \begin{cases} \left(-\frac{1}{\beta_0}, 0\right) & \text{si } \beta_0 > \frac{1}{2} \; y \; \alpha_0 > -\frac{1}{2} \\ (\text{máx}\{-\frac{1}{\beta_0}, \frac{\frac{1}{2}+\alpha_0}{\frac{\beta_0}{2}-\frac{1}{4}}\}, 0) & \text{si } 0 < \beta_0 < \frac{1}{2} \; y \; \alpha_0 > -\frac{1}{2} \\ \left(\frac{\frac{1}{2}+\alpha_0}{\frac{\beta_0}{2}-\frac{1}{4}}, 0\right) & \text{si } \beta_0 < 0 \; y \; \alpha_0 > -\frac{1}{2} \end{cases}$$

This theoretical essay allows us to obtain an upper limit for the time of the successive communications.

However, apart from the time between consecutive communications, there are other types of indicators whose theoretical calculation is not essential using these methods, for example the number of communications and the convergence time based on the free parameters. To analyze these data, we opted for an empirical evaluation of the data through simulations based in the numerical methods studied. This will allow us to determine the values of α_0 and β_0 for which the multi-step method achieves consensus in fewer communications.

To test the simulations was necessary the code method implementation in a n-dimensional static sensor network created randomly in Python. In our case, the Monte Carlo experiments have been designed to study the properties of the Euler algorithm compared to the method of two steps and order one. In the simulations, 100 differents networks with 100 nodes and fix topology has been considered. For each network, the nodes have been randomly positioned generating a of $1x1$ meter square and two nodes are communicated if they are at a distance of less than 0.7 meters, and in all cases the network has been forced to be connected.

These experiments have allowed us to compare the number of communications and convergence time with respect to the Euler in the different combinations, which have been obtained by modifying the parameters that define this method.

Once the method has been run, several conclusions have been drawn and can be seen in this document. On the other hand, we have included some research lines that could be run following the essay.

Índice general

Capítulo 1

Introducción

1.1. Motivación

El desarrollo tecnológico y científico de las últimas décadas ha hecho posible que los robots y las redes de sensores se utilicen en la actualidad en un gran número de tareas. Sin embargo, para poder alcanzar todo su potencial todavía es necesario solucionar muchos retos importantes en términos de sus sistemas teóricos. Uno de los ejemplos podría ser una situación donde un conjunto de sensores buscan realizar una medición conjunta de una variable de interés del entorno, como puede ser la temperatura. En este ejemplo hay que considerar que la temperatura medida por cada sensor puede variar en relación a su localización. Por tanto, resulta necesario proporcionar al conjunto de sensores de mecanismos de comunicación y comportamiento que permitan agregar toda la información recabada de forma robusta y distribuida.

Este problema se conoce en la literatura científica como el problema de consenso distribuido. Partiendo de condiciones iniciales diferentes, el objetivo consiste en conseguir que todos los sensores obtengan el mismo valor de la cantidad de interés, bien sea por un criterio de mayoría, mínimo, máximo ó media, definido por la red. Este problema depende fuertemente de la configuración de los sensores y de los caminos que los comunican. Por ejemplo, el comportamiento de la red se verá modificado si los sensores tienen un rango sensorial unidireccional que si tienen cámaras con campo de visión limitado. En relación al número de comunicaciones entre los distintos sensores, los algoritmos se comportarán de manera distinta si el número de enlaces de comunicación es elevado ya que tendrá repercusión en el flujo de información de la red. Por ende, se podrán identificar diferencias entre una red con un gran número de enlaces de comunicación entre los sensores y aquellas que tienen un sistema de comunicación limitado, como se puede observar en la Figura 1.1.

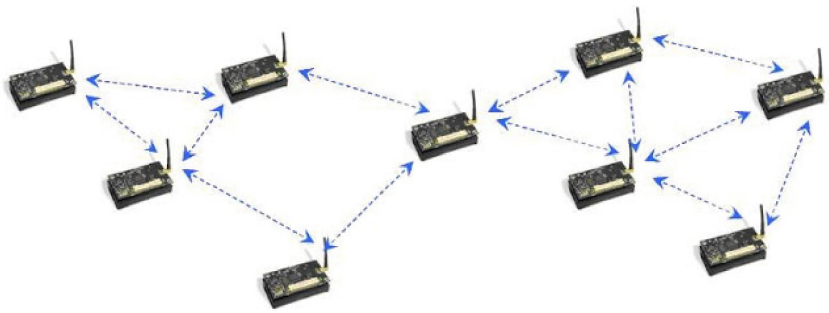

Figura 1.1: Estructura de una red inalámbrica de sensores. Se observa únicamente comunicaciones directas entre los sensores que vienen dadas por flechas bidireccionales.

Una posible solución consiste en dotar al sistema de un nodo central encargado de recibir toda la información y combinarla adecuadamente. Sin embargo, esta solución es poco robusta a fallos del nodo central, además de ser poco escalable con el número de sensores. Las soluciones puramente distribuidas

eliminan estás limitaciones, aunque el problema de consenso es más complejo. Por un lado, se requiere de estrategias eficientes de mezclado de información en base a datos locales y, por otro lado, al aumentar el número de mensajes que cada sensor envía por la red, es necesario tener en cuenta restricciones en la frecuencia con la que se pueden enviar dichos mensajes. Por tanto, resulta de interés buscar nuevas soluciones al problema que permitan al conjunto de sensores alcanzar el consenso en un número de comunicaciones menor o comunicándose cada más tiempo.

Los método de integración multi-paso se presentan como una herramienta potencialmente interesante para conseguir ambos objetivos. Esto es debido a la formulación del problema de consenso mediante un sistema de ecuaciones diferenciales [3]. Al realizarse las comunicaciones en tiempos finitos y discretos, la forma habitual de transformar el problema matemático en una solución implementable, consiste en aplicar técnicas de discretización sencillas, siendo el método de Euler la principal. La ventaja de esta técnica es su sencillez, pero presenta una gran limitación en cuanto al tiempo de paso máximo necesario para que el sistema discretizado sea convergente. Conforme aumenta el número de sensores, este tiempo se aproxima a cero muy rápidamente, lo que hace que los algoritmos no se puedan utilizar en condiciones reales. Los métodos multi-paso pueden suponer una mejora importante en este aspecto, ya que con un diseño adecuado se pueden conseguir tiempos de paso mayores, sin necesidad de afectar drásticamente a la velocidad de convergencia. En este trabajo se pretende realizar un estudio sobre la utilidad y aplicación de estos métodos al problema de consenso distribuido.

1.2. Objetivos

El objetivo principal de este trabajo es desarrollar un algoritmo de tipo numérico que resuelva el problema de consenso distribuido, reduciendo el número de comunicaciones y aumentando el tiempo entre comunicaciones. Para ello, se plantean los siguientes sub-objetivos:

- Estudiar el problema de consenso distribuido y entender su formulación básica como sistema de ecuaciones diferenciales, así como la discretización del mismo mediante el método de Euler.
- Estudiar y entender el funcionamiento de los métodos de integración multi-paso como alternativa de discretización de sistemas dinámicos.
- Aplicar dichos métodos al problema de consenso distribuido para mejorar el dominio de estabilidad absoluta del método de Euler y aumentar el paso de integración.
- Realizar una evaluación empírica sistemática de la solución propuesta mediante simulaciones.

1.3. Alcance

En este Trabajo Fin de Grado se ha realizado un estudio al Problema de Consenso distribuido en sistemas multi-robot [2] y [3]. La tesis [2] trata sobre el consenso distribuido en sistemas multi-robot con percepción visual y ha sido utilizada para extraer la idea del método de iteración lineal desarrollado. El libro [3] se ha utilizado para obtener el marco teórico del problema y la conexión con la matemática aplicada .

El estudio de los métodos multipaso se ha realizado a partir del libro [1], el que nos ha permitido obtener : la formulación del método, la convergencia, el cálculo del mínimo número de pasos para obtener la máxima convergencia, la definición de dominio e intervalo de estabilidad y los métodos para obtenerlo. Para el estudio de la estabilidad se ha utilizado [4].

Una vez realizado el estudio de los marcos teóricos, se han desarrollado nuevos métodos que optimizan el número de comunicaciones necesarias para alcanzar el consenso, maximizan el dominio de estabilidad absoluta y permitan que el tiempo entre cada comunicación sea más grande. Se trata de métodos óptimos para el tipo de problemas diferenciales que tenemos.

Por último, se ha implementado este método numérico en el entorno de Python, a partir del cual se han realizado distintas simulaciones.

Capítulo 2

Teoría de grafos y problema del consenso distribuido

En este capítulo se introducen los conceptos necesarios para comprender el propósito del TFG. En primer lugar, se formula matemáticamente el problema a resolver, introduciendo los conceptos necesarios de teoría de grafos. Por otro lado, se describen los sensores y los caminos empleados para interactuar con el resto. También se definen las comunicaciones de la red usando grafos. En segundo lugar, se revisan los algoritmos distribuidos basados en iteraciones lineales y su aplicación en el problema de consenso. En tercer lugar, y para finalizar, se muestra un ejemplo de simulación en Python para ilustrar el problema. Si se desea más información sobre los aspectos técnicos de este capítulo se puede localizar en [2].

2.1. Modelo de la red de sensores mediante un grafo

A lo largo del TFG se ha considerado la red de sensores como una red de pequeños dispositivos con capacidad limitada de medición, cálculo y comunicación. Se asume que el sistema multi-sensor está compuesto por n sensores, representados por $i \in V = \{1,\dots,n\}$.

Como consecuencia de las restricciones en las comunicaciones, no todos los sensores pueden recibir o transmitir información a todos los demás. Estas limitaciones pueden ser modeladas utilizando un grafo G=$\{V,E\}$, donde $E \subset V \times V$ contiene las parejas de sensores que pueden comunicarse directamente.

Definición. Diremos que hay una comunicación fija entre i y j si pueden intercambiar mensajes directamente. Esto ocurre si y sólo si $(i,j) \in E$. Se definen los vecinos de un sensor $i \in V$ como el conjunto de sensores que puede comunicarse directamente con i,

$$N_i = \{j \in V | (i,j) \in E\}.$$

A lo largo del TFG se trabaja con grafos no dirigidos. Esto quiere decir que si el sensor i está comunicado con el sensor j entonces j también lo está con i,

$$(i,j) \in E \Leftrightarrow (j,i) \in E \text{ y } j \in N_i \Leftrightarrow i \in N_j.$$

Definición. Dado un grafo no dirigido G, un recorrido de longitud l desde el vértice i al j es una sucesión de vértices $(i_0,i_1,i_2,\dots,i_l)$ tal que $i_0 = i$, $i_l = j$, e $(i_{h-1},i_h) \in E$ para $h = 1,2,\dots,l$. Un camino es un recorrido en el que no aparecen aristas repetidas. La distancia de dicho camino, definida como el número de aristas del mismo, se denotará por d_{ij}.

Definición. Un grafo G se dice conexo si para todo par de sensores i y j existe un camino de comunicaciones fijas que empieza en i y termina en j.

Definición. Dado un grafo G, se define su diámetro como la máxima longitud de los caminos , medida en enlaces de comunicación, entre dos sensores cualesquiera del grafo. El diámetro de G es denotado por d_v y en el caso de redes estáticas es siempre más pequeño que n, se denota por $d_v = \max\min d_{ij}$.

El diámetro de la red se ve modificado por el número de comunicaciones directas del grafo. En la Figura 2.1 se ejemplifica estas diferencias.

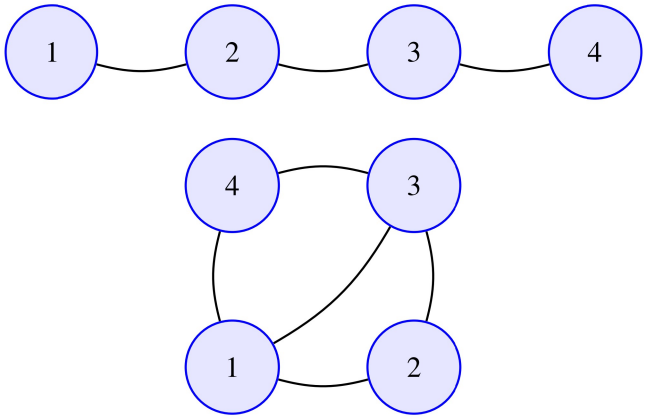

Figura 2.1: Grafos con 4 sensores y distinta topología definida por el grafo. En el segundo caso el diámetro del grafo es inferior al del primero.

Definición. La matriz laplaciana, L, que define G, viene dada por

$$(L(G))_{ij} = \begin{cases} -1 & (i,j) \in E \\ |N_i| & i = j \\ 0 \text{ } en\ otro\ caso \end{cases} .$$

En el caso de un grafo conexo y no dirigido, se trata de una matriz simétrica y semidefinida positiva. Todos los valores propios de la matriz laplaciana cumplen que $0 = \lambda_1(G) < \lambda_2(G) \leqslant \cdots \leqslant \lambda_n(G) \leq n$.

2.2. Consenso distribuido

En esta sección se define el problema del consenso y se presenta una solución al mismo usando iteraciones lineales distribuidas.

Al hablar de consenso es imprescindible considerar que los dispositivos perciben alguna cantidad de interés con sus sensores. En este trabajo se ha considerado que dicha cantidad es un número escalar por simplicidad. Además, se asume que cada sensor está dotado de un estado inicial $x_i(0) = x_{i,0}$, $\forall i = 1,\ldots,n$, que es la medida de interés del nodo.

Definición. (El problema del consenso). Dadas condiciones iniciales $x_{i,0}$, $i = 1,\ldots,n$, se define el problema del consenso como el problema de hacer que el estado de todos los sensores alcance el mismo valor de la cantidad de interés, computado como una función, f, de las observaciones iniciales : $x_i = x_j = f(x_{1,0}, x_{2,0}, ..., x_{n,0})$, $\forall i, j \in V$.

La única función del consenso que se va a utilizar en este TFG es la media generada por las condiciones iniciales, $f(\mathbf{x_0}) = \overline{\mathbf{x_0}} = \frac{1}{n}\sum_{i\in V} x_{i,0}$. Esta medida es relevante cuando las observaciones de los sensores tienen la misma importancia.

En el trabajo se han estudiado las soluciones al problema basadas en esquemas de iteración lineal. Esto es debido a que son muy fáciles de implementar, ya que sólo requieren calcular combinaciones lineales de diferentes cantidades, y son susceptibles de una implementación puramente distribuida.

El algoritmo de iteración lineal en derivadas está relacionado con el *ratio de cambio* en el flujo de información de cada sensor. Éste viene determinado por la suma de los estados relativos, respecto al estado del subconjunto generado por los vecinos. Por lo tanto, la variación del estado en el nodo i está dada por el estado de los nodos vecinos $N(i)$. En concreto, este algoritmo considera como evolución del estado de cada nodo la siguiente ecuación diferencial,

$$\dot{x}_i(t) = \sum_{j \in N(i)} (x_j(t) - x_i(t)). \tag{2.1}$$

Las ventajas de utilizar iteración lineal (2.1) son las siguientes:

- Se trata de un algoritmo totalmente distribuido porque el cambio de estado de cada sensor se obtiene usando la información que poseen los vecinos.
- Los sensores no necesitan conocer la topología de toda la red para ejecutar este algoritmo, sólo se requiere información local.
- El gasto computacional es muy bajo debido a que solo se tiene combinaciones lineales de los datos. Además, cada sensor tiene que almacenar solamente un dato, en lugar de tantos datos como número de sensores definen la red.

A continuación se procede a estudiar el conjunto de soluciones que se alcanzan resolviendo el sistema de ecuaciones diferenciales descrito en (2.1). Si se considera el ratio de cambio de todos los sensores dados en (2.1) se puede modelar poniendo las interacciones en forma matricial. Se pasa a estudiar el siguiente sistema dinámico

$$\dot{\mathbf{x}}(t) = -L(G)\mathbf{x}(t), \tag{2.2}$$

donde $\mathbf{x}(t) = \begin{pmatrix} x_1(t) & \cdots & x_n(t) \end{pmatrix}^T \in \mathbb{R}^n$, siendo $x_i(t)$ el estado asociado al nodo i, $i = 1,...,n$ y $L(G)$ es la matriz laplaciana generada por las interacciones de los sensores que definen la red.

Al tener $L(G)$ un único valor propio igual a cero y el resto positivos, podemos afirmar que el sistema dinámico definido en (2.2) converge a un estado estacionario. Si se considera $U = \begin{pmatrix} \mathbf{u_1} & \mathbf{u_2} & \cdots & \mathbf{u_n} \end{pmatrix}$ la matriz formada por los vectores propios de $L(G)$, siendo $\mathbf{u_i}$ el vector propio asociado al valor propio λ_i. La condición de equilibrio se cumple cuando

$$\dot{x}_i(t) = \sum_{j \in N(i)} (x_j(t) - x_i(t)) = 0, \forall i = 1, \ldots, n, \tag{2.3}$$

lo que se satisface si $x_i = x_j$ para todo i, j, o, lo que es lo mismo, el estado estacionario del sistema se corresponde con un consenso de los valores de todos los nodos. Adicionalmente, de aquí se deduce que $\mathbf{u_1} = \mathbf{1}$.

2.2.1. Discretización del método

Al considerar un escenario en el que los nodos realizan comunicaciones con sus vecinos, para poder implementar la iteración anterior en un sistema real resulta necesario considerar una versión discreta del método. Además, se trata de un problema distribuido y las comunicaciones son locales, y un nodo no conoce todas las componentes de la matriz Laplaciana. Por lo tanto, no se puede obtener el estado de cada sensor resolviendo analíticamente la ecuación diferencial. El estado de cada sensor después de cada comunicación en un tiempo h se puede obtener discretizando (2.1) con el método de Euler,

$$x_{i,k+1} = x_{i,k} + h \sum_{j \in N(i)} (x_{j,k} - x_{i,k}), \ \ \forall i = 1, \ldots, n. \tag{2.4}$$

De manera análoga, definiendo el conjunto de estados (2.4) simultáneamente, el estado de cada sensor se puede modelar como un sistema dinámico lineal discreto n-dimensional

$$\mathbf{x}_{k+1} = (I - hL(G))\mathbf{x}_k \tag{2.5}$$

Los valores propios de la matriz $I - hL(G)$ son de la forma $1 - h\lambda_i$, $\forall i = 1,\ldots,n$ y satisfacen : $1 < 1 - h\lambda_2 \leqslant \cdots \leqslant 1 - h\lambda_n$. Los vectores propios asociados a los valores propios $1 - h\lambda_i$, $\forall i = 1,\ldots,n$ son los mismos que los de la matriz laplaciana. La condición inicial, $\mathbf{x}(0)$, se puede expresar mediante una combinación lineal de los vectores propios de $L(G)$,

$$\mathbf{x_0} = \mathbf{x}(0) = \gamma_1 \mathbf{u_1} + \ldots + \gamma_n \mathbf{u_n}, \tag{2.6}$$

donde $\gamma_i \in \mathbb{R}$, $\forall i = 1,\ldots,n$. Con la singularidad de que el coeficiente γ_1 es igual al promedio de las condiciones iniciales, $\gamma_1 = \overline{\mathbf{x_0}}$.

Combinando y desarrollando (2.5) y (2.6) se obtiene

$$\mathbf{x}_k = (I - hL)^k \mathbf{x}_0 = \gamma_1 \mathbf{u_1} + (1 - h\lambda_2)^k \gamma_2 \mathbf{u_2} + \ldots + (1 - h\lambda_n)^k \gamma_n \mathbf{u_n}.$$

Se observa que en este caso el requisito para obtener convergencia es que $|1 - h\lambda_i| < 1$, $\forall i$.

Aquí aparece una limitación fundamental de la discretización mediante el método de Euler. Para conseguir que $|1 - h\lambda_i| < 1$ se necesita que h sea suficientemente pequeño. En concreto, por las propiedades de los valores propios de $L(G)$, una cota superior de h es

$$h < \frac{2}{\text{máx}\,|\lambda_j|}. \tag{2.7}$$

Claramente, esta cota es decreciente con el número de nodos, n, ya que al aumentar n aumenta el mayor de los valores propios, lo que hace que cuanto mayor sea la red, mayor sea la frecuencia con la que los nodos se deben comunicar para garantizar convergencia al consenso. Esto puede convertirse en un problema si consideramos que en un sistema real, el tiempo entre comunicaciones sucesivas estará acotada inferiormente por $\overline{h}$, debido a limitaciones tecnológicas. Por otro lado, la velocidad de convergencia será menor cuanto mayor sea el tamaño de la red. Ésto nos lleva a estudiar otros métodos que permitan pasos de integración, y por tanto tiempo entre comunicaciones, más altos y, si es posible, velocidades de convergencia mayores.

Ejemplo 1. A continuación se muestra un ejemplo sencillo con 3 nodos para entender de una forma más aplicada el problema de consenso y su relación con los sistemas dinámicos. Supongamos que se tiene tres sensores ubicados de manera estratégica y se quiere medir la temperatura media. Cada uno de los sensores se encarga de medir la temperatura, siendo $x_{i,0}$, $i = 1,2,3$ la temperatura inicial en cada zona y se impone que todas ellas sean distintas. Supongamos que la configuración de los sensores definen la topología del grafo de la Figura 2.2.

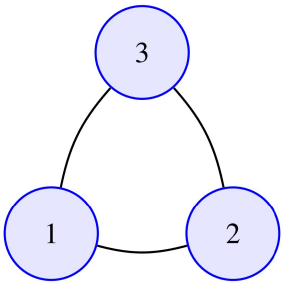

Figura 2.2: Red estática generada por los tres sensores, cada uno de ellos se encarga de medir la temperatura local de cada zona.

A partir de la topología del grafo se obtiene la matriz laplaciana

$$L(G) = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix},$$

, todos los elementos extra-diagonales no positivos, con valores propios $\lambda_1 = 0$, $\lambda_{2,3} = 3$, y vectores propios,

$$\mathbf{u_1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{u_2} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \mathbf{u_3} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}.$$

La matriz del sistema discretizado (2.5) mediante Euler es

$$\begin{pmatrix} x_{1,k+1} \\ x_{2,k+1} \\ x_{3,k+1} \end{pmatrix} = \begin{pmatrix} 1-2h & h & h \\ h & 1-2h & h \\ h & h & 1-2h \end{pmatrix} \begin{pmatrix} x_{1,k} \\ x_{2,k} \\ x_{3,k} \end{pmatrix}.$$

Cualquier condición inicial $\mathbf{x_0}$ se puede expresar como combinación lineal de los vectores propios $\mathbf{u_1}$, $\mathbf{u_2}$, $\mathbf{u_3}$. Por tanto, se obtiene,

$$\mathbf{x_k} = \gamma_1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + (1-3h)^k \gamma_2 \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} + (1-3h)^k \gamma_3 \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}. \tag{2.8}$$

Claramente, si $h < \frac{2}{3}$, $\mathbf{x_k} \to \gamma_1 \mathbf{u_1}$ y el sistema converge al consenso. Si $h = \frac{1}{3}$, el sistema converge en una única iteración. Si por el contrario $h \geqslant \frac{2}{3}$ no hay convergencia, lo que quiere decir que el tiempo entre comunicaciones consecutivas no puede ser mayor que $\frac{2}{3}$.

La limitación $|1 - h\lambda_i| < 1$ coincide precisamente con la condicion de que los productos $|h\lambda_i|$ estén en el dominio de estabilidad del método de Euler. Si se quiere aumentar el tiempo entre comunicaciones, se necesitará discretizar con un método que tenga un dominio de estabilidad mayor que el del metodo de Euler, pero que a su vez mantenga propiedades similares de orden y la sencillez de implementación práctica. En este trabajo se va a considerar métodos lineales multipaso.

Capítulo 3

Métodos lineales multipaso

Los métodos numéricos para la resolución de ecuaciones diferenciales son necesarios cuando el problema no puede resolverse analíticamente, lo cual es lo más frecuente en los problemas prácticos. En el caso de las ecuaciones que aparecen en los problemas de consenso los métodos numéricos son necesarios incluso en el caso de que la solución del problema pueda resolverse analíticamente. Esto se debe a que el problema se tendría que resolver por cada nodo, pero cada sensor tiene una información local de la ecuación diferencial, es decir, cada sensor conoce la información de los sensores con lo que tiene definidas comunicaciones directas y por lo tanto solo conoce una parte del campo vectorial. Por lo tanto no se puede obtener la solución del problema debido a que se trata de un problema de sistemas distribuidos. Por otra parte, como el nodo i-ésimo es capaz de evaluar la componente i-ésima de la función derivada, algunos métodos numéricos sí que pueden ser utilizados de manera distribuida. El método habitualmente utilizado en el campo del consenso es el método de Euler explícito, que es el más sencillo que se puede encontrar, por su sencillez y facilidad de uso. Otros métodos, como los métodos lineales multipaso o los de tipo Runge-Kutta también pueden aplicarse de forma distribuida y pueden tener mejores propiedades que el método de Euler. En este trabajo se han planteado el uso de los métodos lineales multipaso explícitos para la resolución de los problemas de consenso, ya que requieren, en cada paso de integración, una única evaluación de la función derivada de la ecuación diferencial, lo que se traduce en una única comunicación entre los nodos de la red. En este capítulo se estudiarán los conceptos y resultados que harán falta para el desarrollo de métodos adecuados para resolver los problemas de consenso.

3.1. Formulación de los métodos lineales multipaso

En esta sección se estudia la resolución numérica a través de los métodos lineales multipaso del problema de valor inicial

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(0) = y_0 \end{cases} \tag{3.1}$$

donde $f\colon \quad [0,T] \times \mathbb{R}^s \longrightarrow \quad \mathbb{R}^s$ continua y verifica la condición de Lipschitz respecto a la variable y.

En los métodos multipaso, la integración avanza, paso a paso, utilizando la información sobre la solución de varios pasos previos. Esta información consiste en los valores de la solución y la derivada en los puntos de la red. El algoritmo permite determinar $y_n \simeq y(t_n)$ en función de y_j, $f_j = f(t_j, y_j)$, $j = 0, ..., n-1$ aproximaciones a la solución (3.1) y a la derivada en los puntos t_j de una red en el intervalo $[0,T]$.

Para simplificar se considera que la red $\{t_j\}_0^N$ es uniforme con paso fijo h, es decir, $t_j = jh, \quad j = 0, \cdots, N$ y $Nh = T$. Además, se limita el estudio a los *métodos lineales multipaso con coeficientes constantes* en los que el algoritmo que define y_n es lineal respecto y_{n-j}, f_{n-j} $(j = 0, \cdots, k)$. Se obtiene

por la fórmula

$$\sum_{j=0}^{k} \alpha_{k-j} y_{n-j} = \sum_{j=0}^{k} \beta_{k-j} f(t_{n-j}, y_{n-j}). \tag{3.2}$$

donde α_j, β_j son coeficientes constantes (independientes del paso h) con $\alpha_k \neq 0$ y $\alpha_0 \beta_0 \neq 0$, que caracterizan el método.

En el Problema de Valor Inicial (3.1) sólo se conoce el valor de la solución en un punto inicial. En consecuencia, un método de k pasos ($k > 1$) no se puede aplicar directamente, por tanto es necesario emplear un método de un paso para calcular y_j, $y'_j = f(t_j, y_j)$, aproximaciones de la solución y su derivada en t_j, $j = 1, \cdots, k-1$. Dicho algoritmo es llamado de *iniciación* y estará implícito en todo método de k pasos, $\forall n = 0, \cdots, k-1$:

$$\begin{cases} y_n = \eta_n \\ y'_n = f(t_n, y_n) \end{cases} \tag{3.3}$$

Nota 1.

- Cuando los coeficientes α_j, β_j de la fórmula (3.2) se multiplican por una constante real no nula se obtiene el mismo método multipaso. Esta indeterminación se evita introduciendo alguna condición normalizadora. En nuestro caso se considera

$$\sum_{j=0}^{k} \beta_j = 1$$

- Si se denota con E al operador desplazamiento ($E y_n = y_{n+1}$) y se introduce los llamados polinomios característicos

$$\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j, \sigma(\zeta) = \sum_{j=0}^{k} \beta_j \zeta^j$$

del método (3.3), las ecuaciones (3.2) se pueden escribir

$$\rho(E) y_{n-k} = h \sigma(E) f_{n-k}, ; \ n = k, \cdots, N.$$

3.2. Orden de consistencia

Se considera el operador diferencial $L = L[y(t); h]$ llamado *operador error local*

$$L[y(t); h] = \sum_{j=0}^{k} [\alpha_{k-j} y(t - jh) - \beta_{k-j} y'(t - jh)] \tag{3.4}$$

Definición. El operador lineal (3.4) y el método multipaso correspondiente (ρ, σ) se dice de orden p, si p es el máximo entero positivo tal que $L[y(t); h] = O(h^{p+1})$ $(h \to 0^+)$, $\forall y \in C^{p+1}[0, T]$.

Escribiendo el desarrollo de Taylor de $y(t - jh)$ e $y'(t - jh)$ en un punto t_1, el Operador error local (3.4) viene dado por

$$L[y(t); h] = \sum_{r=0}^{\infty} \frac{h^r}{r!} \left\{ \sum_{j=0}^{k} [\alpha_j \tau_{k-j}^r - r \beta_j \tau_{k-j}^{r-1}] \right\} y^{(r)}(t_1) = C_0 y(t_1) + C_1 h y'(t_1) + C_2 \frac{h^2 y''(t_1)}{2!} + \cdots$$

siendo $\tau_j h = -t_1 + t - jh$ y $C_p = \sum_{j=0}^{k} [j^p \alpha_j - p j^{p-1} \beta_j]$ cuando $t_1 = t - kh$ y $\tau_j = k - j$ se tiene así el siguiente teorema.

Teorema 3.2.1. *El método es de orden p si y sólo si*

$$C_0 = C_1 = \cdots = C_p = 0 \, y \, C_{p+1} \neq 0$$

Demostración. La demostración se recoge en [1]. □

La *constante del error* del método multipaso viene dada por $\frac{C_{p+1}}{(p+1)!}$.

Definición. Un método se llama consistente si y sólo si su orden es ≥ 1, es decir, $C_0 = C_1 = 0$, lo que puede escribirse, en términos de los polinomios característicos, como $\rho(1) = 0, \rho'(1) = \sigma(1)$.

3.3. Estabilidad y convergencia.

La estabilidad de un método multipaso estudia la acotación de las soluciones numéricas y_n cuando el paso de integración h tiende a cero. El concepto de estabilidad está relacionado con el comportamiento de las soluciones frente a perturbaciones, de manera que si las ecuaciones del método son modificadas por perturbaciones pequeñas, se podrá asegurar que la modificación en la solución también será pequeña (información más detallada sobre el concepto de estabildad puede encontrarse en [4]).

En un método lineal multipaso, cuando $h \to 0$ las ecuaciones se reducen a

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + \cdots + \alpha_0 y_n = 0 \tag{3.5}$$

y se dirá que es estable cuando la solución y_n esté acotada cuando $n \to \infty$ para toda condición inicial. Notemos que estas ecuaciones se obtienen también cuando el método se aplica al problema

$$y' = 0.$$

Se trata de estudiar las condiciones sobre los coeficientes α_i, $\forall i = 0, \ldots, k$ que garanticen la acotación de la solución y_n. Para responder a esta cuestión, se usará el siguiente Lema 3.3.1.

Lema 3.3.1. *Sea $\zeta_1, \cdots, \zeta_l$ las raices de $\rho(\zeta)$,respectivamente con multiplicidad $m_1, \cdots, m_l$.Entonces la solución general de (3.5) está dada por*

$$y_n = p_1(n)\zeta_1^n + \cdots + p_l(n)\zeta_l^n \tag{3.6}$$

donde $p_j(n)$ es un polinomio de grado $m_j - 1$.

Demostración. La demostración se recoge en [4]. □

La fórmula (3.6) muestra que para la acotación de y_n ($n \to \infty$) se necesita que las raíces de $\rho(\zeta)$ se encuentren en el disco unidad y las de módulo 1 sean simples.

Teorema 3.3.2. *Un método lineal multipaso es estable si y sólo si el polinomio $\rho(\zeta)$ verifica la "Condición de las Raíces ",es decir,*

$$si\ \rho(\zeta) = 0 \Rightarrow \begin{cases} |\zeta| < 1 \\ o\ bien \\ |\zeta| = 1\ y\ \zeta\ raiz\ simple \end{cases}$$

Demostración. La demostración se recoge en [4]. □

Por otra parte, el estudio de la convergencia tiene por objeto comparar la solución exacta de (3.1) con la numérica calculada por el método (3.2).

Definición. El método (ρ, σ) se dice *convergente* si para toda función $f \in C_L$ y solución $y(t)$ de la ecuación (3.1) se verifica

$$\lim_{h \to 0^+} max_{k \le n \le N = [T/h]} \mid y_n - y(t_n) \mid = 0$$

siempre que los valores de iniciación η_n se satisfagan

$$\lim_{h \to 0^+} \mid y_n - \eta_n \mid = 0, n = 0, \cdots, k-1$$

Teorema 3.3.3. *Un método lineal multipaso es convergente si y sólo si es consistente y estable.Es decir, si verifica $\rho(1) = 0$, $\rho'(1) = \sigma(1)$ y $\rho(\zeta)$ cumple la condición de las raíces.*

Demostración. La demostración se recoge en [1]. □

3.3.1. Convergencia:Orden máximo alcanzable.

Una vez concretada la definición de convergencia, es interesante estudiar el mínimo número de pasos para obtener el máximo orden posible por un método lineal de k pasos.

Lema 3.3.4. *Sea $\psi(\zeta)$ la función*

$$\psi(\zeta) = \frac{\rho(\zeta)}{\log(\zeta)} - \sigma(\zeta)$$

El operador error local L asociado al método (ρ, σ) es de orden p sí y sólo si $\psi(\zeta)$ tiene en $\zeta = 1$ un cero de orden p.

Demostración. La demostración se recoge en [4]. □

Se considera las transformaciones complejas

$$\zeta = \frac{1+x}{1-x} \quad x = \frac{1+\zeta}{1-\zeta}$$

que lleva el interior del círculo unidad del ζ-plano en el semiplano negativo del x-plano y los puntos de la circunferencia unidad sobre el eje imaginario.

Pasando a la variable x, la condición de las raíces se traduce en que las raíces de $\rho(\zeta) = \rho(\frac{1+x}{1-x})$ quedan situadas en el semiplano negativo y las imaginarias puras son simples.

Se define en función de $\rho(\zeta)$, $\sigma(\zeta)$ y $\psi(\zeta)$

$$R(x) = \left(\frac{1-x}{2}\right)^k \rho\left(\frac{1+x}{1-x}\right) \tag{3.7}$$

$$S(x) = \left(\frac{1-x}{2}\right)^k \sigma\left(\frac{1+x}{1-x}\right) \tag{3.8}$$

$$F(x) = \left(\frac{1-x}{2}\right)^k \psi\left(\frac{1+x}{1-x}\right) = \frac{R(x)}{log\left(\frac{1+x}{1-x}\right)} - S(x)$$

siendo R y S polinomios en x de grado $\leq k$.

Lema 3.3.5. *Si el método es estable y consistente entonces*

$$R(x) = a_1 x + a_2 x^2 + \cdots + a_k x^k$$

donde $a_1 \neq 0$ y $a_i a_1 \geq 0$, $i = 2, 3, \cdots, k$.

Demostración. La demostración se recoge en [4]. □

Lema 3.3.6. *Si el método es consistente y estable,el desarrollo de Taylor*

$$\frac{R(x)}{log\left(\frac{1+x}{1-x}\right)} = \sum_{n\geq 0} r_n x^n$$

verifica:
-si k es impar$\Rightarrow r_{k+1} = 0$.
-si k es par $\Rightarrow r_{k+2} = 0$. Además, $r_{k+1} = 0 \Longleftrightarrow R(x)$ sólo contiene potencias impares de x.

Demostración. La demostración se recoge en [4]. □

Dada la información extraída de los lemas anteriores, se obtiene información suficiente para dar el siguiente teorema.

Teorema 3.3.7. ***(Dahlquist).*** *El orden máximo alcanzable por un método lineal consistente y estable es*
k+1 cuando k es impar
k+2 cuando k es par

Demostración. La demostración se recoge en [4]. □

A cada raíz esencial, raíz simple de módulo 1, del polinomio ρ se le asocia una magnitud llamada *factor de crecimiento* dado por

$$\lambda_i = \frac{\sigma(\zeta_i)}{\zeta_i \rho'(\zeta_i)}$$

Definición. Un método (ρ, σ) se dice fuertemente 0-estable si es 0-estable y los factores de crecimiento asociados a las raíces esenciales de ρ son estrictamente positivos.

3.4. Dominio de Estabilidad Absoluta y algunos resultados clásicos.

En los epígrafes anteriores se han descrito la formulación de los métodos multipaso y, partiendo de estos, se ha definido la estabilidad y convergencia. En este apartado se define el dominio de estabilidad absoluta para los métodos que se han formulado.

En primer lugar, se expone una breve introducción a las ecuaciones en diferencias lineales con coeficientes constantes.

Definición. Sea I un conjunto de enteros consecutivos. Se llama *ecuación en diferencias* de orden k a un conjunto de relaciones del tipo

$$F_n(y_n, y_{n+1}, \cdots, y_{n+k}) = 0, \forall n \in I$$

estando F_n definida en dominios apropiados de manera que son resolubles en el valor y_{n+k}.

En nuestro caso, se asume que las funciones F_n son lineales con coeficientes independientes de n. Se considerarán ecuaciones de la forma

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + \cdots + \alpha_0 y_n = d_{n+k}, \forall n = 0, 1, \cdots \tag{3.9}$$

con $\alpha_0 \alpha_k \neq 0$. Esto indica que (3.9) es exactamente de orden k. Además, se puede escribir como $\rho(E) y_n = d_{n+k}$ siendo $\rho(\zeta) = \sum_{j=0}^{k} \alpha_j \zeta^j$ su polinomio característico.

3.4.1. Definiciones básicas.

Supongamos un método lineal de k pasos (ρ,σ) (3.2) que se aplica a la ecuación escalar de prueba

$$y' = \lambda y, \lambda \in C \tag{3.10}$$

con paso fijo h. Se obtiene la ecuación en diferencias

$$\sum_{j=0}^{k} (\alpha_{k-j} - h\lambda \beta_{k-j}) y_{n-j} = 0, \;\; n \geq k \tag{3.11}$$

Sería deseable que las soluciones numéricas calculadas mediante (3.11) presentaran un comportamiento asintótico similar al de las soluciones analíticas correspondientes a la ecuación diferencial (3.10). Puesto que $\forall \lambda \in \mathbb{C}$ con $Re\lambda \leq 0$ las soluciones de (3.10) son estables cualesquiera que sean las condiciones iniciales, se debería verificar que para $Re(h\lambda) \leq 0$,las sucesiones $\{y_n\}_{n=0}^{\infty}$ solución de (3.11) sean estables. Esta propiedad no es satisfecha por muchos métodos y se tiene que calcular los valores de $z = h\lambda$ para los que las soluciones de (3.11) son estables.

Definición. Se llama *dominio de estabilidad absoluta* D del método lineal (ρ,σ) al conjunto de puntos z del plano complejo para los que las ecuaciones

$$\begin{cases} y_n = \eta_n, \ \ n = 0,1,\ldots,k-1 \\ \sum_{j=0}^{k}(\alpha_{k-j} - z\beta_{k-j})y_{n-j} = 0, \ \ n \geq k \end{cases} \tag{3.12}$$

tienen una única solución verificando

$$\sup_{n\geq 0}|y_n| \leq K \max_{0\leq n\leq k-1}|\eta_n|$$

para una cierta constante K independiente de las condiciones iniciales.

Nota 2.

- Si z es tal que $\alpha_k - z\beta_k = 0$, la ecuación en diferencias (3.11), llamada polinomio de estabilidad ,no tiene solución única y por tanto $z = \frac{\alpha_k}{\beta_k}$ no puede pertenecer a D.
- Sea
$$\pi(\zeta;z) = \rho(\zeta) - z\sigma(\zeta)$$
el polinomio característico asociado a la ecuación en diferencias (3.12) y $\zeta_i(z)$, i=1,...,k sus raíces. De acuerdo con el Lema 3.3.1, las soluciones de (3.11) estarán acotadas si el polinomio de estabilidad cumple la Condición de la Raíces. Esto nos lleva a la siguiente definición equivalente del Dominio de Estabilidad Absoluta
$$D = \{z \in \mathbb{C} | \pi(\zeta;z) \ verifica \ C.R \ y \ tiene \ grado \ exacto \ k\} \tag{3.13}$$
- $\infty \in D$ si y sólo si el polinomio σ verifica la C.R y tiene grado exacto k, debido a que $\frac{1}{z}\rho(\zeta) - \sigma(\zeta) = 0$.
- Si se consideran sistemas lineales $y' = Ay$ con A matriz constante diagonalizable, se verifica que las soluciones numéricas están acotadas si y sólo si $h\lambda_j \in D$ para todo valor propio λ_j de A.
- Para z=0 , $\pi(\zeta;0) = \rho(\zeta)$. La Condición de las raíces de $\rho(\zeta)$ se llamará *0-estabilidad*. El método es 0-estable si y sólo si $0 \in D$.
- Si el problema diferencial es lineal **ẏ= A y** con matriz **A** simétrica, los autovalores de **A** serán reales y por lo tanto todos los productos $h\lambda_i$ serán reales. En consecuencia es importante estudiar la restricción del dominio de estabilidad al eje real. Se define el intervalo de estabilidad absoluta como
$$D = \{z \in \mathbb{R} | \pi(\zeta;z) \ verifica \ C.R \ y \ tiene \ grado \ exacto \ k\} \tag{3.14}$$

Propiedad 1. El interior de D está dado por

$$D^\circ = \{z \in \overline{C} | \ |\zeta_i(z)| < 1 \ \ , i = 1,\ldots,k\}$$

Demostración. La demostración se recoge en [4]. □

Consecuencias

1. D° es el conjunto de puntos $z \in \overline{C}$ tales que toda sucesión (y_n) que verifica $\pi(E;z)y_n \to 0$ cuando $n \to \infty$.

2. Si el método (ρ,σ) es 0-estable y consistente,el origen pertenece a D y no a D°. Este punto se encuentra en la frontera del dominio de estabilidad absoluta.

A la hora de separar los dominios de estabilidad e inestabilidad juegan un papel importante los valores de z para los cuales el polinomio $\pi(\zeta;z)$ tiene alguna raíz de módulo 1. Se define un nuevo conjunto Γ que se llamará Boundary Locus, dado paramétricamente por:

$$\Gamma = \left\{ \frac{\rho(e^{i\theta}))}{\sigma(e^{i\theta})} \ ; \ 0 \leq \theta < 2\pi \right\} \tag{3.15}$$

Propiedad 2. El interior de D está formado por la unión de un número finito de componentes conexas de $\overline{C}\backslash\Gamma$.

Nota 3. De acuerdo con la propiedad anterior, para determinar el interior del dominio de estabilidad absoluta basta con obtener la curva paramétrica Γ, bien sea de forma analítica o numérica. En segundo lugar, para saber si una componente conexa Ω, determinada por Γ, pertenece o no a D° es suficiente viendo si se verifica la C.R en un punto cualquiera de ésta.

3.4.2. Criterios para verificar la condición de las raíces

Criterio de Routh Hurwitz: Dado un polinomio de grado k

$$P(z) = \sum_{j=0}^{k} a_j z^{k-j} = a_0 z^k + a_1 z^{k-1} + \ldots + a_k$$

con coeficientes reales y $a_0 > 0$, el criterio de Routh-Hurwitz da condiciones necesarias y suficientes para que todas las raíces de $P(z)$ estén contenidas en el semiplano negativo, $Re(z) < 0$. Estas condiciones se corresponden con que todos los menores principales de la k-matriz

$$Q = \begin{pmatrix} a_1 & a_3 & a_5 & \ldots & \ldots \\ a_0 & a_2 & a_4 & \ldots & \ldots \\ 0 & a_1 & a_3 & a_5 & \ldots \\ 0 & a_0 & a_2 & a_4 & \ldots \\ 0 & 0 & a_1 & a_3 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

sean estrictamente positivos.

Ejemplo 2. 1.Si k=2, $P(z) = a_0 z^2 + a_1 z + a_2$. Las condiciones de Routh-Hurwitz nos dan que $a_0 > 0$, $a_1 > 0$, $a_2 > 0$.

2.Si k=3,se ha de cumplir que $a_0 > 0,\ a_1 > 0,\ a_2 > 0,\ a_3 > 0$ y $a_1 a_2 - a_0 a_3 > 0$.

Para aplicar el criterio de R-H se considera la aplicación de Möbius

$$w = \frac{\zeta + 1}{\zeta - 1} \ \ o \ \ \zeta = \frac{w+1}{w-1}$$

que transforma

-el disco abierto unidad $|\zeta| < 1$ en el semiplano $\mathrm{Re}\, w < 0$.

-la circunferencia unidad $|\zeta| = 1$ en el eje imaginario.

-$\zeta = \infty$ en $w = 1$.

-$\zeta = 1$ en $w = \infty$.

Si se denota por

$$p(w;z) = R(w) - zS(w) \tag{3.16}$$

siendo R y S polinomios definidos en (3.7) y (3.8) respectivamente, la C.R para el polinomio $\pi(\zeta;z)$ equivale a decir que las raíces de $p(w;z)$, $w_i(z)$, verifican:

$$\mathrm{Re}\, w_i(z) \leq 0,\ \ i = 1, \ldots, k$$

$$Si\ \mathrm{Re}\, w_i(z) = 0 \Rightarrow w_i(z)\ es\ raiz\ simple$$

Un polinomio con estas condiciones se denomina de Hurwitz. Si los coeficientes son reales, el criterio de los menores principales de Q sirve para reconocerlo.

Capítulo 4

Estudio de técnicas de discretización en problemas de consenso distribuido

Uno de los objetivos de este trabajo es encontrar métodos numéricos que permitan resolver las ecuaciones de los problemas de consenso (en nuestro caso las ecuaciones (2.2)) de forma distribuida buscando además que el número de comunicaciones en la red sea mínimo y que admitan un tiempo entre dos comunicaciones sucesivas grande, lo que en términos del método significa que el paso de integración pueda ser grande. Como el problema diferencial es lineal, el paso de integración está limitado por la estabilidad y debe ser tal que para todos los valores propios de la matriz Laplaciana λ_i, los productos $h\lambda_i$ han de estar contenidos en el intervalo de estabilidad absoluta. Cuanto mayor sea, menor será la restricción en el tamaño del paso de integración. Nos proponemos diseñar una familia de métodos que dependa de uno o varios parámetros libres. Se tratará entonces de buscar las condiciones que deben cumplir los parámetros libres para que el método tenga un orden dado y cuyo intervalo de estabilidad absoluta sea lo mayor posible.

En los problemas de consenso, el método habitualmente usado es el método de Euler explícito

$$x_{i,k+1} = x_{i,k} + hx'_{i,k}(t) = x_{i,k} + h\sum_{j\in N(i)}(x_{j,k} - x_{i,k}),\ \ \forall i = 1,\ldots,n,$$

, que tiene orden uno. El intervalo de estabilidad absoluta para el método de Euler es $(-2,0)$. Por lo tanto el paso de integración h estará limitado por la condición

$$h < \frac{2}{\text{máx}\,|\lambda_j|}, \forall j = 1,\ldots,n. \tag{4.1}$$

lo que implica que el tiempo requerido entre dos comunicaciones consecutivas entre sensores no puede ser superior a este valor del paso de integración. Si alguno de los valores propios λ_i es grande, el paso máximo será muy pequeño, lo que puede imponer una restricción muy severa en la red de sensores. Nuestro primer objetivo será diseñar métodos multipaso explícitos de dos pasos y orden uno

$$\sum_{j=0}^{2}(\alpha_{2-j} - z\beta_{2-j})y_{n-j} = 0. \tag{4.2}$$

y cuyo intervalo de estabilidad absoluta sea mayor que el del método de Euler. De esta forma, con este método la integración podrá avanzar con mayor paso de integración, lo que en términos del problema de consenso significa que el tiempo de espera entre dos comunicaciones consecutivas podrá ser mayor.

Considerando los polinomios característicos $\rho(\zeta) = \alpha_2\zeta^2 + \alpha_1\zeta + \alpha_0$, $\sigma(\zeta) = \beta_2\zeta^2 + \beta_1\zeta + \beta_0$ que definen el método de dos pasos se tiene que $\pi(\zeta;z) = \rho(\zeta) - z\sigma(\zeta)$. Solo se van a utilizar métodos explícitos, por consiguiente, $\beta_2 = 0$.

Por otra parte, se va a considerar un método de orden 1. *Las condiciones de orden 1* son $\rho(1) = 0$ y $\rho'(1) = \sigma(1)$. Aplicándolas a (4.2) :

$$\alpha_2 + \alpha_1 + \alpha_0 = 0$$
$$2\alpha_2 + \alpha_1 = \beta_1 + \beta_0$$

Se va a imponer la *condición normalizadora* para obtener la unicidad del método. Si no se impone, cuando se multiplican todos los coeficientes α_j, β_j de (4.2) por una misma constante real no nula se obtiene el mismo método.

$$\beta_1 + \beta_0 = 1$$

Se tiene un sistema de tres ecuaciones con 5 incógnitas. Supongamos que los parámetros libres son $\alpha_0, \beta_0 \in \mathbb{R}$. Poniendo el resto de valores en función de ambos,

$$\beta_1 = 1 - \beta_0$$
$$\alpha_2 = 1 + \alpha_0$$
$$\alpha_1 = -1 - 2\alpha_0$$

Para finalizar se impone que el método sea 0 estable, es decir, los valores de ζ para los que

$$\rho(\zeta) = (1+\alpha_0)\zeta^2 + (-1-2\alpha_0)\zeta + \alpha_0 = 0$$

deben cumplir la condición de las raíces. Estas vienen dadas por

$$\zeta = 1 \ , \ \zeta = \frac{\alpha_0}{1+\alpha_0}. \tag{4.3}$$

De (4.3) se concluye que $\alpha_0 > -\frac{1}{2}$.

Para el cálculo del intervalo de estabilidad se aplica la transformación de Möbius $\frac{\zeta+1}{\zeta-1}$ y se calcula el polinomio $p(w;z) = R(w) - zS(w)$, siendo

$$R(w) = \frac{1}{2}w + \frac{1}{2} + \alpha_0$$
$$S(w) = \frac{1}{4}w^2 - \frac{\beta_0}{2}w + \frac{\beta_0}{2} - \frac{1}{4}$$

Por lo tanto,

$$p(w;z) = R(W) - zS(w) = -\frac{z}{4}w^2 + \left(\frac{1}{2} + \frac{\beta_0}{2}z\right)w + \left(\frac{1}{2} + \alpha_0 - z\left(\frac{\beta_0}{2} - \frac{1}{4}\right)\right)$$

Aplicando el criterio de Routh-Hurwitz a la matriz

$$Q = \begin{pmatrix} \frac{1}{2} + \frac{\beta_0}{2}z & 0 \\ \frac{1}{2} + \alpha_0 - z(\frac{\beta_0}{2} - \frac{1}{4}) & -\frac{z}{4} \end{pmatrix}$$

definida a partir de los coeficientes del polinomio $p(w,z)$, se encuentran las condiciones necesarias y suficientes para que z este en el interior del intervalo de estabilidad absoluta:

$$-\frac{z}{4} > 0, \ \frac{1}{2} + \frac{\beta_0}{2}z > 0, \ \frac{1}{2} + \alpha_0 - z\left(\frac{\beta_0}{2} - \frac{1}{4}\right) > 0.$$

Se ha de cumplir que $z < 0$ (extraído por la definición), $1 + \beta_0 z > 0 \Longleftrightarrow \beta_0 z > -1$. (El caso $\beta_0 = 0$ se estudiará por separado). Cuando $\beta_0 \neq 0$

$$\begin{cases} z > -1/\beta_0, \ Si \ \beta_0 > 0 \\ z < -1/\beta_0, \ Si \ \beta_0 < 0 \end{cases}$$

De la tercera inecuación, necesariamente $\frac{1}{2}+\alpha_0-z\left(\frac{\beta_0}{2}-\frac{1}{4}\right)>0 \Longleftrightarrow \frac{1}{2}+\alpha_0-z>\left(\frac{\beta_0}{2}-\frac{1}{4}\right)$.

Dominio de estabilidad absoluto según los valores de α_0, β_0

$$IEA=\begin{cases}\left(-\frac{1}{\beta_0},0\right) & \text{si } \beta_0>\frac{1}{2} \text{ y } \alpha_0>-\frac{1}{2}\\ \left(\text{máx}\{-\frac{1}{\beta_0},\frac{\frac{1}{2}+\alpha_0}{\frac{\beta_0}{2}-\frac{1}{4}}\},0\right) & \text{si } 0<\beta_0<\frac{1}{2} \text{ y } \alpha_0>-\frac{1}{2}\\ \left(\frac{\frac{1}{2}+\alpha_0}{\frac{\beta_0}{2}-\frac{1}{4}},0\right) & \text{si } \beta_0<0 \text{ y } \alpha_0>-\frac{1}{2}\end{cases}$$

Respecto al primer caso, se obtiene que el máximo intervalo de estabilidad absoluta es $(-2,0)$. Por lo tanto, no se observa mejoría respecto al método de Euler; Por ello, no resulta de interés.

El mayor intervalo de estabilidad en el segundo caso se alcanza cuando $\beta_0 \to 0^+$ y $\alpha_0 \to \infty$ y en el último caso cuando $\beta_0 \to 0^-$ y $\alpha_0 \to \infty$. Por lo tanto, se trata de un método en el que para dichos valores de α_0 y β_0, el intervalo de estabilidad es infinito. Esto nos permitiría coger el valor de h tan grande como se quiera. Se puede observar que el segundo caso es un caso particular del tercero, por consiguiente el estudio posterior se centrará en los valores del intervalo de estabilidad $\left(\frac{\frac{1}{2}+\alpha_0}{\frac{\beta_0}{2}-\frac{1}{4}},0\right)$.

Si $\beta_0=0$ ha de cumplirse $\frac{1}{2}+\alpha_0+\frac{z}{4}<0$ y $z<0$,es decir, $z>-2-4\alpha_0$ y $z<0$. El intervalo de estabilidad absoluta viene dado por $(-2-4\alpha_0,0)$, es decir, es tan grande como se quiera y se modifica en función del valor de α_0. Como consecuencia, se ha mejorado el dominio de estabilidad absoluta respecto al método de Euler. Sin embargo, la constante del error es

$$\frac{C_2}{2!}=\frac{\sum_{j=0}^{2}(j^2\alpha_j-2j\beta_j)}{2}=\frac{\alpha_1-2\beta_1+4\alpha_2}{2}.$$

Poniendo los coeficientes libres en función de α_0, β_0 se observa que

$$\frac{C_2}{2!}=\frac{1}{2}+\alpha_0+\beta_0 \tag{4.4}$$

aumenta al incrementar el valor de α_0.

Para finalizar, se va a estudiar como se modifica la región de estabilidad para el caso $\beta_0=0$ cuando se modifica el valor de α_0, como se observa en la Figura4.1. Se puede identificar que a mayor intervalo de estabilidad absoluta, menor es la longitud del eje menor de la elipse que lo define. Además, se observa que es un método fuertemente 0-estable ya que $0\in D$ y $0\notin D^\circ$.

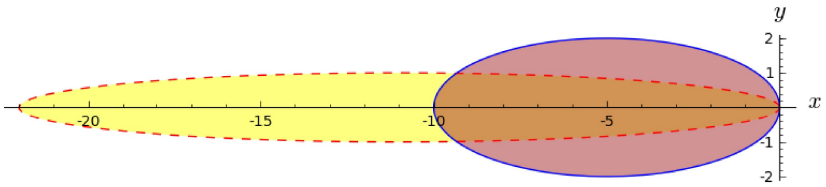

Figura 4.1: Contraste de la región de estabilidad cuando se modifica los parámetros libres α_0 y $\beta_0=0$. La curva representada con linea continua es el Boundary Locus cuando $\alpha_0=3$ y con línea discontinua es el Boundary Locus cuando $\alpha_0=10$. El contorno coloreado representa el dominio de estabilidad absoluta para dichos valores de α_0, es decir, los valores de z para los cuales $|\zeta_i(z)|<1, \forall i=1,\ldots,n$.

Además, se puede ver que el módulo de las raíces del polinomio de estabilidad, en el interior del dominio aumenta al aumentar el valor de α_0. En definitiva, al aumentar α_0 aumenta el IEA, pero también aumenta el módulo de las raíces y la constante del término del error.

En la práctica interesa elegir α_0 lo suficientemente grande para que el paso de integración se ajuste a los requisitos de comunicación, pero no excesivamente grande para evitar una constante del error muy grande y que las raíces del polinomio de estabilidad sean muy próximas a 1.

Capítulo 5

Evaluación empírica del método propuesto

5.1. Motivación

Una vez estudiado el marco teórico se ha observado que con el método de dos pasos y orden uno la frecuencia entre las comunicaciones es mayor. Sin embargo, aparte del tiempo entre comunicaciones consecutivas, existen otro tipo de indicadores cuyo cálculo teórico no es trivial utilizando estos métodos, por ejemplo el número de comunicaciones y el tiempo de convergencia en función de los parámetros libres. Para analizar estos datos se ha optado por una evaluación empírica de los datos por medio de simulaciones a partir de los métodos numéricos estudiados. Esto nos permite determinar los valores de α_0 y β_0 para los que el método multi-paso alcanza el consenso en un menor número de comunicaciones.

5.2. Descripción de los experimentos

Los Experimentos de Monte Carlo se han diseñado para estudiar diferentes propiedades del algoritmo de Euler en comparación con el método de dos pasos y orden uno. Se ha elegido Python como lenguaje de implementación, dado que es el lenguaje utilizado en las asignaturas del grado que guardan relación con los conocimientos aplicados en este TFG. La implementación del código del programa se puede ver en el Anexo. En todas las simulaciones se consideran 100 redes diferentes con 100 nodos cada una y topologías fija. Para cada red, los nodos han sido posicionados aleatoriamente generando un anillo en un cuadrado de $1x1$ metro. Dos nodos están comunicados si están a una distancia menor a 0,7 metros y en todos los casos se ha forzado que la red sea conexa.

Como algoritmo de iniciación del método multi-paso se ha usado Euler. Se han establecido dos comunicaciones sucesivas con el método de Euler. En las comunicaciones posteriores, el estado de cada sensor se obtiene a partir del método de dos pasos y orden 1 de tipo numérico que viene dado por:

$$\mathbf{x_k} = \frac{-\alpha_1 \mathbf{x_{k-1}} - \alpha_0 \mathbf{x_{k-2}} + h\beta_1 \dot{\mathbf{x}}_{\mathbf{k-1}} + h\beta_0 \dot{\mathbf{x}}_{\mathbf{k-2}}}{\alpha_2}, k = 2, 3, \ldots$$

siendo $\mathbf{x_k}$ el vector de estados del conjunto de nodos tras cada comunicación cuando el problema de consenso se discretiza.

Los objetivos del estudio son:

- Comparar los tiempos de paso medio y máximo de ambos métodos en función de los parámetros α_0 y β_0.
- Comparar el número de rondas de comunicación y el tiempo necesario para converger con una tolerancia dada modificando el valor de h (se trata de un valor límite,muy próximo al h óptimo) en función de los valores de α_0 y β_0.

- Tomando un tiempo de paso , h, fijo y menor que h_{max} de Euler, comparar de nuevo el número de rondas de comunicación necesarias para alcanzar el consenso en función de los métodos elegidos.

Para el estudio se han seleccionado 3 valores distintos de α_0 y 4 de β_0, generando 12 combinaciones distintas del método multipaso. Dichos valores vienen dados en el Cuadro 5.1.Se han utilizado valores distintos y que oscilan en un gran rango, esto permite un estudio más detallado para ver como se modifica el número de comunicaciones.

Por otro lado, al ser métodos asintóticos, el estudio de la convergencia, número de comunicaciones y tiempo, se ha realizado en función del error cometido con respecto al valor de consenso,

$$error = \|\overline{x_0}\mathbf{1} - \mathbf{x_k}\|,$$

siendo $\overline{x_0}\mathbf{1}$ e $\mathbf{x_k}$ el vector de consenso, media de las condiciones iniciales, y el del conjunto de estados tras cada iteración del método correspondiente respectivamente. Se han fijado diferentes tolerancias de error y se ha medido el tiempo necesario para converger en cada caso.

5.3. Resultados obtenidos

5.3.1. Caso I: Frecuencia entre dos comunicaciones variables en funcion de los valores de α_0 y β_0

- **Frecuencia entre comunicaciones**

Primeramente, se va a estudiar el tiempo medio y máximo entre comunicaciones consecutivas de las simulaciones dadas por los métodos definidos. El tiempo de paso máximo se define como

$$h^* = \frac{\frac{1}{2} + \alpha_0}{\text{máx}(\lambda_j)\left(\frac{\beta_0}{2} - \frac{1}{4}\right)}.$$

Se trata de un supremo del conjunto de $h's$ que hacen que converja el método. Los valores de h_{medio} vienen definidos por el promedio de los valores de

$$h = -\frac{\frac{1}{2} + \alpha_0}{(\text{máx}(\lambda_j) + 1)\left(\frac{\beta_0}{2} - \frac{1}{4}\right)}$$

que se generan en las 100 simulaciones. Aparece $\text{máx}(\lambda_j) + 1$ en el denominador porque se trate de un caso límite para el que el método converja puesto que $h < h^*$.

En el cuadro 5.1 se pueden observar los valores de $h_{\text{máx}}$ y h_{medio} para los modelos que se han definido.

Se puede ver que el máximo paso de integración se obtiene cuando $\beta_0 \to 0^-$ y $\alpha_0 \to \infty$. Además, cuanto más grande es α_0 mayor es el tiempo de paso. Cabe destacar que cuando β_0=-305.45, -204.35, -102.45 y α_0=167.5, los tiempos entre comunicaciones consecutivas son similares a los del método de Euler. Por lo tanto, se podría aplicar cualquiera de los dos métodos (dependiendo del número de comunicaciones necesarias para alcanzar el consenso). Para el resto de casos es significativamente superior, esto permite hacer cualquier estudio en condiciones reales cuando el número de sensores que definen una red es elevado.

- **Número de comunicaciones y tiempo de convergencia**

En el cuadro 5.2 se pueden observar el número medio de rondas de comunicación y tiempo de convergencia en función de los parámetros y la tolerancia.

El método de Euler necesita un menor número de comunicaciones en la mayoría de casos, excepto en $MP10$, $MP11$, $MP12$. Esto nos quiere decir que cuando $\beta_0 \to -\infty$ y $\alpha_0 \to \infty$, el número de comunicaciones aumenta. Para dichos casos es mejor el método de Euler, siempre que sea posible comunicarse

Paso de integración	β_0	α_0	$h_{\text{máx}}$	h_{medio}	Constante del error
Euler			0.025	0.024	
MP1	-305.450	16750	1.360	1.306	16445.050
MP2	-305.450	1675	0.136	0.131	1370.050
MP3	-305.450	167.500	0.014	0.013	-137.450
MP4	-204.350	16750	2.036	1.950	16546.150
MP5	-204.350	1675	0.204	0.195	1471.150
MP6	-204.350	167.500	0.041	0.039	-36.350
MP7	-102.450	16750	4.051	3.881	16648.050
MP8	-102.450	1675	0.405	0.388	1573.050
MP9	-102.450	167.500	0.0406	0.039	65.550
MP10	-0.135	16750	656.852	629.204	16750.365
MP11	-0.135	1675	65.703	62.937	1675.365
MP12	-0.135	167.500	6.588	6.311	167.865

Cuadro 5.1: Distintos métodos multipaso en los que se ha estudiado el número de comunicaciones y el tiempo de convergencia. También se puede ver el tiempo de paso máximo y medio generado por las 100 simulaciones programadas. Por otro lado, se ha realizado un estudio de la constante del error, en él se observa que su valor aumenta cuando $\beta_0 \to 0$ y $\alpha_0 \to \infty$.

Tolerancia	10^{-2}		10^{-3}		10^{-4}		10^{-5}	
Métodos	Com.	Tiempo	Com.	Tiempo	Com.	Tiempo	Com.	Tiempo
Euler	293.74	**7,03**	496.97	**11,82**	688.36	**16,43**	879.13	**20,98**
MP1	731.24	954.50	2141.40	2785.51	3546.31	4632.80	4952.60	6469.17
MP2	675.45	88.21	2086.08	271.46	3491.50	456.28	4895.16	639.62
MP3	678.14	8.92	2093.78	27.36	3504.16	45.96	4912.3	64.40
MP4	573.63	1117.51	1516.79	2945.66	2456.34	4791.39	3395.41	6622.79
MP5	549.66	107.14	1493.09	290.08	2433.08	474.77	3371.76	657.88
MP6	321.57	12.89	1213.36	47.17	2156.69	83.82	3098.53	120.47
MP7	366.07	1416.27	832.80	3214.71	1303.55	5055.91	1773.62	6879.96
MP8	361.49	139.93	827.2	319.43	1298.06	503.64	1768.3	686.16
MP9	313.34	12.19	738.8	28.64	1211.42	47.17	1683.32	65.53
MP10	**54,87**	33340.84	**83,01**	50800.26	**110,03**	68012.38	**138,53**	85930.57
MP11	54.88	3335.64	83.07	5085.80	110.24	6816.24	138.74	8608.64
MP12	54.93	334.82	84.06	516.21	112.41	829.48	141.52	880.74

Cuadro 5.2: Número medio de comunicaciones y tiempo de convergencia al realizar las 100 simulaciones. Se puede observar que para cualquier orden del error, el método MP10 necesita un menor número de comunicaciones. Por otra parte, el tiempo medio de convergencia de Euler es menor. Sin embargo, una desventaja es que el tiempo entre comunicaciones consecutivas es pequeño y requiere un número de comunicaciones elevados, por lo tanto no nos interesa. Para los casos MP10, MP11 y MP12 el tiempo de convergencia es elevado, siendo MP10 el valor más grande y el que necesitaba un menor número de comunicaciones. Esta relación viene dada debido a que a mayor tiempo entre comunicaciones consecutivas, mayor es el tiempo de convergencia.

con la frecuencia requerida por dicho método. Los métodos $MP10$, $MP11$ y $MP12$ necesitan un menor número de comunicaciones para alcanzar el consenso y se observa que la diferencias en el número medio de iteraciones entre el método de Euler y los multipaso son mayores cuanto menor es el valor de la tolerancia. El mínimo número de comunicaciones necesarias para alcanzar el consenso se encuentra en

un entorno de β_0=-0.1315 y $\alpha_0 \to \infty$.

El método multi-paso respecto al tiempo de convergencia es peor. Se puede minimizar cuando $\beta_0 \to -\infty$ y $\alpha_0 \to 0$ como se puede ver en el método MP3.

El problema es que en el método de Euler el tiempo entre dos comunicaciones sucesivas es muy pequeño, próximo a $0.0239s$. En la vida real es muy difícil que se trate de un intervalo de tiempo tan pequeño, no se podría observar las modificaciones tras cada comunicación y además se había establecido que el tiempo entre dos comunicaciones sucesivas estaba acotado inferiormente por $\overline{h}$. Una de las ventajas de los métodos multipaso es que el tiempo de paso entre comunicaciones es mayor, esto implica que la convergencia será más lenta.

5.3.2. Caso II: Tiempo de paso fijo e igual para todos los modelos

Supongamos que el tiempo de paso es igual para todos los métodos y menor que el necesario para que el método de Euler converja. Necesariamente $h_{Euler} = h_{multipaso}$, es decir, $\frac{2}{\text{máx}(\lambda_j)+1} = -\frac{\frac{1}{2}+\alpha_0}{(\text{máx}(\lambda_j)+1)\left(\frac{\beta_0}{2}-\frac{1}{4}\right)}$.
Esta condición se cumple cuando $\beta_0 = -\alpha_0$. En este caso particular, sólo interesa el estudio del número de comunicaciones necesarias porque el tiempo de convergencia se obtiene al multiplicarlo por h. Se van a dar 4 valores distintos a α_0 que generan las 4 modificaciones distintas del método multipaso para poder compararlos, vienen definidos en el Cuadro 5.3.

Número de comunicaciones

En el cuadro 5.3 se puede observar el número medio de iteraciones cuando se modifica la tolerancia del método. El objetivo principal es estudiar como se modifica el número de comunicaciones necesarias para alcanzar el consenso y ver para que valores de α_0 se minimizan las comunicaciones.

Métodos	α_0	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler		12.06	17.24	22.82	28.27
MP1	305.45	13.05	18.42	23.91	29.46
MP2	204.35	12.99	18.35	23.75	29.46
MP3	102.45	13.11	18.34	24.03	29.48
MP4	0.135	13.06	18.24	23.82	29.27

Cuadro 5.3: Número medio de comunicaciones al realizar las 100 simulaciones en las distintas variantes del método. Se puede observar que para cualquier orden del error, el método de Euler necesita un menor número de comunicaciones. Por lo tanto, en este caso es mejor el método.

En todos los métodos *MPi* se puede observar que es necesaria aproximadamente una comunicación más para converger. Por lo tanto, si se quiere tomar un valor fijo y pequeño h es mejor utilizar el método de Euler.

Capítulo 6

Conclusiones finales

6.1. Conclusión

En este trabajo, a partir de una red de sensores n-dimensional se ha tratado de implementar al problema de consenso por medio de un método de dos pasos y orden uno que permitiese discretizarlo. Esto permite obtener el estado de los sensores después de cada comunicación.

A partir del estudio teórico del método multipaso, como alternativa de discretización del sistema dinámico que definen las ecuaciones del problema del consenso, se ha obtenido que el intervalo de estabilidad absoluta puede ser tan grande como se quiera en función de los parámetros α_0 y β_0. Esto permite coger el paso de integración tan grande como se quiera cogiendo α_0 y β_0 oportunos. Por consiguiente se ha obtenido una mejora bastante notable respecto a Euler.

Se ha tratado de implementar el código para poder realizar una evaluación empírica de la solución por medio de simulaciones en Python. Como hay indicadores en los que el cálculo teórico no es trivial, como el número de comunicaciones y el tiempo de convergencia, se van a calcular por medio de las simulaciones.

Cuando el paso de integración es variable, en el resultado de estas evaluaciones empíricas se obtiene que el número de comunicaciones para alcanzar el consenso es menor cuando el tiempo de paso es grande. Esto implica que el tiempo de convergencia es muy grande. En el resto de casos, se puede observar que el número de comunicaciones es mayor, por consiguiente no se observa mejoría respecto al método de Euler. Sin embargo, cuando el paso de integración es fijo para ambos, el mínimo número de comunicaciones necesarias es menor para Euler.

A pesar de no presentar mejoría en este aspecto, se termina con uno de los grandes inconvenientes que presentan los métodos utilizados hasta el momento para resolver este tipo de problemas, el tiempo entre comunicaciones consecutivas. Esto permite la realización de experimentos en condiciones más realistas y no establecer que el tiempo entre dos comunicaciones sucesivas sea 2 milésimas de segundo.

6.2. Lineas futuras

Existen varios puntos en los que se podría hacer un estudio más detallado. Algunos de ellos, vienen dadas por el problema estudiado. Por ejemplo:

- Cuando se tenía el caso $\beta_0 = 0$ se observaba que a mayor valor de α_0 la constante del error va aumentando, esto hace que el orden del método se reduzca y pase a ser de orden cero. Por lo tanto, sería interesante realizar este mismo estudio pero con un método de tres pasos y orden dos. Para ello, se tendría que realizar de nuevo el estudio y ver como se modifica la constante del error para dicho método. Otra pregunta interesante que podría plantear un estudio: ¿ Es más importante perder un poco de dominio de estabilidad absoluta para que el módulo de las raíces sea menor y así lograr que el número de comunicaciones para alcanzar el consenso sea menor ?

- Por otro lado, al realizar la comprobación en el caso particular de $\beta_0 = 0$ se observa que al variar el valor de la z dentro del intervalo de estabilidad, el módulo de las raíces complejas es siempre el mismo. Se puede estudiar si es consecuencia de alguna propiedad de convergencia.
- Realizar el estudio con un método de orden mayor, por ejemplo un método de 3 pasos y orden 2 para ver como se modifica el problema planteado.

Bibliografía

[1] M.CALVO, J.I.MONTIJANO Y L. RÁNDEZ, *Métodos lineales multipaso para la resolución numérica de ecuaciones diferenciales ordinarias*, Curso Análisis Numérico , Dpto. de Matemática Aplicada Universidad de Zaragoza.Páginas 201-288

[2] EDUARDO MONTIJANO MUÑOZ, *Distributed Consensus in Multi-Robot System with Visual Perception*, Tesis ,Dpto. de Informática e Ingeniería de Sistemas. Escuela de Ingeniería y Arquitectura en Universidad de Zaragoza. Páginas 21-39

[3] MEHRAN MESBAHI AND MAGNUS EGERSTEDT, *Graph Theoretic Methods in Multiagent Networks*,Princeton Series in Applied Mathematics . Páginas 3-50

[4] E.HAIRER,S.P.NØRSETT Y G.WANNER, *Solving Ordinary Differential Equations I Nonstiff Problems*, 2.ª ed., Springer, 1987,379-380. Páginas 240-242

Anexo

Programa en Python.

```
from numpy import *
from scipy import *
from pylab import *
import numpy as np
from matplotlib.pylab import hist, show
import matplotlib.pyplot as plt
from numpy import linalg as LA
import seaborn as sns
from random import *

# Definición de funcion derivada
def fcn(t, x, L):
    return dot(-L,x)
#PARÁMETROS CONOCIDOS
#numero de sensores
n = 100;
p = 0.7;
#tiempo inicio de la simulación
t0 = 0.0;
#numero de comunicaciones máximas para alcanzar el consenso
npasos =1000;
#numero de simulaciones
num_sims=100;
#tolerancia del método
tol=0.001

#DEfinición de listas para guaradar la frecuencia
#de comunicaciones de los métodos
paso_integración_Euler=[]
paso_integración_MP1=[]
paso_integración_MP2=[]
paso_integración_MP3=[]
paso_integración_MP4=[]
paso_integración_MP5=[]
paso_integración_MP6=[]
paso_integración_MP7=[]
paso_integración_MP8=[]
paso_integración_MP9=[]
paso_integración_MP10=[]
```

```
paso_integración_MP11=[]
paso_integración_MP12=[]

#Definición de listas para guardar la máxima
# frecuencia de comunicaciones de los métodos

paso_integración_max_Euler=[]
paso_integración_max_MP1=[]
paso_integración_max_MP2=[]
paso_integración_max_MP3=[]
paso_integración_max_MP4=[]
paso_integración_max_MP5=[]
paso_integración_max_MP6=[]
paso_integración_max_MP7=[]
paso_integración_max_MP8=[]
paso_integración_max_MP9=[]
paso_integración_max_MP10=[]
paso_integración_max_MP11=[]
paso_integración_max_MP12=[]

#Definición de listas para guardar el número de
#comunicaciones necesarias para alcanzar el consenso

numer_pasos=[]
numer_pasos_multipaso1=[]
numer_pasos_multipaso2=[]
numer_pasos_multipaso3=[]
numer_pasos_multipaso4=[]
numer_pasos_multipaso5=[]
numer_pasos_multipaso6=[]
numer_pasos_multipaso7=[]
numer_pasos_multipaso8=[]
numer_pasos_multipaso9=[]
numer_pasos_multipaso10=[]
numer_pasos_multipaso11=[]
numer_pasos_multipaso12=[]

#Definición de listas para guardar el tiempo de
#convergencia de los métodos para alcanzar el consenso

t_converEuler=[]
t_converMP1=[]
t_converMP2=[]
t_converMP3=[]
t_converMP4=[]
t_converMP5=[]
t_converMP6=[]
t_converMP7=[]
t_converMP8=[]
t_converMP9=[]
t_converMP10=[]
```

```
t_converMP11=[]
t_converMP12=[]

X = []
Y = []

for _ in range(num_sims):
    #inicializacion de la matriz de adyacencia
    A = []
    for i in range(n):
        A.append([0]*n)
    #calculo de la matriz de adyacencia
    for i in range(n):
        for j in range(i+1,n):
            if np.random.rand() < p:
                A[i][j] = 1;
                A[j][i] = 1;
            else:
                A[i][j] = 0;
    L=np.diag(np.sum(A, axis=1))-A;
    r=LA.eigvals(L)
    u=max(r)
    hmax=2/u
    h=2/(u+1)
    #lista que guarda los valores de h
    paso_integración_Euler.append(h)
    #lista que guarda los valores máximo de h(Se trata de un supremo)
    paso_integración_max_Euler.append(h)
    #vector de condiciones iniciales aleatorios
    valorinicial=xn2=np.random.rand(n)
    x = array(zeros(size(xn2)));
    tab = zeros([npasos,4])
    #inicialización del error del método de Euler
    error_Euler=100;
    i=1

    while error_Euler > tol and i < npasos:
          tn = t0 + (i-1)*h
          x = xn2 +  h*fcn(tn, xn2, L)   #Euler
          xn2 = x
          X.append(xn2)
          #tab[i]=tn+h;
          error_Euler=np.linalg.norm(mean(valorinicial)*np.ones(n) - x)
          i +=1
    #Listas que nos guardan el número de pasos del método de Euler
    numer_pasos.append(i)
    #Listas que nos guardan el tiempo de convergencia del método de Euler
    t_converEuler.append(i*h)
    y  = array(zeros(3*size(xn2)));
    #Valores que toma la variable beta0
    beta0_list=[-0.1315,-0.1325,-0.1335,-0.1345]
```

```
for s in range(0,4):
    beta1=1-beta0_list[s];
    #valores que toma la variable alfa0
    alfa0_list=[171.5,172,172.5]
    #inicialización de los errores de los métodos multipaso
    error_multipaso=[100,100,100]
    for k in range(0,3):
        #CALCULO DE DATOS
        #CONDICIONES INICIALES(Algoritmo de iniciación Método Euler)
        xn2=valorinicial;
        fxn2=fcn(t0, xn2, L)
        xn1 = xn2 +  h*fcn(t0, xn2, L)
        fxn1=fcn(t0,xn1,L)
        alfa1=-1-2*alfa0_list[k];
        alfa2=1+alfa0_list[k];
        h_=-(1/2+alfa0_list[k])/((u+1)*(beta0_list[s]/2-1/4))
        #Valor máximo de h (Se trata de un supremo)
        h_max=-(1/2+alfa0_list[k])/((u)*(beta0_list[s]/2-1/4))
        if s==0 and k==0:
           paso_integración_MP1.append(h_)
           #Paso de integración máximo del método multipaso MP1
           paso_integración_max_MP1.append(h_max)
        if s==0 and k==1:
           paso_integración_MP2.append(h_)
           #Paso de integración máximo del método multipaso MP2
           paso_integración_max_MP2.append(h_max)
        if s==0 and k==2:
           paso_integración_MP3.append(h_)
           #Paso de integración máximo del método multipaso MP3
           paso_integración_max_MP3.append(h_max)
        if s==1 and k==0:
           paso_integración_MP4.append(h_)
           #Paso de integración máximo del método multipaso MP4
           paso_integración_max_MP4.append(h_max)
        if s==1 and k==1:
           paso_integración_MP5.append(h_)
           #Paso de integración máximo del método multipaso MP5
           paso_integración_max_MP5.append(h_max)
        if s==2 and k==2:
           paso_integración_MP6.append(h_)
           #Paso de integración máximo del método multipaso MP6
           paso_integración_max_MP6.append(h_max)
        if s==2 and k==0:
           paso_integración_MP7.append(h_)
           #Paso de integración máximo del método multipaso MP7
           paso_integración_max_MP7.append(h_max)
        if s==2 and k==1:
           paso_integración_MP8.append(h_)
           #Paso de integración máximo del método multipaso MP8
           paso_integración_max_MP8.append(h_max)
        if s==2 and k==2:
```

```
            paso_integración_MP9.append(h_)
            #Paso de integración máximo del método multipaso MP9
            paso_integración_max_MP9.append(h_max)
        if s==3 and k==0:
            paso_integración_MP10.append(h_)
            #Paso de integración máximo del método multipaso MP10
            paso_integración_max_MP10.append(h_max)
        if s==3 and k==1:
            paso_integración_MP11.append(h_)
            #Paso de integración máximo del método multipaso MP10
            paso_integración_max_MP11.append(h_max)
        if s==3 and k==2:
            paso_integración_MP12.append(h_)
            #Paso de integración máximo del método multipaso MP12
            paso_integración_max_MP12.append(h_max)
        j=1
    #BUCLE DE SIMULACIÓN
        while error_multipaso[k] > tol and j < npasos:
              tn = t0 + (j-1)*h_
              #MÉTODO DOS PASOS
              y= (-alfa1*xn1-alfa0_list[k]*xn2 +
                  h_*beta1*fxn1 + h_*beta0_list[s]*fxn2)/alfa2
              fy=fcn(tn, y, L)
              xn2=xn1
              xn1=y
              fxn2=fxn1
              fxn1=fy
              Y.append(xn2)
              error_multipaso[k]=np.linalg.norm(mean(valorinicial)*np.ones(n)-y)
              j += 1
        if s==0:
           if k==0 and error_multipaso[0] < tol:
              #Lista del número de comunicaciones del método MP1
              #siendo j el número de comunicaciones de MP1 y 2 de Euler
              numer_pasos_multipaso1.append(j+2)
              #Lista del tiempo de convergencia del método MP1
              #siendo j*h_ el tiempo de convergencia de MP1 y 2*h de Euler
              t_converMP1.append(j*h_+2*h)
           if k==1 and error_multipaso[1] < tol:
              #Lista del número de comunicaciones del método MP2
              #siendo j el número de comunicaciones de MP2 y 2 de Euler
              numer_pasos_multipaso2.append(j+2)
              #Lista del tiempo de convergencia del método MP2
              #siendo j*h_ el tiempo de convergencia de MP2 y 2*h de Euler
              t_converMP2.append(j*h_+2*h)
           if k==2 and error_multipaso[2] < tol:
              #Lista del número de comunicaciones del método MP3
              #siendo j el número de comunicaciones de MP3 y 2 de Euler
              numer_pasos_multipaso3.append(j+2)
              #Lista del tiempo de convergencia del método MP3
              #siendo j*h_ el tiempo de convergencia de MP3 y 2*h de Euler
```

```
            t_converMP3.append(j*h_+2*h)
      if s==1:
         if k==0 and error_multipaso[0] < tol:
            #Lista del número de comunicaciones del método MP4
            #siendo j el número de comunicaciones de MP4 y 2 de Euler
            numer_pasos_multipaso4.append(j+2)
            #Lista del tiempo de convergencia del método MP4
            #siendo j*h_ el tiempo de convergencia de MP4 y 2*h de Euler
            t_converMP4.append(j*h_+2*h)
         if k==1 and error_multipaso[1] < tol:
            #Lista del número de comunicaciones del método MP5
            #siendo j el número de comunicaciones de MP5 y 2 de Euler
            numer_pasos_multipaso5.append(j+2)
            #Lista del tiempo de convergencia del método MP5
            #siendo j*h_ el tiempo de convergencia de MP5 y 2*h de Euler
            t_converMP5.append(j*h_+2*h)
         if k==2 and error_multipaso[2] < tol:
            #Lista del número de comunicaciones del método MP6
            #siendo j el número de comunicaciones de MP6 y 2 de Euler
            numer_pasos_multipaso6.append(j+2)
            #Lista del tiempo de convergencia del método MP6
            #siendo j*h_ el tiempo de convergencia de MP6 y 2*h de Euler
            t_converMP6.append(j*h_+2*h)
      if s==2:
         if k==0 and error_multipaso[0] < tol:
            #Lista del número de comunicaciones del método MP7
            #siendo j el número de comunicaciones de MP7 y 2 de Euler
            numer_pasos_multipaso7.append(j+2)
            #Lista del tiempo de convergencia del método MP7
            #siendo j*h_ el tiempo de convergencia de MP7 y 2*h de Euler
            t_converMP7.append(j*h_+2*h)
         if k==1 and error_multipaso[1] < tol:
            #Lista del número de comunicaciones del método MP8
            #siendo j el número de comunicaciones de MP8 y 2 de Euler
            numer_pasos_multipaso8.append(j+2)
            #Lista del tiempo de convergencia del método MP8
            #siendo j*h_ el tiempo de convergencia de MP8 y 2*h de Euler
            t_converMP8.append(j*h_+2*h)
         if k==2 and error_multipaso[2] < tol:
            #Lista del número de comunicaciones del método MP9
            #siendo j el número de comunicaciones de MP9 y 2 de Euler
            numer_pasos_multipaso9.append(j+2)
            #Lista del tiempo de convergencia del método MP9
            # siendo j*h_ el tiempo de converencia de MP9 y 2*h de Euler
            t_converMP9.append(j*h_+2*h)
      if s==3:
         if k==0 and error_multipaso[0] < tol:
            #Lista del número de comunicaciones del método MP10
            #siendo j el número de comunicaciones de MP10 y 2 de Euler
            numer_pasos_multipaso10.append(j+2)
            #Lista del tiempo de convergencia del método MP10
```

```
                    #siendo j*h_ el tiempo de convergencia de MP10 y 2*h de Euler
                    t_converMP10.append(j*h_+2*h)
                if k==1 and error_multipaso[1] < tol:
                    #Lista del número de comunicaciones del método MP11
                    #siendo j el número de comunicaciones de MP11 y 2 de Euler
                    numer_pasos_multipaso11.append(j+2)
                    #Lista del tiempo de convergencia del método MP11
                    # siendo j*h_ el tiempo de convergencia de MP11 y 2*h de Euler
                    t_converMP11.append(j*h_+2*h)
                if k==2 and error_multipaso[2] < tol:
                    #Lista del número de comunicaciones del método MP12
                    #siendo j el número de comunicaciones de MP12 y 2 de Euler
                    numer_pasos_multipaso12.append(j+2)
                    #Lista del tiempo de convergencia del método MP12
                    #siendo j*h_ el tiempo de convergencia de MP12 y 2*h de Euler
                    t_converMP12.append(j*h_+2*h)

#PASOS DE INTEGRACIÓN
#Máximo

print("Máximo paso de integración de Euler:")
print(np.max(paso_integración_max_Euler))
print("Máximo paso de integración de MP1:")
print(np.max(paso_integración_max_MP1))
print("Máximo paso de integración de MP2:")
print(np.max(paso_integración_max_MP2))
print("Máximo paso de integración de MP3:")
print(np.max(paso_integración_max_MP3))
print("Máximo paso de integración de MP4:")
print(np.max(paso_integración_max_MP4))
print("Máximo paso de integración de MP5:")
print(np.max(paso_integración_max_MP5))
print("Máximo paso de integración de MP6:")
print(np.max(paso_integración_max_MP6))
print("Máximo paso de integración de MP7:")
print(np.max(paso_integración_max_MP7))
print("Máximo paso de integración de MP8:")
print(np.max(paso_integración_max_MP8))
print("Máximo paso de integración de MP9:")
print(np.max(paso_integración_max_MP9))
print("Máximo paso de integración de MP10:")
print(np.max(paso_integración_max_MP10))
print("Máximo paso de integración de MP11:")
print(np.max(paso_integración_max_MP11))
print("Máximo paso de integración de MP12:")
print(np.max(paso_integración_max_MP12))

#Media

print("Media del paso de integración de Euler:")
print(np.mean(paso_integración_Euler))
```

```
print("Media del paso de integración de MP1:")
print(np.mean(paso_integración_MP1))
print("Media del paso de integración de MP2:")
print(np.mean(paso_integración_MP2))
print("Media del paso de integración de MP3:")
print(np.mean(paso_integración_MP3))
print("Media del paso de integración de MP4:")
print(np.mean(paso_integración_MP4))
print("Media del paso de integración de MP5:")
print(np.mean(paso_integración_MP5))
print("Media del paso de integración de MP6:")
print(np.mean(paso_integración_MP6))
print("Media del paso de integración de MP7:")
print(np.mean(paso_integración_MP7))
print("Media del paso de integración de MP8:")
print(np.mean(paso_integración_MP8))
print("Media del paso de integración de MP9:")
print(np.mean(paso_integración_MP9))
print("Media del paso de integración de MP10:")
print(np.mean(paso_integración_MP10))
print("Media del paso de integración de MP11:")
print(np.mean(paso_integración_MP11))
print("Media del paso de integración de MP12:")
print(np.mean(paso_integración_MP12))

#ESTADÍSTICAS DEL NÚMERO DE ITERACIONES DE CADA MÉTODO (MEDIA, VARIANZA)

print("Media del número de iteraciones en el método de Euler")
print(np.mean(numer_pasos),)
print("Varianza del número de iteraciones en el método de Euler")
print(np.var(numer_pasos, ddof=0))
print("Media del número de iteraciones en el método MP1")
print(np.mean(numer_pasos_multipaso1))
print("Varianza del número de iteraciones en el método MP1")
print(np.var(numer_pasos_multipaso1, ddof=0))
print("Media del número de iteraciones en el método MP2")
print(np.mean(numer_pasos_multipaso2))
print("Varianza del número de iteraciones en el método MP2")
print(np.var(numer_pasos_multipaso2, ddof=0))
print("Media del número de iteraciones en el método MP3")
print(np.mean(numer_pasos_multipaso3))
print("Varianza del número de iteraciones en el método MP3")
print(np.var(numer_pasos_multipaso3, ddof=0))
print("Media del número de iteraciones en el método MP4")
print(np.mean(numer_pasos_multipaso4))
print("Varianza del número de iteraciones en el método MP4")
print(np.var(numer_pasos_multipaso4, ddof=0))
print("Media del número de iteraciones en el método MP5")
print(np.mean(numer_pasos_multipaso5))
print("Varianza del número de iteraciones en el método MP5")
print(np.var(numer_pasos_multipaso5, ddof=0))
```

```
print("Media del número de iteraciones en el método MP6")
print(np.mean(numer_pasos_multipaso6))
print("Varianza del número de iteraciones en el método MP6")
print(np.var(numer_pasos_multipaso6, ddof=0))
print("Media del número de iteraciones en el método MP7")
print(np.mean(numer_pasos_multipaso7))
print("Varianza del número de iteraciones en el método MP7")
print(np.var(numer_pasos_multipaso7, ddof=0))
print("Media del número de iteraciones en el método MP8")
print(np.mean(numer_pasos_multipaso8))
print("Varianza del número de iteraciones en el método MP8")
print(np.var(numer_pasos_multipaso8, ddof=0))
print("Media del número de iteraciones en el método MP9")
print(np.mean(numer_pasos_multipaso9))
print("Varianza del número de iteraciones en el método MP9")
print(np.var(numer_pasos_multipaso9, ddof=0))
print("Media del número de iteraciones en el método MP10")
print(np.mean(numer_pasos_multipaso10))
print("Varianza del número de iteraciones en el método MP10")
print(np.var(numer_pasos_multipaso10, ddof=0))
print("Media del número de iteraciones en el método MP11")
print(np.mean(numer_pasos_multipaso11))
print("Varianza del número de iteraciones en el método MP11")
print(np.var(numer_pasos_multipaso11, ddof=0))
print("Media del número de iteraciones en el método MP12")
print(np.mean(numer_pasos_multipaso12))
print("Varianza del número de iteraciones en el método MP12")
print(np.var(numer_pasos_multipaso12, ddof=0))

#MEDIA DEL TIEMPO DE CONVERGENCIA DE CADA MÉTODO (MEDIA, VARIANZA)

print("Media del tiempo de convergencia en el método de Euler")
print(mean(t_converEuler))
print("Varianza del tiempo de convergencia en el método de Euler")
print(np.var(t_converEuler, ddof=0))
print("Media del tiempo de convergencia en el método MP1")
print(mean(t_converMP1))
print("Varianza del tiempo de convergencia en el método MP1")
print(np.var(t_converMP1, ddof=0))
print("Media del tiempo de convergencia en el método MP2")
print(mean(t_converMP2))
print("Varianza del tiempo de convergencia en el método MP2")
print(np.var(t_converMP2, ddof=0))
print("Media del tiempo de convergencia en el método MP3")
print(mean(t_converMP3))
print("Varianza del tiempo de convergencia en el método MP3")
print(np.var(t_converMP3, ddof=0))
print("Media del tiempo de convergencia en el método MP4")
print(mean(t_converMP4))
print("Varianza del tiempo de convergencia en el método MP4")
print(np.var(t_converMP4, ddof=0))
```

```
print("Media del tiempo de convergencia en el método MP5")
print(mean(t_converMP5))
print("Varianza del tiempo de convergencia en el método MP5")
print(np.var(t_converMP5, ddof=0))
print("Media del tiempo de convergencia en el método MP6")
print(mean(t_converMP6))
print("Varianza del tiempo de convergencia en el método MP6")
print(np.var(t_converMP6, ddof=0))
print("Media del tiempo de convergencia en el método MP7")
print(mean(t_converMP7))
print("Varianza del tiempo de convergencia en el método MP7")
print(np.var(t_converMP7, ddof=0))
print("Media del tiempo de convergencia en el método MP8")
print(mean(t_converMP8))
print("Varianza del tiempo de convergencia en el método MP8")
print(np.var(t_converMP8, ddof=0))
print("Media del tiempo de convergencia en el método MP9")
print(mean(t_converMP9))
print("Varianza del tiempo de convergencia en el método MP9")
print(np.var(t_converMP9, ddof=0))
print("Media del tiempo de convergencia en el método MP10")
print(mean(t_converMP10))
print("Varianza del tiempo de convergencia en el método MP10")
print(np.var(t_converMP10, ddof=0))
print("Media del tiempo de convergencia en el método MP11")
print(mean(t_converMP11))
print("Varianza del tiempo de convergencia en el método MP11")
print(np.var(t_converMP11, ddof=0))
print("Media del tiempo de convergencia en el método MP12")
print(mean(t_converMP12))
print("Varianza del tiempo de convergencia en el método MP12")
print(np.var(t_converMP12, ddof=0))

#DIAGRAMA DE CAJAS DE NÚMERO DE PASOS SEPARADOS POR LOS TRECE MÉTODOS

numer_pasos_graf = [numer_pasos, numer_pasos_multipaso1, numer_pasos_multipaso2,
                    numer_pasos_multipaso3,numer_pasos_multipaso4,numer_pasos_multipaso5,
                    numer_pasos_multipaso6,numer_pasos_multipaso7,numer_pasos_multipaso8,
                    numer_pasos_multipaso9,numer_pasos_multipaso10,numer_pasos_multipaso11,
                    numer_pasos_multipaso12]

t_conver_graf = [t_converEuler, t_converMP1, t_converMP2, t_converMP3,
                 t_converMP4,t_converMP5,t_converMP6,t_converMP7,
                 t_converMP8,t_converMP9,t_converMP10,t_converMP11,t_converMP12]

# Creando el objeto figura
fig = plt.figure(1, figsize=(9, 6))

# Creando el subgrafico
ax = fig.add_subplot(111)
```

```
# creando el grafico de cajas
bp = ax.boxplot(t_conver_graf)
#TÍTULO
plt.title('Diagrama de cajas del tiempo de convergencia')
#Ejes
xlabel('$Métodos$'); ylabel('$Tiempo$ $de$ $convergencia$');show()
# visualizar mas facile los atípicos
for flier in bp['fliers']:
    flier.set(marker='o', color='red', alpha=0.5)
# los puntos aislados son valores atípicos
```