GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

SIMULADOR MULTI – TREN DE UNA LÍNEA DE METRO PARA ENSAYOS DE REGULACIÓN AUTOMÁTICA DEL TRÁFICO

Autor: Álvaro Cidoncha González

Directores: Adrián Fernández Rodríguez; Antonio Fernández Cardador

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

SIMULADOR MULTI – TREN DE UNA LÍNEA DE METRO PARA ENSAYOS DE REGULACIÓN AUTOMÁTICA DEL TRÁFICO

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2021/22 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Álvaro Cidoncha González Fecha: 22 / 06 / 2022

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fdo.: Adrián Fernández Rodríguez y Antonio Fernández Cardador Fecha: 22 / 06 / 2022

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

SIMULADOR MULTI – TREN DE UNA LÍNEA DE METRO PARA ENSAYOS DE REGULACIÓN AUTOMÁTICA DEL TRÁFICO

Autor: Álvaro Cidoncha González

Directores: Adrián Fernández Rodríguez; Antonio Fernández Cardador

Madrid

Agradecimientos

A mis directores del proyecto, Adrián, Antonio y Paloma, por darme la oportunidad de adentrarme en el sector ferroviario de su mano, por haber respondido todas mis dudas con tanta paciencia y por haber conseguido que haya disfrutado y aprendido tanto.

A mis padres, Carmen y Miguel, por confiar en mí en todo momento y por su apoyo incondicional y sacrificio. También a Ane, por haber hecho que se me hayan pasado volando los años de carrera y por sacarme hacia delante con todo el cariño del mundo cuando las cosas se ponían cuesta arriba.

A los amigos que me ha brindado el ICAI, en especial a Juan Antonio, por su desinteresada ayuda en momentos difíciles.

A todos los profesores que durante la carrera han contribuido en mi formación, por su profesionalidad y cercanía.

SIMULADOR MULTI – TREN DE UNA LÍNEA DE METRO PARA ENSAYOS DE REGULACIÓN AUTOMÁTICA DEL TRÁFICO

Autor: Cidoncha González, Álvaro.
Directores: Fernández Rodríguez, Adrián; Fernández Cardador, Antonio.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

El objetivo principal del proyecto es el desarrollo de un modelo gemelo (twin model) de una línea de metro para el ensayo y análisis de estrategias de regulación de tráfico de trenes en lenguaje de programación Matlab. En particular, se desarrollarán modelos para replicar de manera realista el movimiento, la conducción y el consumo de energía de un tren. La línea para la que se diseña dicho modelo gemelo, corresponde a una importante línea de metro de España.

Además, se implementarán modelos que simulen el comportamiento de varios trenes en una línea incluyendo la influencia de los tiempos de parada, de naturaleza estocástica, y las interferencias entre unos trenes y otros, que estarán condicionadas por el sistema de señalización.

Finalmente, se aplica un regulador de tráfico eficiente y funcional que sea capaz de mantener un intervalo regular entre trenes y de corregir desviaciones en la velocidad comercial.

Palabras clave: Modelo de tráfico; Regulador automático de tráfico; ATO; Simulación de circulación.

1. Introducción

En una línea de metro moderna, el tráfico de trenes se regula desde el puesto central de mando enviando a cada tren la consigna de conducción que debe realizar. Para ello, es necesario el diseño de un modelo de regulación automática del tráfico eficiente y preciso.

Adicionalmente, la implementación de mejoras en la regulación del tráfico en las líneas de metro supone un aumento de la calidad del servicio, tanto en el mantenimiento de la velocidad comercial como en la regularidad y, por lo tanto, un mayor uso de este transporte por parte de los pasajeros.

Consecuentemente, el incremento del uso del metro por parte de los ciudadanos supondría una disminución en la emisión de gases de efecto invernadero, ya que el uso del ferrocarril en las grandes ciudades es el método más eficiente para reducir la polución resultante del transporte.

2. Definición del Proyecto

Para crear el modelo de regulación automática del tráfico para trenes de una línea de metro, será necesario diseñar todos los modelos que a su vez gobiernan el funcionamiento y movimiento de un solo tren. Una vez que se ha comprobado que un solo tren es capaz de circular por las vías, se implementarán varios trenes en la misma línea para, finalmente, diseñar el modelo de regulación automática del tráfico.

A modo de introducción, en la Ilustración 1 se presenta un posible esquema de la estructura del proyecto:

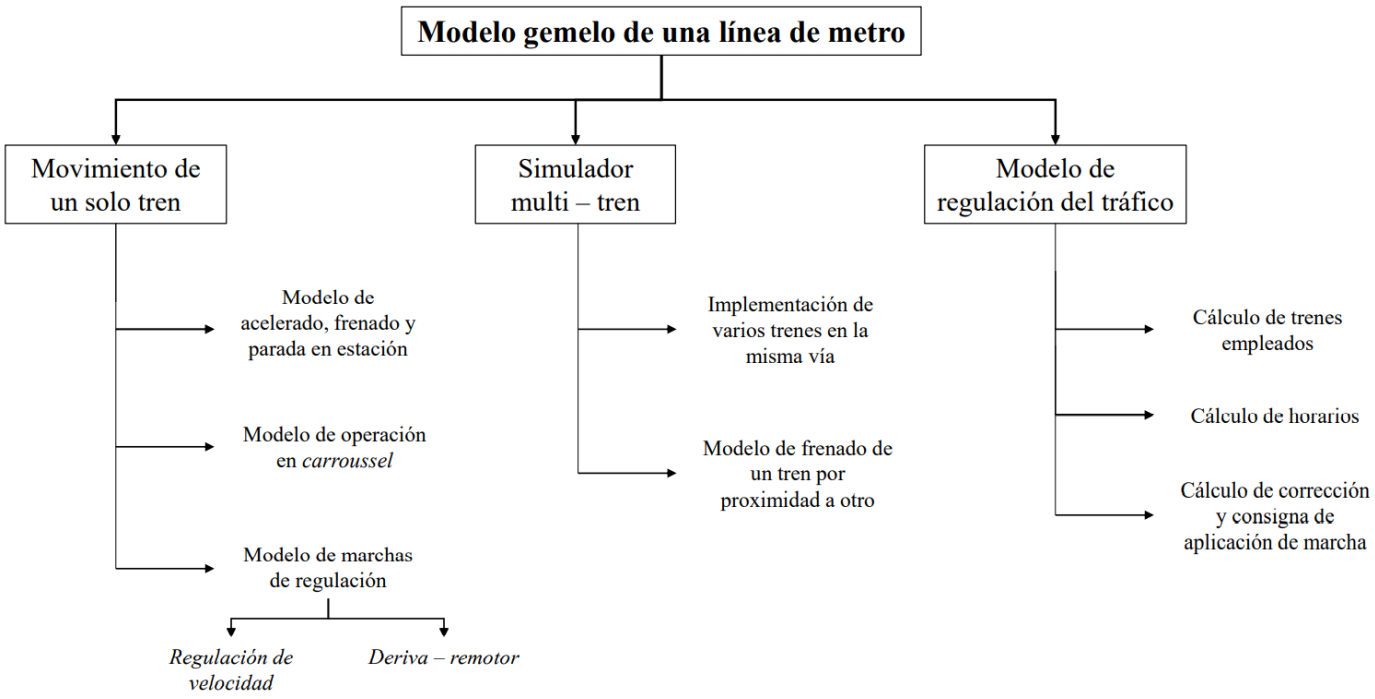

Ilustración 1: esquema de la estructura del proyecto

3. Descripción del modelo

Según se ha mencionado anteriormente, para obtener el modelo gemelo de una línea de metro y poder realizar ensayos de regulación automática del tráfico, se han de cumplimentar los modelos de las siguientes partes:

- Modelo del movimiento de un tren.
- Modelo del simulador multi – tren.
- Modelo de regulación del tráfico.

Modelo del movimiento de un tren

Para comenzar el diseño del código del modelo gemelo en Matlab, lo primero que hay que hacer es definir los objetos del tren y de las vías: vía 1 cuando el tren recorra la vía en un sentido y vía 2 cuando la recorra en sentido contrario.

En el objeto *Tren* se almacenarán sus propiedades constantes como la masa, la longitud y los coeficientes *a*, *b*, y *c* involucrados en la fuerza de resistencia al avance y sus propiedades variables, es decir, la posición y la velocidad del tren que éste lleva en cada iteración de tiempo. Por otro lado, en los objetos *Linea* y *Linea2* se definirán las propiedades constantes de las vías 1 y 2 que son la velocidad máxima que puede llevar el tren en cada tramo, la pendiente de la vía en cada punto y los puntos kilométricos de las estaciones. En la Tabla 1 se muestran las propiedades constantes del objeto *Tren*:

Dato	Valor	Unidades
longitud	*89,16*	*m*
masa	*159,7*	*Tn*
a	*3003,957*	*N*
b	*31,94*	*N/(km/h)*
c	*0,67074*	*N/(km/h)^2*

Tabla 1: propiedades constantes del tren

Una vez definidos los objetos, es necesario diseñar el modelo del ATO (*Automatic Train Operation*), el cual es un modo de conducción que gobierna al tren de forma automática. El objetivo será calcular la aceleración y, consecuentemente, la velocidad y la posición a través de las fórmulas cinemáticas en cada instante de tiempo [1].

Por un lado, se diseñará el modelo de acelerado del tren, lo que se resume en el cálculo de la aceleración cuando éste debe aumentar su velocidad. Dicha aceleración se calcula como la suma de la fuerza que tiene que ejercer el motor, la fuerza de resistencia al avance y la fuerza por la pendiente de la vía, con sus respetivos signos, dividida entre la masa del tren.

Por otro lado, el tren tiene que frenar cuando se aproxima a un tramo de vía en el que se reduce la velocidad máxima o cuando se está acercando a una estación. Será entonces necesario el diseño del modelo de frenado, el cual se basa en el cálculo de una curva de frenado desde el punto donde existe la reducción de velocidad hasta donde se encuentra el tren. La velocidad inicial desde donde parte la curva de frenado será la nueva velocidad máxima o cero si el tren se está aproximando a una estación.

Se comparará la velocidad máxima debido al frenado con la velocidad que lleva el tren y, si ésta última es mayor, se modificará la fuerza que tiene que ejercer el motor para que el tren sea fiel a la curva de frenado. En la Ilustración 2 se muestra un esquema de un posible recorrido, donde la curva verde representa las velocidades máximas de la vía, la curva negra representa la curva de frenado y la curva azul representa la velocidad del tren:

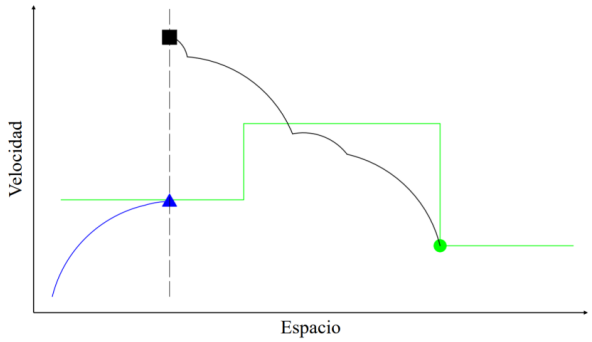

Ilustración 2: esquema de curva de frenado

Hay que tener en cuenta, que en caso de que el tren se haya detenido en una estación, este debe de respetar el tiempo de parada, el cual se ha establecido en 15 segundos.

Una vez que el tren circula correctamente por la vía, hay que tener en cuenta que cuando llega al final de ésta, debe invertir su marcha. A esta forma de operación se le conoce como

operación en *carroussel*. El modelo de cambio de vía se basa en la activación de una variable *flag* para indicar que se ha llegado al final de la vía y que el tren debe de pasar de circular en la vía 1 a la vía 2.

Por último, cuando no se quiere que el tren circule a marcha tendida (su velocidad máxima), se emplean marchas de regulación, las cuales son útiles para combatir los retrasos y aumentar la eficiencia de la circulación. Las marchas de regulación implementadas en el proyecto han sido:

- Regulación por velocidad. Se impone una velocidad límite en la vía menor que la máxima.
- Regulación por deriva – remotor. Se impone una velocidad deriva consigna y una velocidad remotor consigna. El tren acelera hasta alcanzar la velocidad de deriva y en ese momento, la fuerza ejercida por el motor es cero, consecuentemente cuando el tren decelere y llegue a la velocidad de remotor, volverá a existir fuerza tractora repitiéndose el proceso descrito de manera continuada.

Hay que tener en cuenta que el uso de regulación por velocidad o deriva – remotor viene impuesto por las marchas disponibles del tren que indican en qué tramos se utiliza cada una de ellas. Dichas marchas son la marcha 0 (M0), la marcha 1 (M1), la marcha 2 (M2) y la marcha 3 (M3), siendo M0 la más rápida y M3 la más lenta.

El modelo de marchas de regulación, permitirá que el tren pueda utilizar la regulación por velocidad y la regulación por deriva – remotor.

Modelo del simulador multi – tren

Una vez que se ha diseñado el modelo que consigue que un tren circule por las vías, es necesario implementar el modelo de la simulación multi – tren, es decir, varios trenes deben circular por la misma vía, al igual que ocurre en una línea de metro real.

Para ello se ha creado un vector de objetos *Tren* y cada iteración de tiempo se simulará para cada posición de ese vector, o lo que es lo mismo, para cada tren.

En esta sección se encuentra uno de los modelos clave para el correcto funcionamiento del regulador: el modelo de frenado por proximidad entre trenes. Dicho modelo seguirá el mismo principio que el de frenado normal: para un tren se creará una curva de freno desde el tren que va delante hasta el punto en el que se encuentra, imponiendo que la velocidad de comienzo de la curva de freno sea nula. Habrá que tener en cuenta que, si el tren de delante se encuentra detenido, la cabeza del que va detrás no frena en la cola del siguiente, sino que es establecerá una distancia de seguridad entre cola y cabeza.

Modelo de regulación del tráfico

Por último, los trenes de una línea de metro están sujetos a horarios. Estos horarios marcan el momento en el que tiene que salir y entrar cada tren de cada estación, por lo tanto, es necesario crear un modelo de regulación que garantice el cumplimiento de los mismos.

Para diseñar el modelo de regulación automática del tráfico, lo primero es fijar el intervalo de tiempo que se quiere entre trenes, es decir, cuánto tiempo pasa desde que sale un tren de una estación hasta que llega otro; en el caso de este proyecto, se ha fijado un tiempo de intervalo de 210 s (3.5 minutos). Una vez se ha decidido el intervalo, hay que escoger la cantidad de trenes que circularán por la vía. En el apartado 7.1 Cálculo del Número de Trenes Empleados, se ha razonado y llegado a la conclusión de que el número óptimo de trenes para un intervalo de 210 s en las vías del proyecto es 15.

A continuación, se debe diseñar la salida escalonada de los trenes desde el origen respetando el intervalo de tiempo que se ha impuesto y calcular el horario nominal de cada uno de ellos. El horario vendrá dado por los tiempos de salida de cada estación que marca un tren cuando circula con su marcha 1 (M1). De esta manera, se calculará el horario del primer tren en salir y el del resto se calculará como el horario del primer tren más *j* veces el intervalo; siendo *j* el número del tren.

En el caso de producirse un retraso, habiéndose calculado previamente el tiempo que se gana o se pierde entre estaciones si se aplican las diferentes marchas (M0, M2, M3), se obtiene una corrección, la cual se traduce en una orden de marcha para que el tren aumente la velocidad si va retrasado, o la reduzca si va adelantado. En el Apartado 7.5 Modelo del ATR: Cálculo de la Corrección y Consigna de Aplicación de Marcha, se explica cómo se obtiene la corrección y cómo se traduce a una orden de marcha.

Finalmente, se ha tenido en cuenta la influencia de las perturbaciones que pueden provocar los pasajeros en la subida y bajada del tren en las estaciones. Para ello, el tiempo de parada procederá de un número aleatorio generado de una distribución lognormal de media 15 segundos y desviación típica 3 segundos. Según el artículo *Statistical dwell time model for metro lines* [2], se ha demostrado que es la distribución más acertada para este tipo de incidencias.

4. Resultados

Para comenzar, en la Ilustración 3 se demuestra que un solo tren circula por las vías de manera correcta, simulando el recorrido del mismo partiendo del origen de la vía 1:

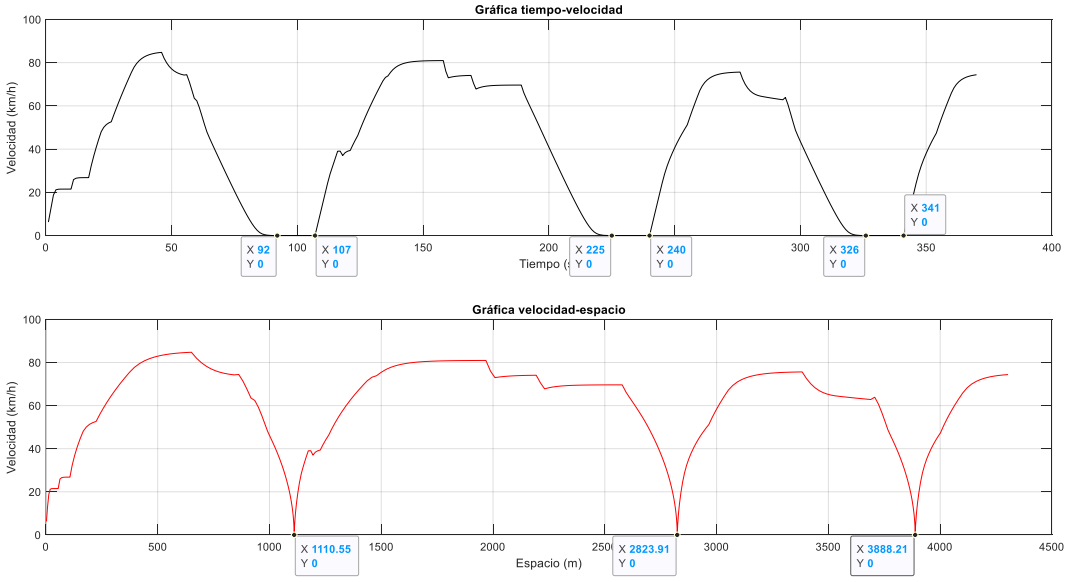

Ilustración 3: simulación de la circulación de un tren

Por otro lado, uno de los resultados más importantes obtenidos es cómo un tren frena porque se está acercando demasiado al siguiente. Para comprobarlo, se ha simulado la circulación de dos trenes; uno de ellos (tren 1) partiendo desde el origen de la vía 1 con un tiempo de parada de 18 s y el otro (tren 2) partiendo de la primera estación de la vía 1 con un tiempo de parada de 50 s.

De esta forma, el tren 1 se irá aproximando al tren 2 viéndose obligado a reducir su marcha e incluso a detenerse por completo. En la Ilustración 4 se muestra la simulación realizada:

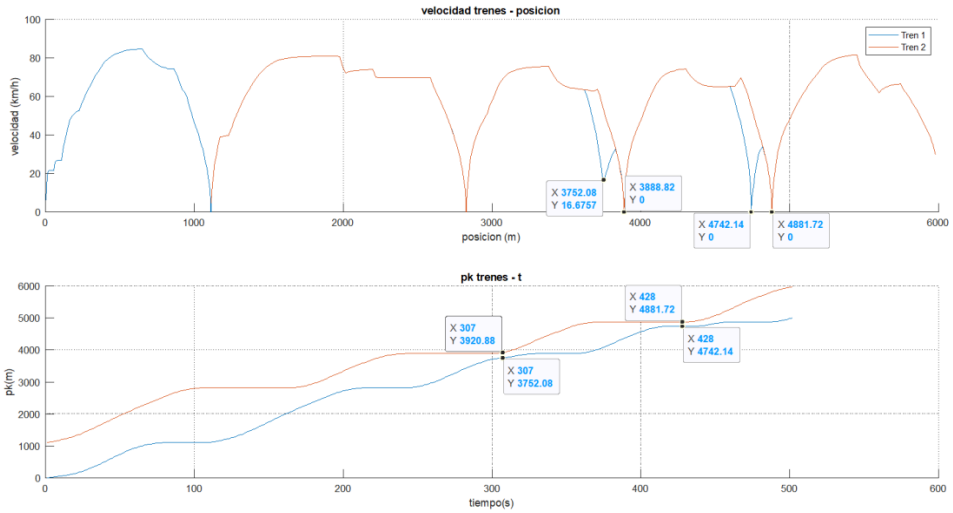

Ilustración 4: simulación de frenado por proximidad entre trenes

Cabe destacar que se ha impuesto una distancia de seguridad entre la cola del tren de delante y la cabeza del tren de detrás de 50 m. Por lo tanto, la distancia mínima entre trenes será de 50 metros más 89.16 metros (longitud del tren), es decir, de 139.16 metros. En la simulación, cuando el tren 1 se ve obligado a frenar entre la estación 3 y 4, se puede comprobar que la distancia entre trenes es de 139.58 metros, respetando así la distancia de seguridad mínima.

Por último, el funcionamiento del regulador automático de tráfico se demuestra con una simulación en la que se le impone a un tren una parada de 5 minutos adicionales en una estación, produciendo así un retraso del mismo tiempo en el tren.

En la Ilustración 5 se representa la evolución de los retrasos en el horario tanto del tren afectado como de los dos que van delante de él. De la misma forma, se compara el modelo que tiene en cuenta las incidencias en el tiempo de parada que provocan los pasajeros con el que no las tiene en cuenta:

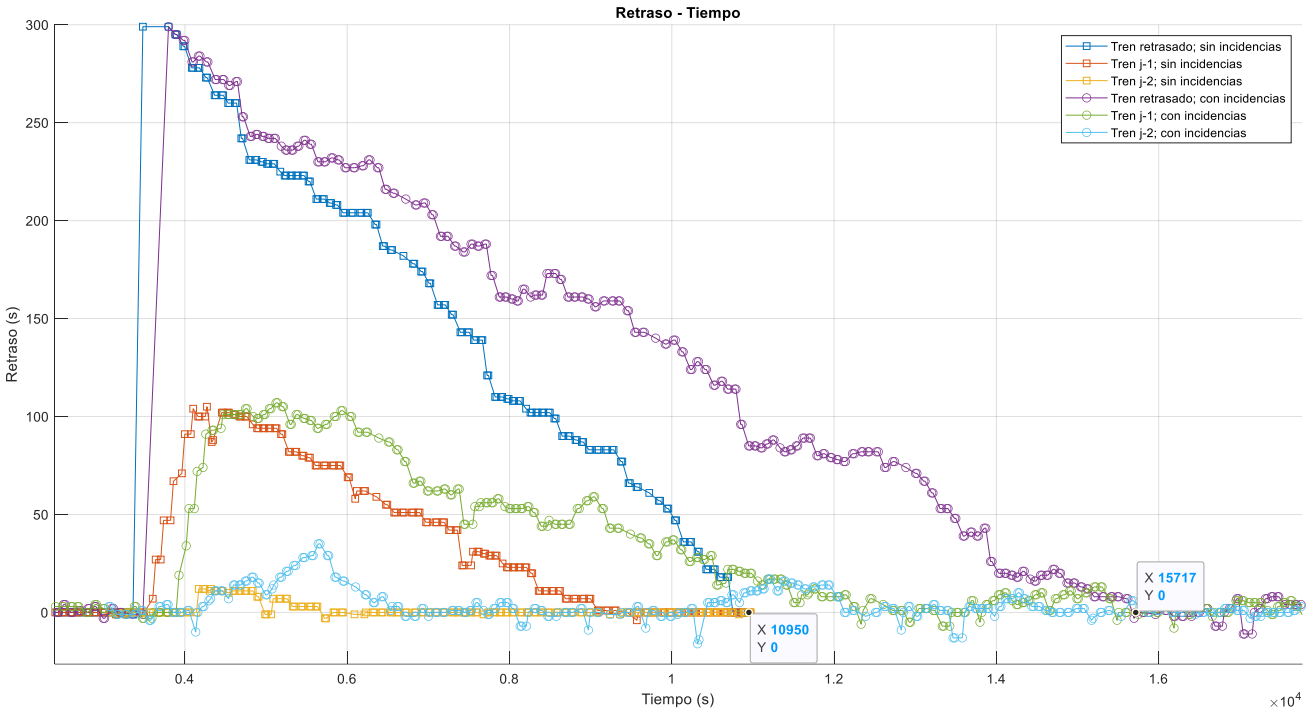

Ilustración 5: comparación de simulación del modelo de tráfico

El modelo de regulación automática de tráfico es capaz de llevar los errores de horario de los trenes a cero. Cabe destacar cómo los trenes que van delante del tren que ha sufrido el retraso reducen su marcha produciendo un retraso en sus respectivos horarios. Esto se debe a que el regulador también le da peso al cumplimiento del tiempo de intervalo entre trenes: el tren que ha sufrido el retraso se encuentra detenido mientras que el que va delante continúa su marcha, aumentando así el tiempo de intervalo entre ellos. Será cuando el regulador le dé más peso al retraso horario que al error en el intervalo cuando se empiece a llevar a cero el error en los horarios de los trenes que van delante del afectado.

Es importante destacar que, en el caso de tener en cuenta las incidencias de los pasajeros en los tiempos de parada, la eliminación del retraso horario lleva más tiempo. Esto se debe a que el tiempo que se ahora con las marchas, es también empleado para recuperar el tiempo perdido en cada parada.

5. Referencias

[1] M. Domínguez, A. Fernández-Cardador, A.P. Cucala, P. Lukaszewicz. Optimal design of metro automatic train operation speed profiles for reducing energy consumption. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit. Vol. 225, nº. 5, pp. 463 - 474, Septiembre 2011. [Online: Agosto 2011] JCR JCR: 0.436 Q3 (2011); 2.359 Q2 (2020) - SJR: Q2 (2011); 0.659 Q2 (2020)

[2] I. Martínez, B. Vitoriano, A. Fernández & A. P. Cucala. *Statistical dwell time model for metro lines.* Source: WIT Transactions on The Built Environment, Vol 96, 2007 WIT Press.

MULTI – TRAIN SIMULATOR OF A METRO LINE FOR AUTOMATIC TRAFFIC REGULATION TESTS

Author: Cidoncha González, Álvaro.
Supervisors: Fernández Rodríguez, Adrián; Fernández Cardador, Antonio.
Collaborating Entity: ICAI – Comillas Pontificial University.

ABSTRACT

The main objective of this project is the development of a twin model of a metro line for the testing and analysis of train traffic regulation strategies in Matlab programming language. Particularly, models will be developed to realistically replicate the movement, driving and energy consumption of a train. The line for which said twin model is designed, corresponds to an important metro line in Spain.

In addition, models that simulate the behavior of several trains on a line will be implemented, including the influence of stop times, of a stochastic nature, and the interference between some trains and others, which will be conditioned by the signaling system.

Finally, an efficient and functional traffic regulator is applied, which is capable of maintaining a regular interval between trains and correcting deviations in the commercial speed.

Keywords: Traffic model; Automatic traffic tegulator; ATO; Circulation simulation.

1. Introduction

On a modern metro line, train traffic is regulated from the central command post, sending each train the driving instructions it must perform. For this, it is necessary to design an efficient and precise automatic traffic regulation model.

Additionally, the implementation of improvements in the regulation of traffic on the metro lines supposes an increase in the quality of the service, both in maintaining commercial speed and in regularity and, therefore, an increment of the use of this transport by passengers.

Consequently, the increase in the use of the metro by citizens would mean a reduction in greenhouse gas emissions, since the use of the railway in large cities is the most efficient method to reduce the pollution resulting from transport.

2. Proyect definition

To create the automatic traffic regulation model for trains of a metro line, it will be necessary to design all the models that govern the operation and movement of a single train. Once it has been verified that a single train is capable of running on the tracks, several trains will be implemented on the same line to finally design the automatic traffic regulation model.

As an introduction, Figure 1 presents a possible scheme of the project structure:

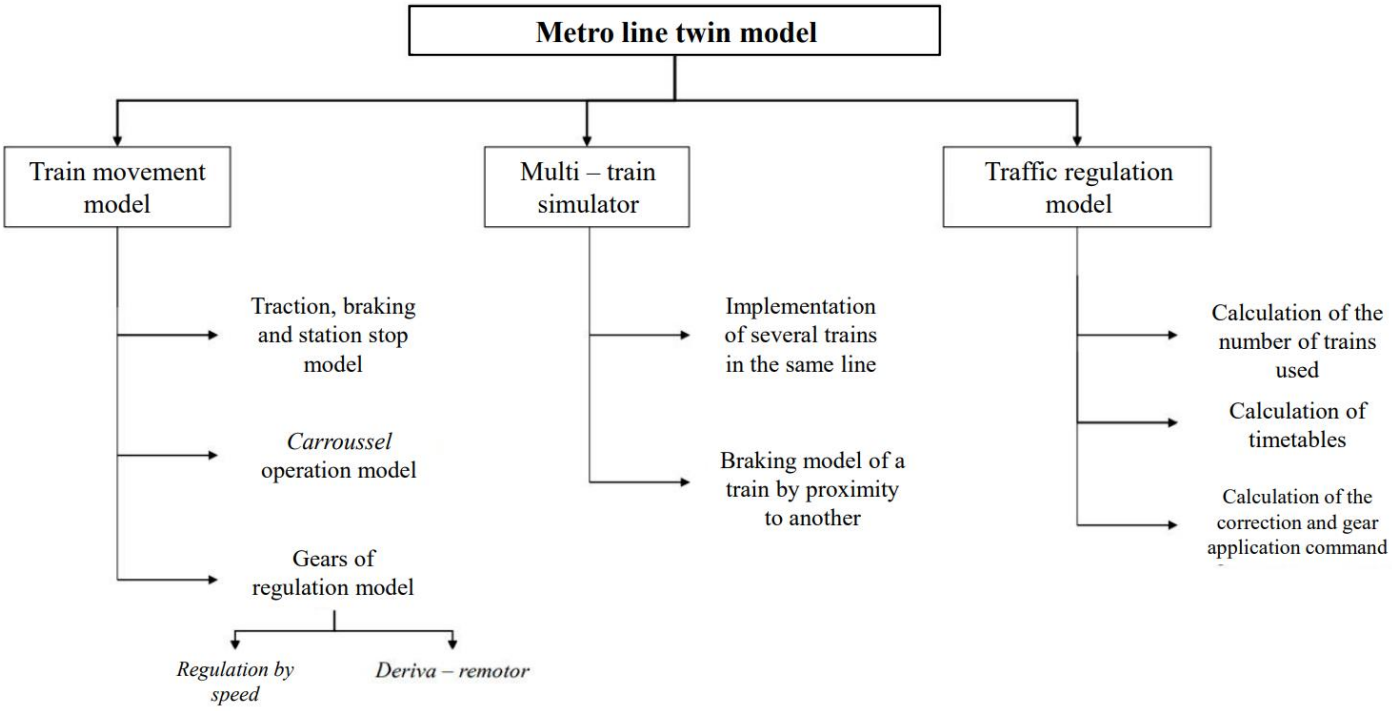

Figure 6: project structure

3. Model description

As mentioned above, in order to obtain the twin model of a metro line and be able to carry out automatic traffic regulation tests, the models of the following parts must be completed:

- Model of the movement of a train.
- Multi-train simulator model.
- Traffic regulation model.

Train movement model

To begin the code design of the twin model in Matlab, the first thing to do is to define the objects of the train and the tracks: track 1 when the train travels the track in one direction and track 2 when it travels in the opposite direction.

In the *Train* object its constant properties will be stored, such as the mass, the length and the coefficients *a*, *b*, and *c* involved in the resistance force to the advance and its variable properties, that is, the position and speed of the train that it is carrying in each time iteration. On the other hand, in the *Linea* and *Linea2* objects, the constant properties of tracks 1 and 2 will be defined, which are the maximum speed that the train can carry in each section, the slope of the track at each point and the kilometer points of the stations. Table 1 shows the constant properties of the Train object:

Property	Value	Units
Length	*89,16*	*m*
Mass	*159,7*	*Tn*
a	*3003,957*	*N*
b	*31,94*	*N/(km/h)*
c	*0,67074*	*N/(km/h)^2*

Table 2: train constant properties

Once the objects have been defined, it is necessary to design the *ATO* (Automatic Train Operation) model, which is a driving mode that governs the train automatically. The objective will be to calculate the acceleration and, consequently, the velocity and the position through the kinematic formulas at each instant of time [1].

On the one hand, the traction model of the train will be designed, which is summarized in the calculation of the acceleration when it must increase its speed. Said acceleration is calculated as the sum of the force that the engine has to exert, the force of resistance to advance and the force due to the slope of the track, with their respective signs, divided by the mass of the train.

On the other hand, the train has to brake when it approaches a section of track where the maximum speed is reduced or when it is approaching a station. It will then be necessary to design the braking model, which is based on the calculation of a braking curve from the point where the speed reduction exists to where the train is. The initial speed from where the braking curve starts will be the new maximum speed or zero if the train is approaching a station.

The maximum speed due to braking will be compared with the speed of the train and, if the latter is greater, the force that the motor must exert so that the train is true to the braking curve will be modified. Figure 2 shows a scheme of a possible route, where the green curve represents the maximum speeds of the track, the black curve represents the braking curve and the blue curve represents the speed of the train:

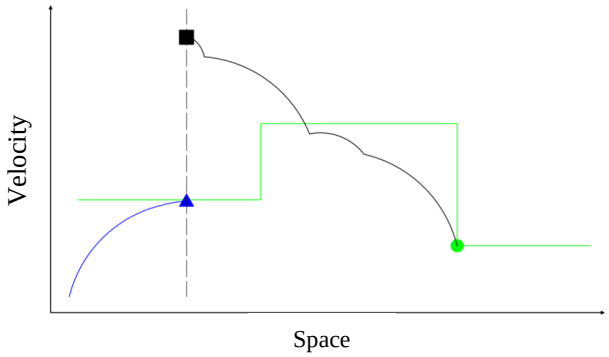

Figure 7: braking curve scheme

It is necessary to remind that if the train has stopped at a station, it must respect the stop time, which has been established at 15 seconds.

Once the train runs correctly along the track, it must be taken into account that when it reaches the end of it, it must reverse its course. This form of operation is known as carrousel operation. The track change model is based on the activation of a flag variable to indicate that the end of the track has been reached and that the train must go from running on track 1 to track 2.

Lastly, when you do not want the train to run at its maximum speed, regulation gears are used, which are useful to combat delays and increase circulation efficiency. The regulation gears implemented in the project have been:

- **Regulation by speed**. A speed limit is imposed on the road which is minor than the maximum speed.
- **Regulation by *deriva – remotor***. A setpoint *deriva* speed and a setpoint *remotor* speed are imposed. The train accelerates until it reaches *deriva* speed and at that moment, the force exerted by the motor is zero, consequently when the train decelerates and reaches *remotor* speed, there will be a tractive force again, repeating the process described continuously.

It must be taken into account that the use of regulation by speed or *deriva – remotor* is imposed by the available gears of the train that indicate in which sections each of them is used. These gears are gear 0 (M0), gear 1 (M1), gear 2 (M2), and gear 3 (M3), with M0 being the fastest and M3 being the slowest.

The regulation gear model will allow the train to use regulation by speed and regulation by *deriva – remotor*.

Multi – train simulator model

Once the model that allows a train to circulate on the tracks has been designed, it is necessary to implement the multi-train simulation model, that is, several trains must circulate on the same track, as happens on a real railway line.

For this, a vector of *Train* objects has been created and each iteration of time will be simulated for each position of that vector, or what is the same, for each train.

This section contains one of the key models for the correct operation of the regulator: the proximity braking model between trains. This model will follow the same principle as the normal braking model: for a train, a brake curve will be created from the train in front to the point where it is, imposing that the start speed of the brake curve is zero. It will be necessary to take into account that, if the train in front is stopped, the head of the one behind does not brake in the tail of the following one, but it will establish a safety distance between tail and head.

Traffic regulation model

Finally, the trains of a metro line must follow their timetables. These schedules mark the time when each train has to leave and enter each station, therefore, it is necessary to create a regulation model that guarantees compliance with them.

To design the automatic traffic regulation model, the first thing is to set the desired time interval between trains, that is, how long it takes from the time a train leaves a station until another arrives; In the case of this project, an interval time of 210 s (3.5 minutes) has been set. Once the interval has been decided, you have to choose the number of trains that will run on the track. In section 7.1 Calculation of the Number of Trains Used, it has been reasoned and reached the conclusion that the optimal number of trains for an interval of 210 s on the project tracks is 15.

Next, the staggered departure of the trains from the origin must be designed respecting the time interval that has been imposed and the nominal timetable of each of them must be calculated. The timetable will be given by the departure times of each station marked by a train when it circulates with gear 1 (M1). In this way, the time of the first train to leave will be calculated and the rest will be calculated as the time of the first train plus j times the interval; where j is the number of the train.

In the event of a delay, having previously calculated the time gained or lost between stations if the different gears are applied (M0, M2, M3), a correction is obtained, which translates into a gear command so that the train speeds up if it is late, or slows down if it is ahead. Section 7.5 Calculation of the Run Application Correction and Setpoint explains how the correction is obtained and how it is translated into a run command.

Finally, the influence of the disturbances that passengers can cause when getting on and off the train at the stations has been taken into account. To do this, the stop time will come from a random number generated from a lognormal distribution with a mean of 15 seconds and a standard deviation of 3 seconds. According to the article Statistical dwell time model for metro lines [2], it has been shown to be the most accurate distribution for this type of incident.

4. Results

To begin with, Figure 3 shows that a single train circulates correctly along the tracks, simulating its route starting from the origin of track 1:

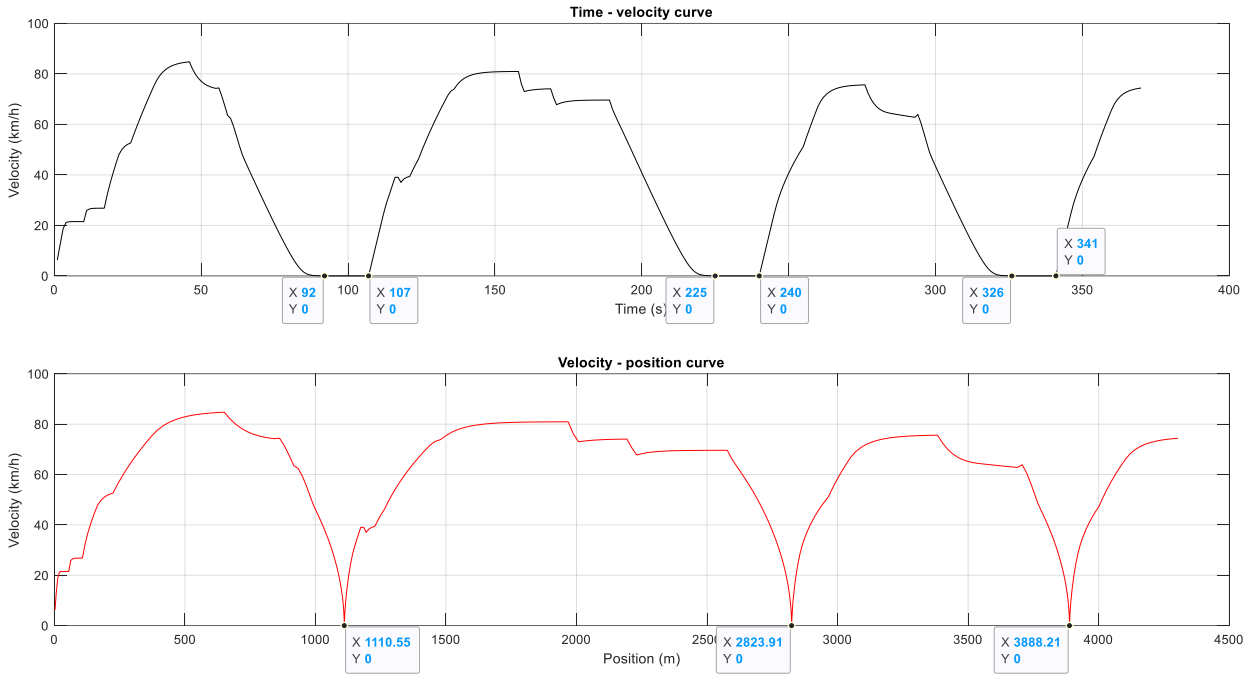

Figure 8: circulation simulation of a train

On the other hand, one of the most important results obtained is how a train brakes because it is getting too close to the next one. To verify this, the circulation of two trains has been simulated; one of them (train 1) starting from the origin of track 1 with a stopping time of 18 s and the other (train 2) starting from the first station on track 1 with a stopping time of 50 s.

In this way, train 1 will gradually approach train 2, being forced to slow down and even come to a complete stop. Figure 4 shows the simulation carried out:

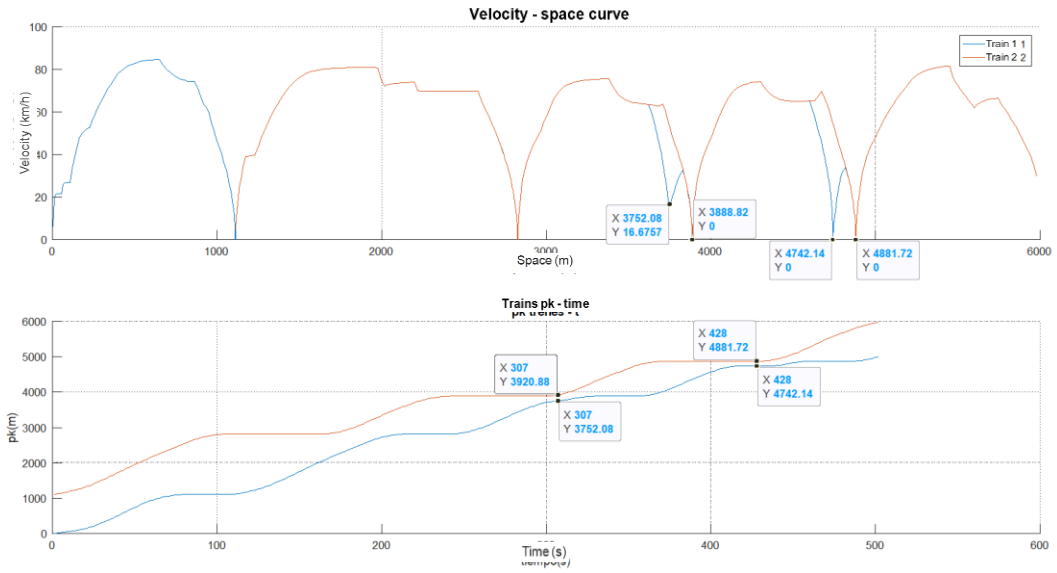

Figure 9: simulation of proximity braking between trains

It should be noted that a safety distance between the tail of the train in front and the head of the train behind of 50 m has been imposed. Therefore, the minimum distance between trains will be 50 meters plus 89.16 meters (train length), that is, 139.16 meters. In the simulation, when train 1 is forced to brake between stations 3 and 4, it can be verified that the distance between trains is 139.58 meters, thus respecting the minimum safety distance.

Finally, the operation of the automatic traffic regulator is demonstrated with a simulation in which a stop of 5 additional minutes is imposed on a train at a station, thus producing a delay of the same time in the train.

Figure 5 shows the evolution of the delays in the schedule of both the affected train and the two that go before it. In the same way, the model that takes into account the aleatory incidents in the stop time caused by passengers is compared with the one that does not take them into account:

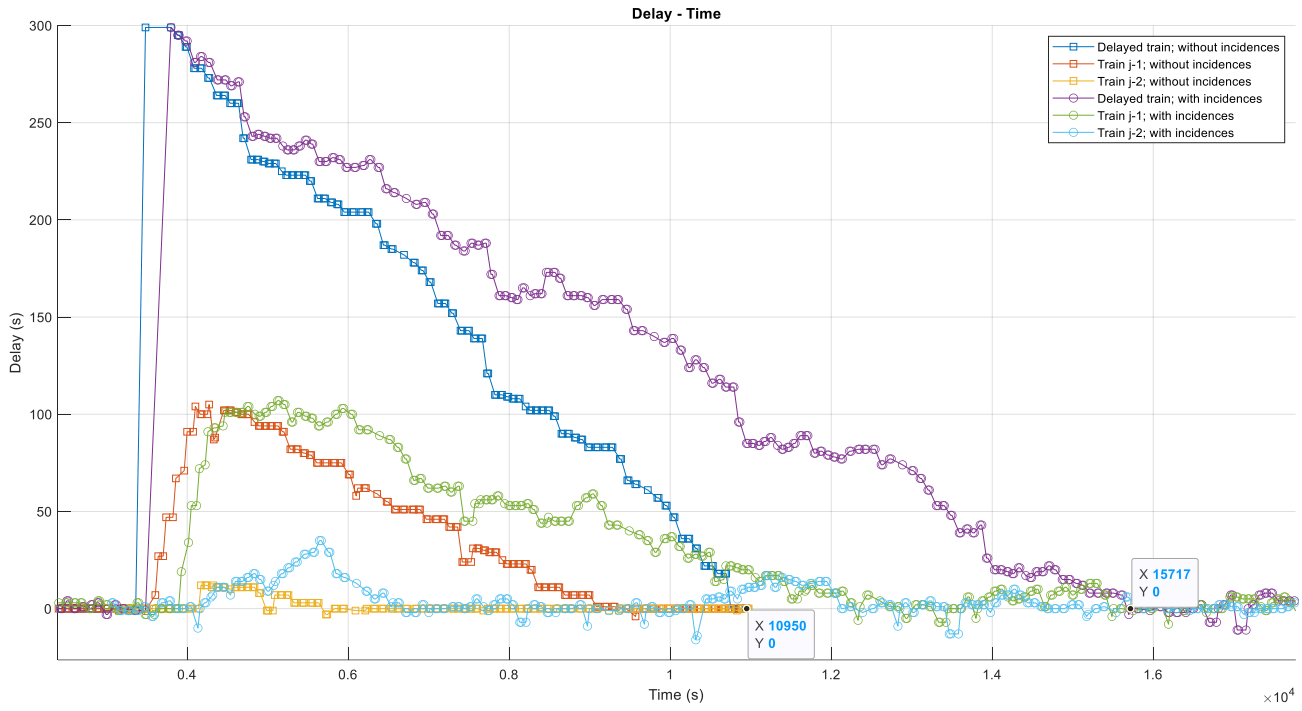

Figure 10: traffic model simulation comparation

The automatic traffic regulation model is capable of bringing train schedule errors to zero. It is worth noting how the trains that go ahead of the train that has suffered the delay slow down, producing a delay in their respective schedules. This is due to the fact that the regulator also gives weight to compliance with the interval time between trains: the train that has suffered the delay is stopped while the one in front continues its march, thus increasing the interval time between them. It will be when the regulator gives more weight to the time delay than to the error in the interval when the error in the timetables of the trains that go ahead of the affected one begins to be zeroed out.

It is important to note that, in the case of taking into account the incidences of passengers in stop times, the elimination of the time delay takes longer. This is because the time saved with the marches is also used to recover the time lost at each stop.

5. References

[1] M. Domínguez, A. Fernández-Cardador, A.P. Cucala, P. Lukaszewicz. Optimal design of metro automatic train operation speed profiles for reducing energy consumption. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit. Vol. 225, nº. 5, pp. 463 - 474, Septiembre 2011. [Online: Agosto 2011] JCR JCR: 0.436 Q3 (2011); 2.359 Q2 (2020) - SJR: Q2 (2011); 0.659 Q2 (2020)

[2] I. Martínez, B. Vitoriano, A. Fernández & A. P. Cucala. *Statistical dwell time model for metro lines.* Source: WIT Transactions on The Built Environment, Vol 96, 2007 WIT Press.

Índice de la memoria

Capítulo 1. Introducción 6

Capítulo 2. Descripción de las Tecnologías 7

Capítulo 3. Estado de la Cuestión 8

Capítulo 4. Definición del Trabajo 9

4.1 Justificación y Objetivos 9

4.2 Metodología 9

4.3 Planificación y Estimación Económica 10

Capítulo 5. Desarrollo del Proyecto: Movimiento de un Tren 13

5.1 Definición de Objetos y Datos de la Línea de Metro 13

5.2 Modelo de Acelerado, Frenado y Parada en Estación 15

5.2.1 Modelo de Tracción 16

5.2.2 Modelo de Frenado 20

5.2.3 Modelo de Parada en Estación 24

5.3 Modelo de Operación en Carrousel 27

5.4 Modelo de Marchas de Regulación 29

5.4.1 Modelo de Regulación por Velocidad 31

5.4.2 Modelo de Regulación Deriva – Remotor 33

Capítulo 6. Desarrollo del Proyecto: Simulador Multi – Tren 36

6.1 Implementación de Varios Trenes en la Misma Vía 36

6.2 Modelo del ATP: Frenado de un Tren por Proximidad a Otro 37

6.3 Caso de Estudio: Parada de un Tren por Proximidad 39

Capítulo 7. Desarrollo del Proyecto: Modelo de Regulación de Tráfico 41

7.1 Cálculo del Número de Trenes Empleados 42

7.2 Cálculo del Horario 44

7.3 Salida Escalonada de los Trenes 49

7.4 Cálculo del Tiempo Ganado o Perdido con M0, M2 y M3 51

7.5 Modelo del ATR: Cálculo de la Corrección y Consigna de Aplicación de Marcha 55

7.6 Retrasos en Estación por Subida y Bajada de Pasajeros 60

Capítulo 8. Análisis de Resultados Y Conclusiones 64

8.1 Simulación sin Incidencias por Pasajeros..... 64

8.2 Efecto de Incidencias Aleatorias por Pasajeros en la Regulación del Tráfico 69

8.3 Conclusiones 72

Capítulo 9. Bibliografía..... 74

ANEXO I 75

Programa Principal y Clases 75

Programa Principal main..... 75

Clase Tren 76

Clase Línea..... 92

Clase Línea 2..... 93

Funciones..... 95

Búsqueda de Pendiente..... 95

Búsqueda de la Siguiente Reducción de Velocidad..... 95

Curva de Freno 96

Función Frenado..... 96

Velocidad No Superable por Estación..... 97

Función de Simulación de Tráfico 98

Curva de Freno Debido a Trenes Previos..... 100

Función para el Cálculo de la Corrección..... 101

Aplicación de Marcha 101

Tiempo de Parada Extra 103

Distribución Lognormal 103

Índice de figuras

Ilustración 1: resultados de la simulación de acelerado 20
Ilustración 2: aclaración de la función de búsqueda de la siguiente reducción de velocidad máxima 21
Ilustración 3: ejemplo de curva de freno 22
Ilustración 4: simulación de 1km de recorrido para un solo tren desde el comienzo de la vía 24
Ilustración 5: simulación de 4.3 km del modelo de parada en estación 26
Ilustración 6: perfil de velocidades del tren a marcha tendida por las vías 1 y 2 28
Ilustración 7: simulación inversión de marcha 28
Ilustración 8: ejemplo de regulación por velocidad 29
Ilustración 9: ejemplo de regulación por deriva – remotor 30
Ilustración 10: simulación ejemplo de regulación por velocidad 33
Ilustración 11: simulación a marcha tendida 33
Ilustración 12: ejemplo de la característica "dientes de sierra" 35
Ilustración 13: caso de estudio de frenado por proximidad entre dos trenes 39
Ilustración 14: simulación con M1 para calcular el tiempo 'T' 43
Ilustración 15: cálculo del horario con M1 45
Ilustración 16: simulación de puesta en servicio de los 15 trenes 51
Ilustración 17: cálculo de los tiempos del tren con M0 52
Ilustración 18: cálculo de los tiempos del tren con M2 53
Ilustración 19: cálculo de los tiempos del tren con M3 53
Ilustración 20: esquema de regulación del tráfico 59
Ilustración 21: retraso producido por incidencias de pasajeros 61
Ilustración 22: velocidad frente al tiempo; simulación de incidencia 62
Ilustración 23: frenado de trenes 65
Ilustración 24: evolución del RH; trenes j, j-1, j-2, j-3 66

Ilustración 25: retraso de los trenes j, j+1, j+2, j+3 67
Ilustración 26: resultados conjuntos para h=80 68
Ilustración 27: comparación h=80 vs h=45 69
Ilustración 28: evolución de los retrasos con incidencias provocadas por pasajeros 70
Ilustración 29: comparación de la evolución en los retrasos cuando hay incidencias por pasajeros y cuando no 71

Índice de tablas

Tabla 1: cronograma cuatrimestre 1 11
Tabla 2: cronograma del cuatrimestre 2 11
Tabla 3: tabla de costes 12
Tabla 4: propiedades constantes del tren 14
Tabla 5: velocidades máximas para la simulación 19
Tabla 6: pendientes para la simulación 19
Tabla 7: datos para simulación de regulación por velocidad 32
Tabla 8: resultado de los horarios para cada tren 48
Tabla 9: tiempos ganados o perdidos entre estaciones con cada marcha 54
Tabla 10: tiempos de parada; simulación de incidencia 63

Capítulo 1. INTRODUCCIÓN

En una línea de metro moderna, el tráfico de trenes se regula desde el puesto central de mando enviando a cada tren la consigna de conducción que debe realizar. El objetivo principal de este proyecto es el desarrollo de un modelo gemelo (twin model) de una línea de metro para el ensayo y análisis de estrategias de regulación de tráfico de trenes. La línea para la que se diseña dicho modelo gemelo, corresponde a una importante línea de metro de España, la cual consta de 18 estaciones y tiene una longitud de 14.6 km aproximadamente.

Además, se utilizarán los resultados del modelo gemelo de la línea para analizar distintas estrategias de regulación del tráfico. La solución tecnológica para conseguir analizar dichas estrategias, reside en la implantación de un código de programación eficiente, donde el usuario pueda simular el recorrido de los trenes y calcular el valor de los indicadores que miden la calidad del servicio.

Adicionalmente, el desarrollo de un modelo gemelo para la regulación automática del tráfico eficiente y fiable permite el desarrollo y ensayo de algoritmos de regulación para mejorar la calidad del servicio, lo que se traduce en un mayor número de pasajeros. De esta forma, se conseguiría reducir la emisión de gases de efecto invernadero en las grandes urbes, ya que el uso del ferrocarril en las ciudades es el método más eficiente de reducir la polución resultante del transporte.

Por otro lado, el proyecto tiene el objetivo adicional de ensayar y configurar un algoritmo de regulación del tráfico que tenga en cuenta la situación de retrasos en la línea. Dicho algoritmo busca mejorar la calidad del servicio para que el transporte por ferrocarril sea más atractivo para los pasajeros, lo que se traduce en un modo de transporte más sostenible. Para esto último, son necesarios simuladores y así poder proponer y analizar las estrategias antes de su puesta en servicio.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Hay que tener en cuenta que el proyecto en su totalidad se basará en el diseño de un código de programación en Matlab de elaboración propia que sea capaz de cumplir los objetivos explicados en el Capítulo 1. Introducción.

De la misma forma, es necesario introducir y explicar de manera simplificada el funcionamiento de un regulador automático del tráfico. La regulación del tráfico se realiza desde el puesto central, donde un sistema automático calcula las desviaciones de horario e intervalo de los trenes. De la misma forma, trata de corregir las posiciones de los mismos enviando al equipo de conducción ATO (Automatic Train Operation) las consignas de velocidad y, si es necesario, consignas de retenciones en la estación.

Las primeras llegan a los trenes a través de balizas a la salida de estación o, en líneas con señalización CBTC (Control de Trenes Basado en Telecomunicaciones en español), a través de comunicaciones inalámbricas en cualquier instante. Las segundas se envían la señalización de salida de estación o a dispositivos específicos que indican al maquinista el momento en el que el tren debe de abandonar la estación.

Capítulo 3. ESTADO DE LA CUESTIÓN

Actualmente, existen tecnologías que permiten realizar una efectiva regulación automática del tráfico en las líneas de metro. En primer lugar, el sistema ATP [1] (*Automatic Train Protection*) establece una supervisión continuada del tren evitando así los alcances entre trenes o que no se supere la velocidad máxima de la vía. En segundo lugar, al ATP se le puede implementar el ATO (*Automatic Train Operation*). Este sistema controla el freno y la tracción del tren permitiendo una conducción cómoda para los pasajeros. Adicionalmente, el ATR (*Automatic Train Regulation*) permite establecer distintas órdenes de marcha y modificar los tiempos de parada en estación para respetar los horarios e intervalos.

Un ejemplo de la combinación de estas tecnologías, es el SIRAT (*Sistema de Regulación Automática del Tráfico*) implementada en el metro de Madrid. Fue desarrollada originalmente por el Instituto de Investigación Tecnológica (IIT) de la Universidad Pontificia Comillas en conjunto con la empresa madrileña *Dimetronic* [2]. Dicha tecnología sigue hoy en servicio.

El SIRAT permite regular automáticamente la frecuencia de paso de los trenes por las estaciones actuando sobre las velocidades que deben de llevar y sobre los tiempos de parada, evitando así la acumulación de convoyes en un sector de la línea y los retrasos que esto pueda ocasionar.

La implementación de este modelo de regulación del tráfico supuso inicialmente una mejora del 43% en la regularidad y una reducción del 18% en el consumo de energía.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN Y OBJETIVOS

Con este proyecto se pretende conseguir un modelo de regulación automática del tráfico que tenga presente todo lo mencionado anteriormente, es decir, se incorporará a su vez el modelo del ATP, del ATO y del ATR y que, a su vez, tenga en cuenta los sistemas de señalización modernos. De esta manera, se conseguirá una regulación del tráfico eficiente y precisa que sea capaz de respetar el intervalo de tiempo entre trenes y elimine los retrasos cuando éstos se produzcan.

Adicionalmente, se pretende que el código diseñado para este proyecto sirva como base para desarrollar en un futuro un regulador para CBTC que permita una continua comunicación tren – tierra para poder modificar la marcha en cualquier punto de la vía, así como para la programación de nuevos algoritmos de regulación basados en la optimización y machine learning.

4.2 METODOLOGÍA

El código en su totalidad será diseñado en Matlab. Se desarrollarán modelos detallados de simulación de la dinámica de un tren, del sistema de conducción automática ATO, de la interacción de varios trenes circulando en la misma vía y del sistema de regulación del tráfico. Posteriormente se integrarán todos los modelos. El correcto funcionamiento de cada modelo ha sido verificado utilizando casos ejemplo a partir de datos reales de una línea de metro.

A modo de introducción, en la Ilustración *A* se presenta un posible esquema de la estructura del proyecto:

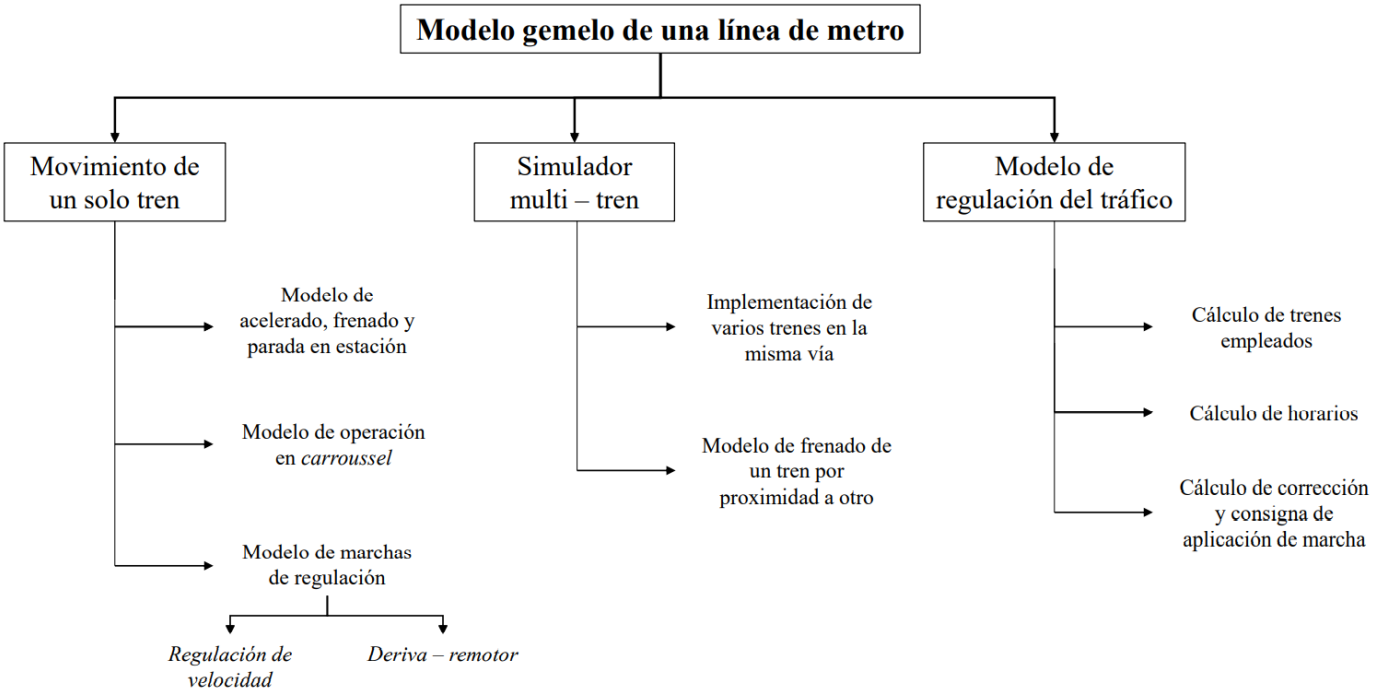

Ilustración A: esquema de la estructura del proyecto

4.3 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

En la Tabla 1 y la Tabla 2 se muestra la organización que se ha llevado a lo largo del curso lectivo durante el primer y segundo cuatrimestre respectivamente:

Actividad	Meses (semanas), cuatrimestre 1														
	Septiembre				Octubre				Noviembre				Diciembre		
Diseño del modelo del ATO															
Diseño del modelo del ATP															
Implementación de varios trenes en la misma vía															
Diseño del modelo del ATR															

Ejecución de las simulaciones																
Redacción de la memoria																

Tabla 1: cronograma cuatrimestre 1

Actividad	Meses (semanas), cuatrimestre 2																							
	Enero				Febrero				Marzo				Abril				Mayo				Junio			
Diseño del modelo del ATO																								
Diseño del modelo del ATP																								
Implementación de varios trenes en la misma vía																								
Diseño del modelo del ATR																								
Ejecución de las simulaciones																								
Redacción de la memoria																								

Tabla 2: cronograma del cuatrimestre 2

Para realizar la estimación económica del proyecto, se ha supuesto una dedicación por parte de los directores de un 12% respecto del tiempo que ha empleado el alumno. En el presupuesto también figura el consumo de electricidad, ya que se trata de un proyeco realizado en su totalidad con herramientas de software donde los tiempos de simulación son considerables. Se ha estimado un coste de la electricidad para el ordenador de 0.382 €/kWh, tomando como fuente los datos de Red Eléctrica de España en el mes de junio de 2022.

De la misma forma, se han supuesto unos costes indirectos del 20% de la totalidad de los costes de personal.

En la Tabla 3 se presenta la tabla de costes:

	Coste unitario	***Cantidad***	***Total (€)***
Alumno	15 €/h	250	3750
Directores	65 €/h	30	1950
Ordenador	0.382 €/kWh		12.8
Costes indirectos			1075
Total			*6787.8*
IVA (21%)			1425.44
Total con IVA			**8213.24**

Tabla 3: tabla de costes

Capítulo 5. DESARROLLO DEL PROYECTO:

MOVIMIENTO DE UN TREN

5.1 DEFINICIÓN DE OBJETOS Y DATOS DE LA LÍNEA DE METRO

El primer paso para desarrollar el código que permita ejecutar una correcta simulación de conducción automática es crear en Matlab los objetos involucrados en el proyecto; éstos son el tren y las líneas, distinguibles entre línea o vía 1 (recorrido desde la estación número 1 a la 18) y línea o vía 2 (sentido opuesto: recorrido desde la estación 18 a la 1). Para ello se definirá una clase para cada objeto y se creará una variable con el nombre del objeto a la que se le atribuirá cada clase. Además, cabe destacar que la longitud de la vía es de 14.6 km aproximadamente.

Las clases sirven, entre otras cosas, para que cada objeto tenga definidas intrínsecamente variables que pueden ser constantes, como la masa o longitud del tren, o variables, como la velocidad del tren en cada momento.

Consecuentemente, para este proyecto se han definido tres clases diferentes:

- Clase del tren (*ClaseTren* en el código).
- Clase de la línea 1 (*ClaseLinea1* en el código).
- Clase de la línea 2 (*ClaseLinea2* en el código).

Por un lado, en *ClaseTren* se han definido las propiedades constantes del tren, las cuales son longitud, masa y tres coeficientes (a, b y c) involucrados en la fuerza de resistencia al avance que sufre el tren, la cuál será explicada posteriormente. Además, es necesario definir las propiedades variables que serán objeto de estudio en la simulación: velocidad y posición del tren. Éstas se irán sobrescribiendo en cada iteración de tiempo de la simulación.

A continuación, se muestra en la Tabla 4 las propiedades constantes del tren:

Dato	Valor	Unidades
longitud	*89,16*	*m*
masa	*159,7*	*Tn*
a	*3003,957*	*N*
b	*31,94*	*N/(km/h)*
c	*0,67074*	*N/(km/h)^2*

Tabla 4: propiedades constantes del tren

Por otro lado, en *ClaseLinea* sólo existirán propiedades constantes que definirán las características de la vía. Éstas son:

- **Velocidad máxima en cada punto de la vía** (*vmax* en el código). Propiedad formada por un vector de dos columnas; la primera indica el punto kilométrico de la línea y la segunda la velocidad máxima que existe para ese punto kilométrico.
- **Pendiente de la vía en cada punto** (*pendiente* en el código). Propiedad formada por un vector de dos columnas; la primera indica el punto kilométrico de la línea y la segunda la pendiente en milímetros que existe para ese punto kilométrico.
- **Punto kilométrico de las estaciones** (*estación* en el código). Propiedad formada por un vector que recoge los puntos kilométricos en los que se encuentra cada estación. La vía consta de 18 estaciones en su totalidad.

Cabe destacar que, para conocer las propiedades constantes indicadas, se han recogido los datos del tren y de las vías en un Excel y se han exportado a Matlab mediante el comando *xlsread*.

Una vez creadas las distintas clases, habrá que definir el objeto al que pertenecen las propiedades de cada clase. Para ello, solo hará falta crear en el código una variable con el nombre del objeto y relacionarlo con su clase:

Tren = ClaseTren;

Linea = ClaseLinea;

Linea2 = ClaseLinea2;

De esta forma, se organizan de forma más sencilla las características de la vía y se podrán tener varios trenes circulando a la vez, ya que la posición y velocidad de cada tren estarán definidas dentro del objeto de cada tren.

En el Anexo I. Programa Principal y Clases se encuentra el código completo de simulación donde se encuentran los códigos de la clase del objeto *Tren* y de las clases de los objetos *Linea* y *Linea2*, aunque cabe destacar que existen propiedades que todavía no han sido explicadas ya que han surgido de necesidades en modelos y algoritmos posteriores a este paso los cuales se irán explicando a lo largo del capítulo.

5.2 MODELO DE ACELERADO, FRENADO Y PARADA EN ESTACIÓN

Una vez definidos los objetos, se procede a programar los modelos de los movimientos básicos del tren. Aunque en el proyecto se han realizado simulaciones de tráfico de varios trenes en la misma vía, al comienzo se ha diseñado el modelo para el correcto funcionamiento de un tren circulando en solitario y una vez ha funcionado con éxito, se han introducido varios trenes en la misma vía. Es por ese motivo por el cual a partir de este punto y hasta que se indique lo contrario, todo lo explicado y resultados obtenidos serán referentes a la circulación de un solo tren en la vía.

Para programar los movimientos básicos del tren será necesario diseñar el modelo del ATO (*Automatic Train Operation*), el cual es un modo de conducción que gobierna al tren de forma automática, es decir, sin la necesidad de que un conductor intervenga. Dicho modelo es muy útil ya que, entre otras cosas, al regular él mismo la velocidad según la distancia que queda para la parada, ejecuta un frenado uniforme y siempre se detiene en los puntos de parada exactos.

Por lo tanto, el primer objetivo deseado del proyecto ha sido que un tren acelere partiendo de velocidad y posición cero con la única restricción de que no se supere la velocidad

máxima de la vía en cada punto y que frene en caso de que la velocidad máxima de la vía se vea reducida o porque existe una parada de estación próxima.

De esta manera, se ha diseñado el modelo del ATO para así calcular la aceleración, velocidad y posición del tren con un paso de simulación de 1s. Esto implica que la simulación del movimiento del tren se ha realizado dentro de un bucle tipo *while* en incrementos de tiempo de 1s ($\Delta t = 1s$).

5.2.1 MODELO DE TRACCIÓN

El objetivo de este modelo es calcular la aceleración, velocidad y posición que lleva el tren durante la tracción. Para ello se ha creado una función (*movimientoacelerando* en el código) perteneciente a la clase del tren donde se calcula la aceleración, velocidad y posición y donde se sobrescriben las propiedades de velocidad (*v*) y posición (*posición*) del objeto *Tren*. Dicha función es llamada en cada iteración de tiempo en el bucle para calcular los nuevos valores. La aceleración se calcula de la siguiente forma:

$$a = \frac{F_{motor} - F_{resis} - F_{pendiente}}{masa} \ (^{m}/_{s^2})$$

Donde las fuerzas involucradas son:

- **Fuerza motor** (F_{motor}). Es la fuerza que ejerce el motor del tren, la cual depende de la velocidad máxima del tramo de la vía en la que se encuentra el tren y de la velocidad que lleve en ese momento. Esta fuerza se calcula como:

  $$F_{motor} = a_0 * F_{max}$$

  Donde a_0 es la consigna de control que da el ATO la cual indica la proporción del esfuerzo máximo del motor que hay que aplicar; F_{max} la fuerza a tracción máxima del tren.

  La variable a_0 es el resultado de un control de tipo proporcional (P) en la velocidad, ya que esta variable se obtiene de la siguiente forma:

$$a_0 = \max\left(\min(k * (v_{máxima} - v_{actual}), 0), 1\right)$$

Donde *k* es un parámetro propio del ATO; $v_{máxima}$ es la velocidad máxima del tramo por donde está pasando el tren, expresada en m/s; v_{actual} es la velocidad que lleva el tren, expresada en m/s. Para saber la velocidad máxima, se ha empleado el comando *find* para que, sabiendo la posición en la que está el tren, se utilice la propiedad *vmax* de las líneas y se pueda recoger la velocidad máxima en la posición del tren. Además, será necesario crear la propiedad *vmax* en la clase del tren para guardar el valor obtenido y así poder operar con la velocidad máxima en cada iteración de tiempo.

Una vez calculada a_0 es necesario saturarla. Esto es, es necesario que la variable quede entre valores de 0 y 1, con lo cual, si en el resultado del cálculo de a_0 se ha obtenido un número mayor que 1, se impondrá que el valor final de la variable sea 1. De lo contrario, el valor final de ésta será el obtenido en la operación inicial.

Por otro lado, la fuerza F_{max}, es la fuerza máxima que puede ejercer el motor, la cual está tabulada y depende de la velocidad que lleve el tren en cada instante. Para incorporarla en el código se ha creado otra propiedad en la clase del tren que toma el nombre de *fmax* (ya que irá variando en cada iteración de tiempo), la cual es un vector de dos columnas. La primera recoge las velocidades posibles del tren y la segunda la fuerza máxima de tracción que puede ejercer el motor para esa velocidad.
En el caso de que el tren circule a una velocidad intermedia entre dos velocidades que están tabuladas, será necesario interpolar entre ambas.

- **Fuerza de resistencia al avance** (F_{resis}). Es la fuerza de resistencia al avance que sufre el tren. Se ha empleado la fórmula de Davis con velocidad del viento exterior 0 km/h. Se calcula como:

$$F_{resis} = a + b * v + c * v^2$$

Donde *a*, *b* y *c* son los coeficientes que dependen de las características físicas del material rodante, expresados en N, N/(km/h) y N/(km/h)2 respectivamente (figuran en la Tabla 4); *v* es la velocidad que lleva el tren en ese momento expresada en km/h.

- **Fuerza provocada por la pendiente de la vía** ($F_{pendiente}$). Esta fuerza puede ayudar al movimiento si la pendiente es negativa u oponer resistencia al avance si la pendiente es positiva. Se calcula de la siguiente forma:

$$F_{pendiente} = \frac{masa * g * pendiente}{1000}$$

Donde *masa* es la masa del tren, expresado en kg; g es la fuerza de la gravedad, expresada en m/s^2; *pendiente* es la pendiente de la vía por la que está pasando el tren, expresada en mm.

Para calcular la pendiente, se ha creado una función (*BuscarPendiente* en el código) a la que se le llama en cada iteración de tiempo. Recibe el objeto *Tren* y *Línea* y devuelve la pendiente. La llamada se puede encontrar dentro de la función *movimientoacelerando* y la función figura en el Anexo I. Búsqueda de Pendiente.

Una vez calculada la aceleración, se obtendrá la velocidad y la posición mediante ecuaciones de la cinemática:

$$v = v_0 + a * \Delta t$$

$$pos = pos_0 + v * \Delta t + \frac{1}{2} * a * \Delta t^2$$

Donde v es la velocidad del tren, expresada en m/s; v_0 es la velocidad del tren en el paso previo de simulación, expresada en m/s; a es la aceleración del tren, expresada en m/s^2; *pos* es la posición del tren, expresada en m; pos_0 es la posición del tren en el paso previo de simulación; Δt es el paso de simulación, expresado en s.

Una vez calculadas las variables, es necesario reescribir las propiedades de velocidad y posición del objeto tren para que cuando la función *movimientoacelerando* lo devuelva, la velocidad y la posición tomen el valor correcto para cada instante de tiempo.

Se recuerda que los cálculos de todas las fuerzas y el de la aceleración, figuran en el Anexo I. Clase Tren dentro de la función *movimientoacelerando*.

Una vez diseñado el modelo de acelerado, se realiza una simulación para sintetizar los conceptos clave del mismo y comprobar que funciona correctamente. Para realizar esta primera simulación se han modificado los valores reales de las velocidades máximas para que sean crecientes hasta el primer kilómetro, que es hasta donde se va a realizar el ensayo, y de la pendiente de la vía. Los datos de la simulación figuran en la Tabla 5 y Tabla 6:

Vel. Máximas	
Pk (m)	v(Km/h)
0	0
10	5
50	10
100	15
250	45
2000	80

Tabla 5: velocidades máximas para la simulación

Pendiente	
Pk (m)	p(mm)
0	-39.99
638.13	40
975.35	0
1253.35	-26.4

Tabla 6: pendientes para la simulación

Consecuentemente, se obtienen los resultados de la Ilustración 1:

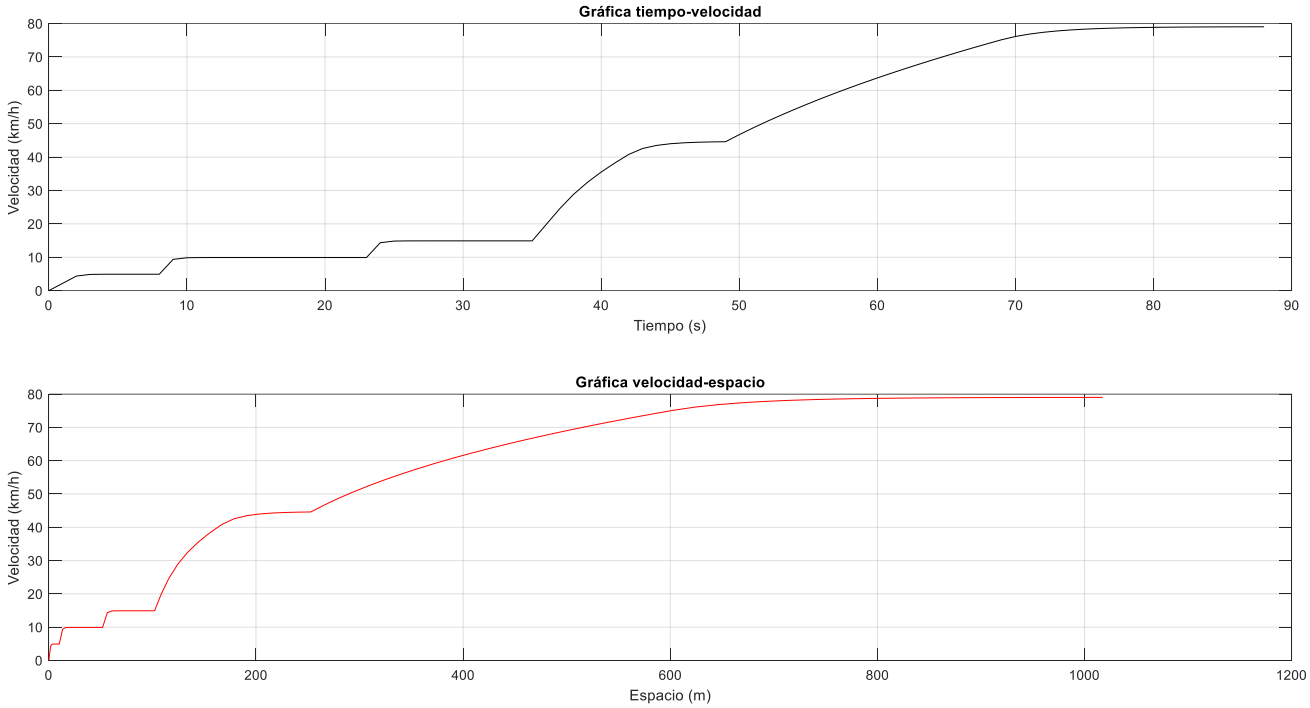

Ilustración 1: resultados de la simulación de acelerado

Se puede apreciar cómo el tren sigue la consigna de no superar las velocidades máximas impuestas en la vía. Cabe destacar que como el control es proporcional, existirá un pequeño error a la baja o a la alta, es decir, no se conseguirá la velocidad consigna exacta.

5.2.2 MODELO DE FRENADO

Una vez se ha conseguido que el modelo de tracción funcione correctamente, se ha programado el modelo de frenado. En este apartado se diseña el algoritmo que hace que el tren frene si la velocidad máxima se va a ver reducida. Por ejemplo, se supone que el tren está circulando en un tramo en el que la velocidad máxima son 50 km/h y que dentro de 200 m la velocidad máxima será de 35 km/h; habrá entonces que conseguir que cuando el tren llegue a dicho punto, la velocidad sea de 35km/h.

Lo anterior se traduce en calcular una nueva fuerza que tiene que ejercer el motor y el sistema de freno (*Fmotor_freno* en el código) que sustituirá a la fuerza motor explicada en el apartado anterior si se cumplen ciertas condiciones, las cuales se indicarán a continuación.

Por lo tanto, para desarrollar el modelo de frenado es necesario implementar las siguientes funciones las cuales serán llamadas dentro de la función del modelo de acelerado (*movimientoacelerando* en el código):

Inicialmente, ha sido necesario desarrollar una función que sea capaz de localizar la siguiente reducción de velocidad desde la posición del tren y devolver el punto kilométrico en el que está y cuál va a ser esta velocidad máxima. Por ejemplo, en la Ilustración 2 se muestra un esquema de un posible punto de estudio donde la posición del tren viene determinada por el triángulo y el perfil de velocidades máximas de la vía por la curva verde. En ese caso, la función debe devolver la posición y velocidad máxima del punto representado con el círculo verde.

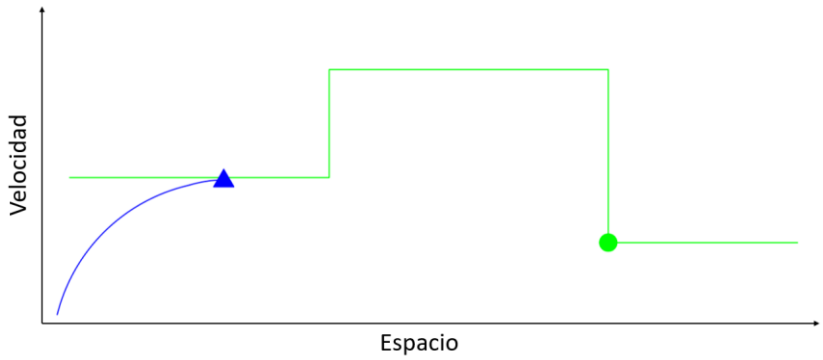

Ilustración 2: aclaración de la función de búsqueda de la siguiente reducción de velocidad máxima

Para conseguir el objetivo explicado, se ha diseñado la función que en el código recibe el siguiente nombre: *BuscarSiguienteReduccionVel*. Dicha función figura en el Anexo I. Búsqueda de la Siguiente Reducción de Velocidad. La dinámica de la función consiste en un bucle *while* que se repite hasta que se consigue encontrar el punto deseado: el algoritmo parte del punto en el que está el tren y, recorriendo en puntos kilométricos ascendentes la propiedad de velocidades máximas del objeto de la línea, compara las velocidades hasta que se obtiene que la velocidad máxima de un punto kilométrico mayor es menor que la del punto kilométrico actual. Una vez encontrado devuelve el punto kilométrico (primera columna de

la propiedad *vmax* del objeto de la línea) y la velocidad máxima (segunda columna de la propiedad *vmax* del objeto de la línea).

Tras haber desarrollado la primera función, es necesario calcular la curva de freno impuesta por la reducción máxima encontrada antes. Para ello se considerará que en la curva de freno existe una deceleración constante de 0.75 m/s^2 (*decel* en el código). La función diseñada para esta tarea figura en el Anexo I. Curva de Freno y recibe el nombre de *VelocidadNoSuperable* en el código. Esta función partirá desde la posición y velocidad de la reducción de velocidad máxima y aplicará desde esa posición hacia detrás las ecuaciones del movimiento rectilíneo uniformemente acelerado entre los tramos de pendiente constante para calcular la velocidad de la curva de freno en la posición actual del tren.

Siguiendo el ejemplo anterior, en la Ilustración 3 se representa un esquema donde se pone un ejemplo de la mecánica de la función, donde la curva negra representa la curva de frenado:

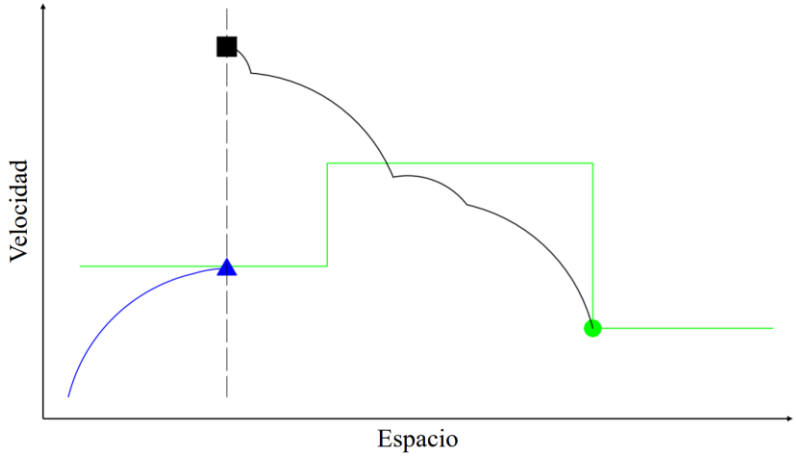

Ilustración 3: ejemplo de curva de freno

Cabe destacar que hay que tener en cuenta que para que el tren realice siempre un esfuerzo de freno similar independiente de la pendiente, hay que corregir la aceleración de frenado con la aceleración provocada por la pendiente. Por ello $a_{decel} = 0.75 + a_p$, donde $a_p$es la aceleración provocada por la pendiente. Por este motivo la curva de freno cambia de aceleración cuando se pasa de un tramo de la vía a otro que tiene distinta pendiente. Una vez realizados los cálculos, la función actual devuelve la velocidad a la que tendría que ir el tren

para llegar a la velocidad máxima de la siguiente reducción (*vel_no_superable_ms* en el código).

Por último, en la función del modelo de aceleración (*movimientoacelerando*) se compara la velocidad que no se puede superar por la curva de freno y la velocidad máxima. Si la velocidad máxima es menor que la velocidad de la curva de freno, entonces la fuerza del motor (*Fmotor*) se calculará según lo explicado en el apartado 5.2.1 Modelo de Acelerado ya que ésta última no impone ningún tipo de restricción en la consigna de velocidad. Sin embargo, si la velocidad máxima es mayor o igual que la velocidad de la curva de freno, entonces la fuerza del motor (*Fmotor*) se calculará mediante la función *FuncionFrenado*, la cual figura en el Anexo I. Función Frenado.

Esta función devuelve únicamente el valor que tiene que tomar la fuerza del motor. Si se cumplen las condiciones para que esta función se ejecute, la fuerza del motor sigue la siguiente expresión:

$$F_{motor} = \max\left(\min\left(a_{0freno} * masa, 0\right), 1\right)$$

Donde: $a_{0freno} = k_{frenado} * (vel\_no\_superable\_ms - v_{actual}) - decel + \frac{pendiente * g}{1000}$.

Se recuerda que *vel_no_superable_ms* es la velocidad de la curva de freno en el punto del tren. De la misma forma, $k_{frenado}$ es un parámetro propio del ATO y *decel* es la deceleración constante por valor de 0.75 m/s^2.

Una vez se ha diseñado el modelo de frenado, se muestra en la Ilustración 4 una pequeña simulación de 1 km de recorrido para comprobar que todo funciona correctamente. Esta vez, los datos de la vía de la simulación ya son los reales:

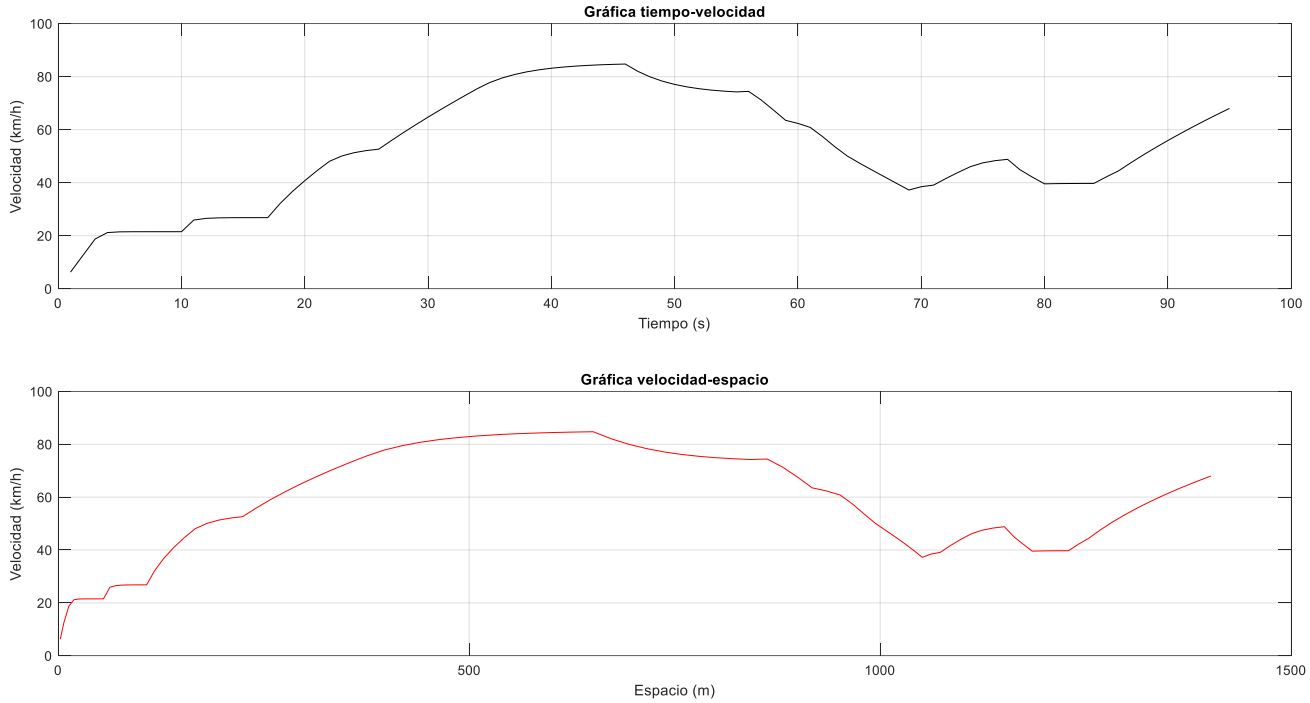

Ilustración 4: simulación de 1km de recorrido para un solo tren desde el comienzo de la vía

Según se ha explicado anteriormente, con este modelo de frenado, se asegura que el tren respeta las velocidades máximas cuando es necesario frenar.

5.2.3 MODELO DE PARADA EN ESTACIÓN

El objetivo del modelo de parada en estación es conseguir que el tren se detenga en el punto de las estaciones y que respete el tiempo de parada impuesto.

Para implementar dicho modelo, ha sido necesario necesario crear las siguientes propiedades del objeto *Tren*:

- Propiedad ***parada***. Representa la estación que tiene delante el tren. Por ejemplo, si el tren está circulando entre la estación número 2 y 3, la propiedad *parada* tomará el valor numérico 3. De esta forma se podrá saber el punto kilométrico en el que está la estación a la que se dirige el tren.

- Propiedad ***tiempo_parada***. Recoge el tiempo en segundos que el tren debe de estar estacionado en la estación. El tiempo de parada nominal en este proyecto será de 15 s.
- Propiedad ***aux_tiempo_parada***. Propiedad auxiliar que se utilizará para compararla con el tiempo de parada. Una vez que estas dos sean iguales, se le dará orden al tren de salir de la estación.
- Propiedad ***parado***. Propiedad binaria o *flag*. Indica si el tren está parado en estación o no (1 parado; 0 circulando).

Mientras el tren se encuentra circulando, será necesario desarrollar la curva de freno debida a la parada en la estación, que no tiene por qué ser igual a la curva de freno debida a una reducción de la velocidad máxima. Para eso se ha creado la función *VelocidadNoSuperableEstación*, la cual figura en el Anexo I. Velocidad No Superable por Estación. Esta función sigue el mismo principio que la explicada en el apartado anterior *VelocidadNoSuperable*, con la salvedad de que, en vez de partir de la posición y la velocidad de la siguiente reducción, ahora parte de la posición de la siguiente estación y de velocidad cero.

La función devuelve la velocidad de la curva de freno debida a la estación en el punto en el que está el tren (*vel_no_superable_estacion_ms* en el código). En este caso, para calcular la velocidad de la curva de freno, se parte desde la posición de la estación que el tren tiene delante y de velocidad nula, aplicando desde esa posición hacia detrás las ecuaciones del movimiento rectilíneo uniformemente acelerado entre los tramos de pendiente constante para calcular la velocidad de la curva de freno en la posición actual del tren.

Ahora se tienen entonces dos velocidades por curva de frenado, la debida a la siguiente estación y la debida a la próxima reducción de velocidad, por lo que habrá que comparar ambas y la mínima compararla a su vez con la velocidad máxima de la vía en el tramo para realizar el proceso de frenado explicado en el apartado anterior.

Cuando el tren ha llegado a la estación, se activa la propiedad binaria *parado* (*parado* = 1) y, mediante una condición tipo *if* y partiendo la propiedad *aux_tiempo_parada* del valor 1,

se impone que la velocidad y aceleración del tren sean nulas y que la posición del tren sea igual al valor del punto de parada. En cada iteración de tiempo la variable auxiliar irá aumentando de unidad en unidad hasta que iguale el valor de *tiempo_parada*. Será entonces cuando la propiedad binaria *parado* tome el valor 0 y se vuelvan a ejecutar los cálculos de aceleración para dar orden de salida del tren. Además, en este punto habrá que sumar una unidad a la propiedad *parada* para que ahora el tren tenga de referencia la parada siguiente de la que ha salido.

En la Ilustración 5 se muestra una pequeña simulación ensayo de 4.3 km con 15 s de parada donde se puede ver que el modelo funciona correctamente.

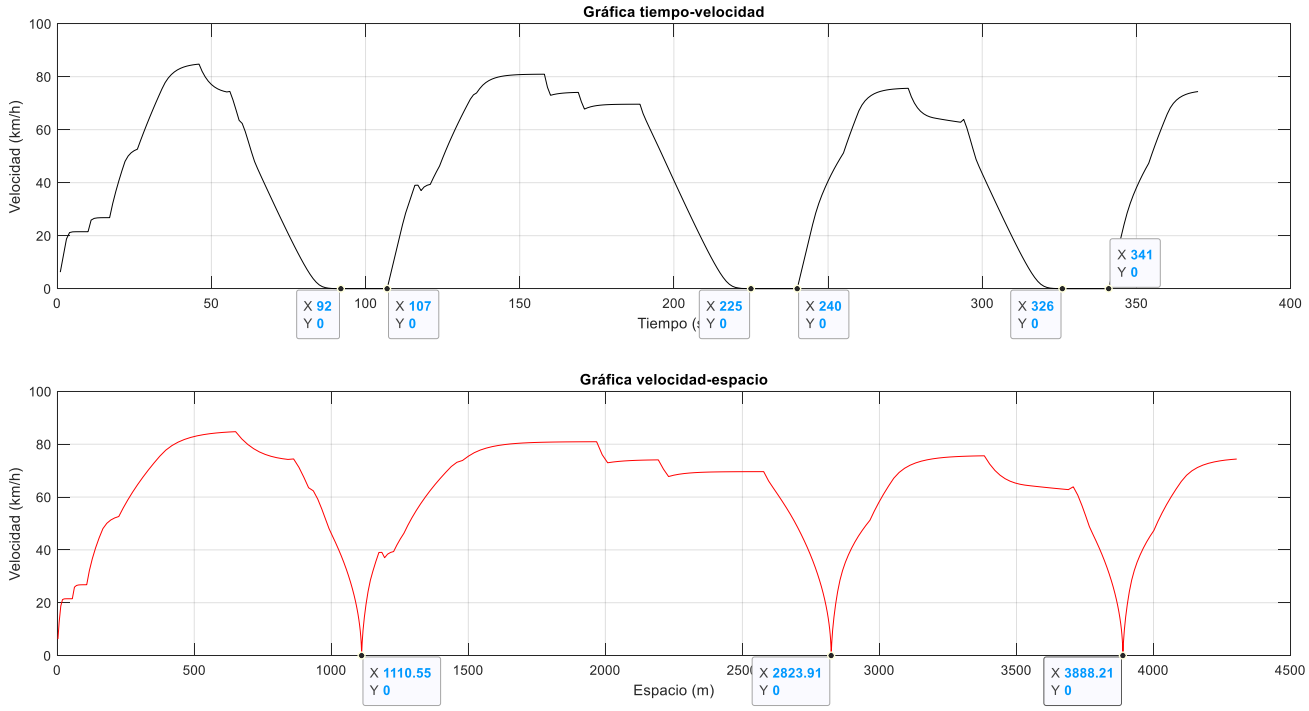

Ilustración 5: simulación de 4.3 km del modelo de parada en estación

En los primeros 4.3 km de recorrido en la vía el tren ha realizado tres paradas en estación respetando los tiempos de parada y los puntos kilométricos en los que se encuentran las estaciones.

5.3 MODELO DE OPERACIÓN EN CARROUSEL

Ahora que el tren circula por la vía en solitario sin problemas, es necesario plantearse la situación de que éste llegue al final de la vía 1 y pueda invertir su marcha para recorrer la vía 2. A esta forma de operar se le conoce como *Carrousel.*

Para ello ha sido necesario crear las siguientes propiedades del objeto *Tren*:

- Propiedad ***línea_simulador***. Tomará el valor del objeto *Linea* si el tren circula por la vía 1 o *Linea2* si el tren circula por la vía 2. Al principio habrá que inicializar la propiedad a la línea donde el tren comience su marcha.
- Propiedad ***flag_ultima_estacion***. Propiedad binaria que se activa si el tren ha entrado en la última estación.

Para invertir la marcha es necesario que las características de la vía cambien. Por ejemplo, si el tren se encontraba circulando por la vía 1 y va a invertir su marcha para recorrer la vía 2, es necesario que se utilice el objeto *Linea2* en vez del objeto *Linea.* Para realizar este cambio se utiliza la propiedad *línea_simulador* que irá tomando el valor del objeto *Linea* o *Linea2* y será en la última parada de la línea donde se realice.

Para distinguir entre la última estación y el resto de estaciones, se ha utilizado la propiedad *flag_ultima_estacion* la cual se activa (valor 1) si el tren se encuentra realizando la parada en la última estación. Se sabe que el tren está en la última estación por el punto kilométrico en el que se encuentra y porque la propiedad *parada* (número de estación a la que se dirige el tren) toma el último valor.

Cuando el tren se encuentre en la situación descrita anteriormente, la propiedad *línea_simulador* tomará el valor del objeto *Linea* si el tren se encontraba circulando en la vía 2 o el valor *Linea2* si el tren se encontraba circulando en la vía 1. De la misma forma, habrá que actualizar el valor de la propiedad *parada* a 2 (parte de la primera estación de la nueva vía hacia la estación 2).

Con los modelos implementados hasta este punto del proyecto, un tren en solitario es capaz de circular por sus vías en *carrousel* a marcha tendida, esto es, con la única limitación de velocidad de no superar las velocidades máximas de las líneas.

En la Ilustración 6 se muestra el perfil de las velocidades del tren a lo largo de la vía 1 y 2 para su marcha tendida:

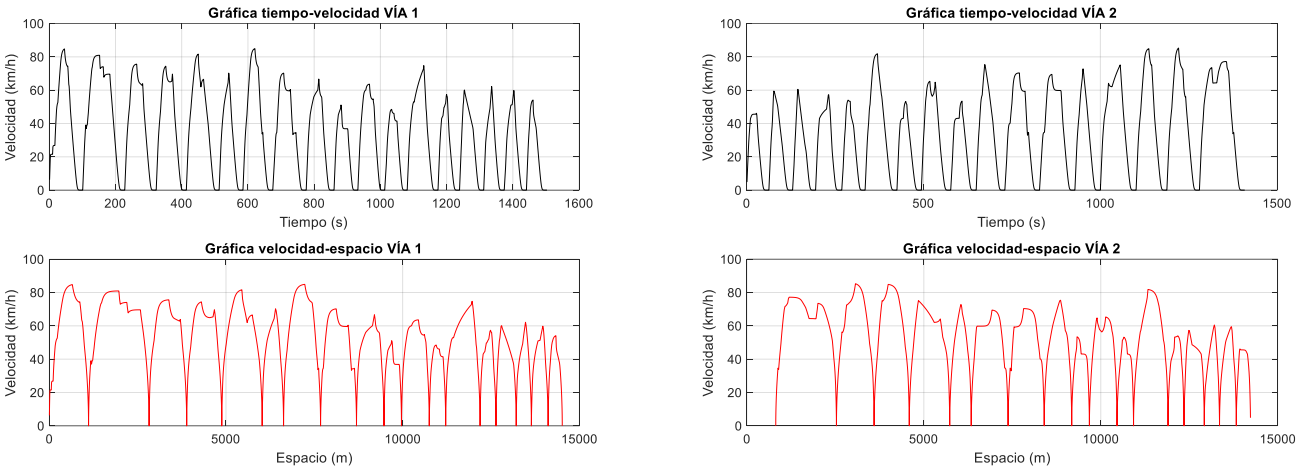

Ilustración 6: perfil de velocidades del tren a marcha tendida por las vías 1 y 2

Se puede comprobar cómo el tren circula por ambas vías realizando todas sus paradas, 18 por vía, respetando las velocidades máximas.

Además, se quiere comprobar que el tren invierte su marcha al llegar al final de la vía de forma satisfactoria. En la Ilustración 7 se muestra una simulación de dos recorridos a cada vía en marcha tendida para un solo tren, representando la posición frente al tiempo:

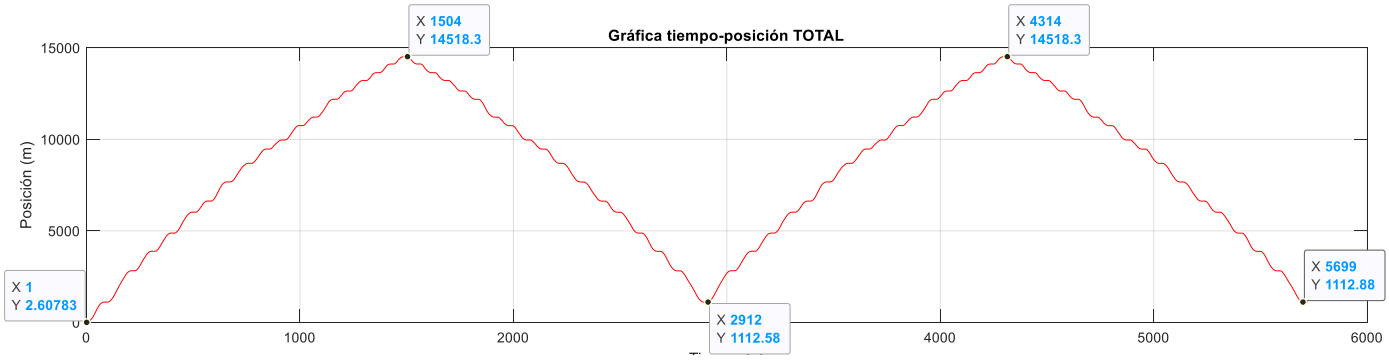

Ilustración 7: simulación inversión de marcha

Al comenzar el tren su recorrido por la vía 1, se puede comprobar como cuando llega al final de ésta invierte su marcha y recorre la vía 2. Se repite el mismo proceso cuando finaliza su trayectoria por la línea 2.

5.4 MODELO DE MARCHAS DE REGULACIÓN

Cuando se quiere que el tren no circule a marcha tendida, se utilizan las marchas de regulación. Estas son modos de circulación del tren a lo largo de las vías que hacen que la consigna de velocidad no sea la velocidad máxima del tramo de la línea, sino una menor. Son útiles en caso de retrasos ya que, si el tren no circula a su velocidad máxima normalmente, tendrá margen de acelerar cuando tenga que recuperar tiempos de retraso. Además, las marchas de regulación permiten ahorrar energía [3] ya que se circula a una velocidad menor y permiten implementar estrategias de conducción eficiente como las derivas. Las marchas de regulación implementadas en el proyecto han sido las siguientes:

- **Regulación por velocidad**. En este caso se impone una velocidad límite que no se ha de superar pero que sea inferior a la velocidad máxima del tramo de la línea. De esta manera el tren seguirá la nueva referencia de velocidad y no alcanzará las velocidades máximas.

  A modo de ejemplo, en la Ilustración 8 se muestra un ejemplo de regulación por velocidad limitada a 40 km/h:

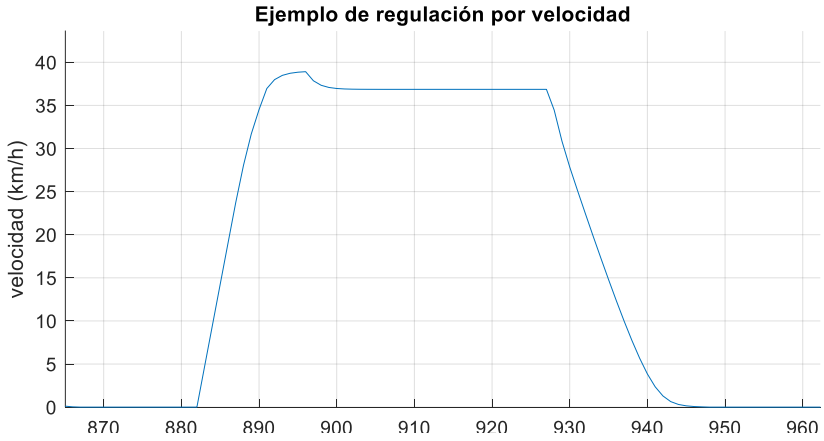

Ilustración 8: ejemplo de regulación por velocidad

- **Regulación deriva – remotor**. Este tipo de regulación consta de dos velocidades consigna: una mayor y otra menor. El tren irá acelerando hasta que alcance la velocidad consigna mayor. Una vez que se llegue a dicha velocidad la fuerza del motor será cero y el tren irá derivando (en "punto muerto" si se hace la analogía con un coche) hasta que alcance la otra velocidad consigna para volver a aplicar fuerza en el motor. Cabe destacar que una vez que el ferrocarril está en punto de deriva, se irá ganando velocidad si la pendiente de la vía es negativa o irá perdiendo si la pendiente es positiva.

  Por ejemplo, el tren se encuentra circulando por un tramo en el que la velocidad máxima son 80 km/h. La velocidad de deriva son 60 km/h y la de remotor 40 km/h. En el momento que el tren alcance los 60 km/h el motor no ejercerá fuerza alguna hasta que la velocidad vuelva a ser 40 km/h, repitiendo este proceso a lo largo del trayecto.

  A modo de aclaración, en la Ilustración 9 se muestra un pequeño ejemplo del tramo entre dos estaciones en el que se está aplicando la regulación por deriva – remotor, siendo la velocidad de deriva 50 km/h y la de remotor 20 km/h:

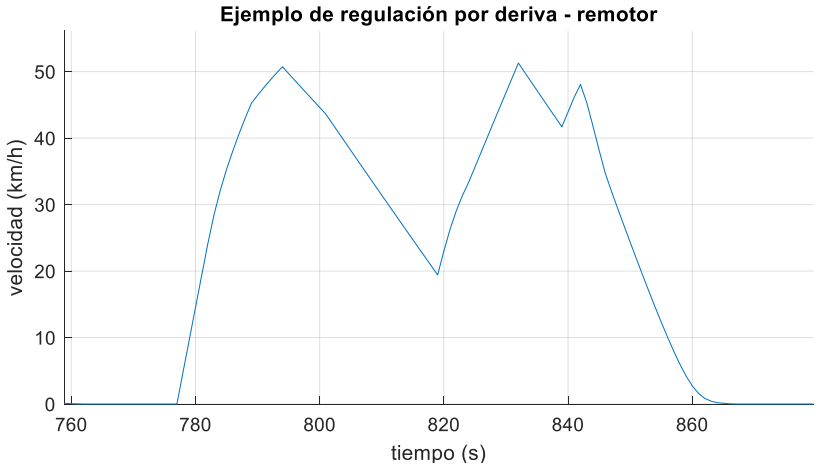

Ilustración 9: ejemplo de regulación por deriva – remotor

El uso de regulación por velocidad o deriva – remotor viene impuesto por las marchas del tren que indican en qué tramos se utiliza cada una de ellas. Dichas marchas son la marcha 0 (M0), la marcha 1 (M1), la marcha 2 (M2) y la marcha 3 (M3), siendo M0 la más rápida y

M3 la más lenta, aunque no entrarán en juego hasta el modelo de regulación del tráfico donde se explicará el razonamiento para la elección de marchas. Por el momento, sólo se utilizará la marcha 1 (M1), la cual es la marcha nominal, es decir, aquella que cumple con el horario.

Los datos de velocidades de las marchas para la regulación de velocidad y para la de deriva – remotor, así como los puntos kilométricos son datos y han sido recogidos de un Excel con el comando *xlsread*.

Por lo tanto, el modelo de marchas de regulación, permitirá que el tren pueda utilizar la regulación por velocidad y la regulación por deriva – remotor.

5.4.1 MODELO DE REGULACIÓN POR VELOCIDAD

Para diseñar este modelo, es necesario crear una propiedad constante por cada marcha (M0, M1, M2, M3) en cada línea (*Linea*, *Linea2*) que sea un vector de dos columnas en el que la primera sea los puntos kilométricos y la segunda la regulación por velocidad. En el código, la propiedad ha recibido el nombre $marcha_i$*_reg_velocidad* siendo *i* 0, 1, 2, 3 según la marcha que se esté guardando.

De la misma manera, se ha creado una propiedad constante en el objeto tren que almacene los datos de la regulación por velocidad que se estén utilizando en cada iteración de tiempo (*reg_vel* en el código). Dicha variable se iguala en cada incremento de tiempo a la propiedad $marcha_i$*_reg_velocidad* de la línea en la que esté el tren.

Para saber si se está utilizando en cada momento la regulación por velocidad, se ha creado una variable (*factor_reg* en el código) que tomará el valor 100 si no se está usando y 1 si sí se está usando. El Excel de datos indica en qué puntos de la vía se debe usar la regulación por velocidad y cuándo no.

Una vez se han cargado los datos y la variable *factor*_reg ha tomado un valor, se compara con las velocidades máximas de la vía en el tramo y se impondrá como velocidad límite la más pequeña de las dos. Es por ese motivo por el cual la variable *factor_reg* toma el valor

100 si no se está usando esta regulación, porque se multiplica las velocidades por 100 haciendo que éstas siempre sean mayores que las velocidades máximas de la vía.

A modo de resumen y como comprobación de que el modelo funciona correctamente, en la Ilustración 10 se muestra una simulación de 1.4 km de vía 1 que es fiel a los valores arbitrarios de regulación por velocidad de la Tabla 7: datos para simulación de regulación por velocidad

Hay que destacar que los valores de regulación por velocidad de dicha tabla no son los que se utilizan en el proyecto, sino que son otros empleados para la comprobación del modelo. Para que se pueda comparar el uso o no de la regulación por velocidad, en la Ilustración 11 se muestra la trayectoria de 1.4 km a marcha tendida:

Punto kilométrico (m)	*Velocidad de regulación (km/h)*
0	50
50	50
104	50
220	50
500	35
1041,75	35
1071,75	35
1184,75	35
1223,25	50
1460,25	50

Tabla 7: datos para simulación de regulación por velocidad

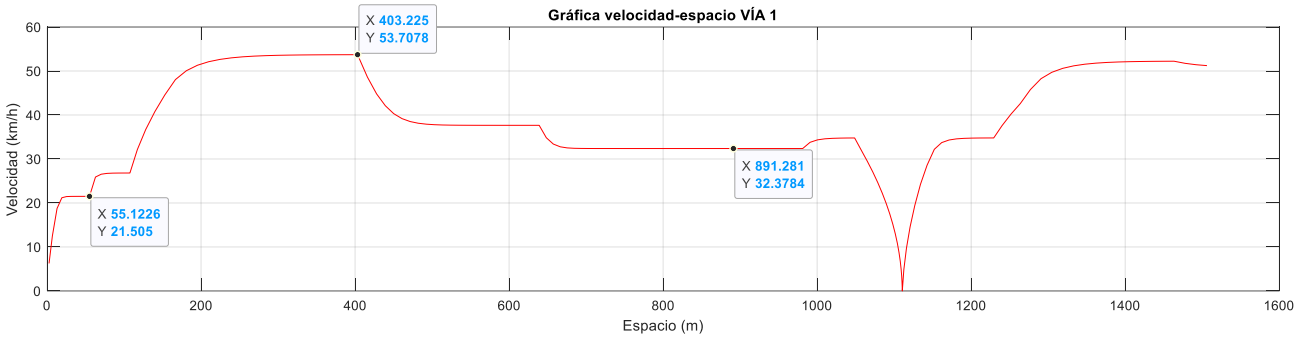

Ilustración 10: simulación ejemplo de regulación por velocidad

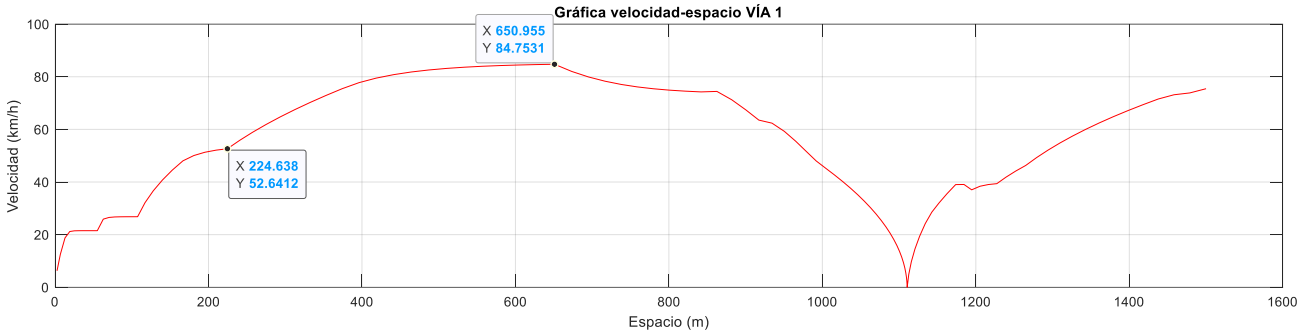

Ilustración 11: simulación a marcha tendida

Se puede apreciar la diferencia en los perfiles de la velocidad frente a la posición en que las velocidades son más bajas y que cuando la velocidad debida a la regulación es menor que la velocidad máxima, el tren impone como límite la velocidad de regulación. En el caso de que la velocidad máxima sea menor, esta será la velocidad consigna.

El procedimiento descrito figura dentro de la función *movimientoacelerando* de la *ClaseTren* en el Anexo I. Clase Tren.

5.4.2 MODELO DE REGULACIÓN DERIVA – REMOTOR

De la misma manera, ha sido necesario crear las siguientes propiedades en los objetos de las líneas:

- Propiedad ***marcha$_i$_deriva***, siendo *i* 0, 1, 2, 3. Recoge los datos de velocidad de deriva en un vector de dos columnas; la primera indica los puntos kilométricos y la segunda las velocidades de deriva.
- Propiedad ***marcha$_i$_remotor***, siendo *i* 0, 1, 2, 3. Recoge los datos de velocidad de remotor en un vector de dos columnas; la primera indica los puntos kilométricos y la segunda las velocidades de remotor.

De la misma forma, se tendrá que definir en el objeto *Tren* las siguientes propiedades:

- Propiedad ***deriva***. Almacena los datos de deriva que se estén utilizando en cada iteración de tiempo. Dicha variable se igualará en cada incremento de tiempo a la propiedad *marcha$_i$_deriva* de la línea en la que esté el tren.
- Propiedad ***remotor***. Almacena los datos de remotor que se estén utilizando en cada iteración de tiempo. Dicha variable se igualará en cada incremento de tiempo a la propiedad *marcha$_i$_deriva* de la línea en la que esté el tren.
- Propiedad ***derivando***. Propiedad binaria que indica si el tren está en deriva (1) o no (0).
- Propiedad ***acelerando***. Propiedad binaria que indica si e tren está acelerando (1) o no (0).

Adicionalmente, se creará una variable (*factor_deriva_remotor* en el código) que indique si en ese tramo de la vía se está usando la regulación por deriva – remotor o no. En el Excel de datos se recoge cuándo se debe de usar deriva – remotor y cuándo no.

Para diseñar el modelo, lo primero que se hace en el comienzo del bucle es comprobar el valor de la velocidad de deriva y de remotor. Si la velocidad que lleva el tren es igual o superior a la velocidad de deriva, entonces el tren pasará a derivar activándose la propiedad flag *derivando.* Si ésta está activada, entonces en los cálculos de la fuerza que tiene que ejercer el motor (*Fmotor*) se impone que la fuerza a tracción sea nula. Para conseguirlo habrá que imponer que la variable a_0 de la ecuación de la fuerza del motor, $F_{motor} = a_0 * F_{max}$, sea la mínima entre el valor 0 y el valor de a_0 saturada.

Si al estar en deriva la velocidad del tren ha disminuido lo suficiente como para haber llegado a la velocidad de remotor, entonces se desactivará la propiedad flag *derivando* haciendo que la fuerza ejercida por el motor se calcule de la forma explicada en el apartado de Modelo de . El algoritmo mencionado figura dentro de la función *movimientoacelerando* de la *ClaseTren* en el Anexo I. Clase Tren.

Cuando este tipo de regulación está siendo utilizada, los perfiles de velocidad adquieren formas más puntiagudas e incluso la forma de lo que se llama "dientes de sierra". En la Ilustración 12 se puede apreciar dicha característica:

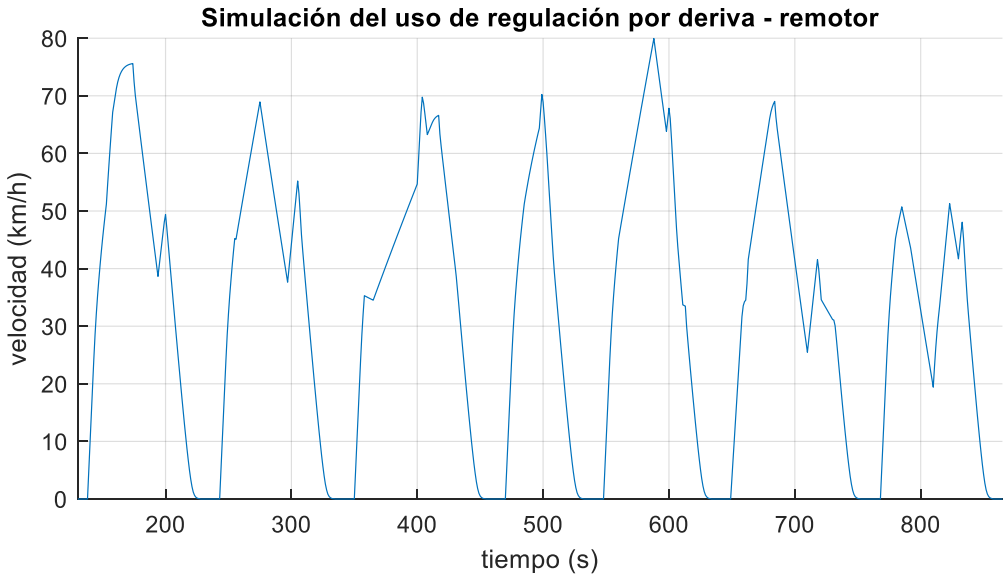

Ilustración 12: ejemplo de la característica "dientes de sierra"

La Ilustración 12 se ha obtenido de una simulación para un tramo en concreto de la vía 1 donde se aplica la regulación por deriva – remotor y donde se pueden apreciar los dientes de sierra.

Una vez diseñados el modelo de deriva – remotor y el de regulación por velocidad, sólo hará falta indicar la marcha (M0, M1, M2 y M3) que se ha de emplear, lo cual se explicará en el apartado de regulación de tráfico.

Capítulo 6. DESARROLLO DEL PROYECTO: SIMULADOR MULTI – TREN

6.1 IMPLEMENTACIÓN DE VARIOS TRENES EN LA MISMA VÍA

Una vez que se ha diseñado el modelo que consigue que un tren circule por las vías, es necesario implementar el modelo de la simulación multi – tren, es decir, hay que conseguir que varios trenes circulen por la misma vía, al igual que ocurre en una línea de metro real.

Para ello se ha creado un vector en el que cada posición es un objeto Tren (*Vector_Trenes* en el código) con longitud igual a la cantidad de trenes que se desea que circulen por la línea.

Para diseñar este modelo se ha creado una función (*SimulaTrafico* en el código) que simule el recorrido de los trenes por las vías. Dicha función es llamada desde el fichero *main* recibiendo los objetos *Tren, Linea* y *Linea2* y los parámetros del ATO y devuelve el vector de trenes *Vector_Trenes,* donde se habrán almacenado las velocidades y posiciones de todos y cada uno de los trenes al finalizar la simulación. Esta función figura en el Anexo I. Función de Simulación de Tráfico.

Desde el fichero *main* se procede a la llamada de la función *SimulaTrafico*. Dentro de dicha función se encuentra el bucle *while* de la simulación, es decir, el bucle que indica hasta cuándo se va a realizar el proceso, ya sea un tiempo máximo o que un tren ha llegado a un punto kilométrico concreto. Dentro del bucle existe otro a su vez de tipo *for*, a través del cual se recorrerá el vector de objetos *Tren* y donde se llamará para cada tren del vector a la función *movimientoacelerando* para calcular en cada iteración de tiempo su velocidad y posición. Se recuerda que en la función *movimientoacelerando* es donde se ha diseñado el modelo del ATO explicado en el Capítulo 5.

Adicionalmente, se ha creado una función para inicializar todos los parámetros necesarios al comienzo de la simulación (*InicializaVariosTrenes* en el código) y la cual figura en el Anexo I. Clase Tren.

En dicha función, se ha implementado un bucle tipo *for* que recorre cada celda del vector *Vector_Trenes* para así poder inicializar los parámetros de cada tren que se dispone a circular. Se tendrán que inicializar las siguientes propiedades del tren:

- **Vía en la que comienza el tren** (*línea_simulador*). Se decide entre la vía 1 o vía 2.
- **Datos de marchas de regulación iniciales del tren** (*reg_vel, deriva, remotor*). Se decide entre M0, M1, M2 y M3.
- **Índice del tren** (*índice*). Se ha definido una propiedad que indica el número del tren circulante, es decir, la posición del vector *Vector_Trenes* que ocupa cada tren para facilitar cálculos y representaciones.
- **Estado del tren** (*parado*). Es necesario que inicialmente el tren esté en movimiento (*parado* = 0).
- **Velocidad y posición** (*v; posición).*
- **Parada a la que se dirige el tren** (*parada*). Es necesario indicar el valor de esta propiedad ya que, si se quiere que el tren no comience desde la primera parada, habrá que indicar la que tiene delante.
- **Propiedad *aux_tiempo_parada***. Hay que inicializarla a 0 para que la comparación con el tiempo de parada nominal sea correcta y el tren salga de la estación cuando corresponde.

6.2 MODELO DEL ATP: FRENADO DE UN TREN POR PROXIMIDAD A OTRO

Uno de los aspectos a tener en cuenta cuando varios trenes están circulando en la misma vía es la distancia de seguridad que tienen que guardar entre ellos, es decir, es necesario que, si un tren se acerca demasiado al que lleva delante, reduzca su velocidad o que incluso se frene.

Para ello, es necesario diseñar un modelo de frenado por proximidad entre trenes, función competencia del ATP (*Automatic Train Protection*). Este modelo seguirá los principios del modelo de frenado explicado en el apartado 5.2.2 Modelo de Frenado. En el modelo de frenado se calculaba una curva de freno desde el punto de siguiente reducción de velocidad o desde el punto de una estación hasta el punto en el que se encontraba el tren, indicando así la velocidad máxima que puede llevar el tren en cada momento. Por lo tanto, se creará una función (*VelocidadNoSuperableArbitraria* en el código) que cree una curva de freno para un tren desde la posición del tren de delante hasta donde se encuentre, imponiendo que la velocidad en el punto del tren de delante sea 0. Es decir, es similar a la función *VelocidadNoSuperableEstacion* la cual creaba la curva de freno para las estaciones, con la salvedad de que en este caso el punto de parada y velocidad 0 es la posición de delante del tren.

Hay que tener en cuenta que, en verdad, no es el punto de parada del tren de delante desde donde se empieza la curva de freno, sino que es el punto del tren de delante menos una distancia de seguridad. La función *VelocidadNoSuperableArbitraria* figura en el Anexo I. Curva de Freno Debido a Trenes Previos.

Por lo tanto, ha sido necesario definir e inicializar en la función *InicializaVariosTrenes* las siguientes propiedades del objeto *Tren*:

- **Propiedad *stren***. Propiedad variable que indica la posición en la que se encuentra el tren de delante. Es necesario que esta propiedad se actualice en cada iteración de tiempo ya que la posición del tren de delante no se mantiene constante.
- **Propiedad *pk_parada***. Propiedad variable que indica el punto de comienzo de la curva de freno. Dicha propiedad se calcula como:

$$pk\_parada = stren - longitud_{tren} - d_{seguridad}$$

6.3 CASO DE ESTUDIO: PARADA DE UN TREN POR PROXIMIDAD

Una vez diseñado el simulador multi – tren, se realizará un caso de estudio para comprobar que funciona correctamente. El caso es el siguiente:

Se propone la circulación de únicamente dos trenes con las siguientes condiciones iniciales y características:

- **Tren 1**. Parte desde el comienzo de la vía 1 con velocidad nula empleando marcha tendida (M0). Este tren realizará un tiempo de parada en cada estación de 18 s.
- **Tren 2**. Parte desde la primera estación de la vía 1 con velocidad nula empleando marcha tendida (M0). Este tren realizará un tiempo de parada en estación de 50 s.
- **La salida de ambos trenes se inicia en el mismo instante de tiempo** (t_0 = 0 s).

Al partir el tren 2 por delante del tren 1 y al tener un tiempo de parada en estación mayor, el tren 1 irá recortando distancias con el tren 2 hasta el punto en el que tenga que frenar y/o detenerse debido a que se encuentra demasiado cerca del tren 2. Se ha fijado una distancia de seguridad de 50 m entre la cola del tren de delante y la cabeza del de atrás. En la Ilustración 13 se muestran los resultados del caso de estudio:

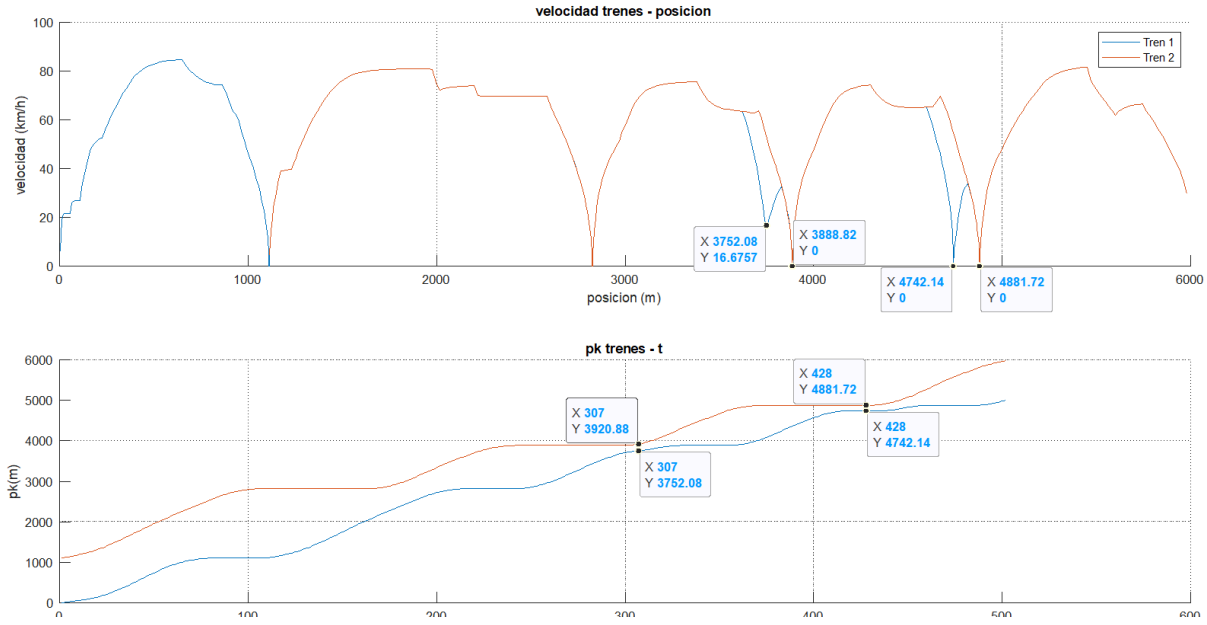

Ilustración 13: caso de estudio de frenado por proximidad entre dos trenes

Según se puede apreciar en la Ilustración 13, el tren 1 circula de manera normal entre la primera y segunda estación. Sin embargo, cuando el tren 2 se encuentra realizando su parada en la estación 3, el tren 1 se va aproximando provocando que tenga que ir frenando debido a que la proximidad entre trenes se está haciendo demasiado grande. Es en el instante de tiempo t = 307 s cuando el tren 2 reanuda su marcha y cuando el tren 1 puede volver a acelerar ya que el tren 2 se está alejando.

De la misma manera, cuando el tren 2 se encuentra realizando su parada en la estación 4, el tren 1 se aproxima tanto a él que es necesario que detenga su trayectoria en la distancia de seguridad impuesta. Es en el instante de tiempo t = 428 s cuando el tren 2 reanuda su trayectoria y, consecuentemente, el tren 1 la suya.

La distancia mínima entre trenes se da para ese mismo instante de tiempo, instante en el que los dos se encuentran detenidos y se comprueba de la siguiente manera:

$$d_{min,trenes} = longitud_{tren} + d_{seguridad} = 89.16 + 50 = 139.16\, m$$

En el instante anteriormente citado (t = 428 s):

$$d_{tren2} - d_{tren1} = 4881.72\ - 4742.14 = 139.58\, m$$

De esta forma y con este caso de estudio, se sabe que el modelo funciona correctamente y será extrapolable a cualquier tren en cualquier momento y situación en su circulación.

Capítulo 7. DESARROLLO DEL PROYECTO: MODELO DE REGULACIÓN DE TRÁFICO

Para completar el proyecto, una vez que el simulador multi – tren ha sido desarrollado, es necesario implementar el modelo de regulación automática del tráfico.

En una línea de metro convencional, los trenes están sujetos a horarios los cuales deben de ser cumplidos. Estos horarios marcan el momento en el que cada tren tiene que entrar y salir de cada estación, por lo tanto, es necesario crear un algoritmo de regulación que haga que los trenes circulantes respeten los tiempos.

En este capítulo se explicará cada concepto que se ha tenido en cuenta para diseñar el regulador [4]. Aunque serán explicados en detalle, a modo de introducción, es necesario tener claro cada paso que se va a seguir:

1. En el proyecto se ha impuesto que se quiere un intervalo entre trenes de 210 s (3.5 min), es decir, es deseado que en cada estación pare un tren cada 3.5 min. El primer paso entonces será averiguar la cantidad de trenes que se necesitan para satisfacer la condición. Además, el tiempo de parada nominal en estación será de 15 s.
2. Se diseña la salida de los trenes. Todos los trenes saldrán desde la estación 1 a velocidad nula cada 210 s de manera escalonada. Es decir, se diseñará el algoritmo que da orden de que cuando pasen los 3.5 min salga el siguiente tren.
3. Se diseña el horario que tiene que seguir cada tren en su travesía. Este horario será un vector de los tiempos que marca un solo tren en recorrer las vías a su velocidad nominal en marcha 1 (M1) y en realizar sus tiempos de parada nominales, es decir, se hará circular un solo tren y se recogerán los tiempos que marca en el momento de salida de estación. El horario de cada tren será:

$$H_i = H_1 + (i-1) * 210$$

Donde: H_i es el horario de cada tren; H_1 es el horario del tren 1; i es el número de la posición del tren circulante.

Por ejemplo, el tren número 4 (cuarto en salir) tendrá el horario $H_4 = H_1 + 3 * 210$.

4. Cuando se producen retrasos o adelantos en los trenes es cuando se debe utilizar las marchas M0, M2 y M3 para corregirlos. Se utilizará la marcha rápida (M0) si el tren está retrasado y las marchas lentas (M2, M3) si el tren va adelantado.

   Para saber el tiempo que se gana o se pierde con cada marcha (M0, M2, M3) será necesario recoger los segundos que se gana o se pierde entre estaciones al usar cada marcha. Eso se consigue simulando la trayectoria de un solo tren circulando a M0, M2 y M3 por las dos vías, recogiendo sus tiempos y comparándolos con los obtenidos al usar la marcha 1 (M1).
5. Se calcula el retraso que lleva el tren respecto de su horario y se aplica consecuentemente una consigna de marcha para corregirlo.

7.1 CÁLCULO DEL NÚMERO DE TRENES EMPLEADOS

El cálculo de la cantidad de trenes que deben de circular por la vía se obtiene de la siguiente forma:

$$N = \frac{T}{I}$$

Donde, N es el número de trenes; T es el tiempo que tarda en recorrer la vía 1 y 2 un tren en marcha 1 (M1), expresado en s; I es el tiempo de intervalo entre trenes, expresado en segundos.

Por lo tanto, el primer paso para obtener el número de trenes necesarios es calcular el tiempo total T. Para ello se ha realizado una simulación en la que un tren recorre las vías con la marcha 1 (M1). Dicha simulación se detendrá cuando el tren haya alcanzado de nuevo la estación 1 de la vía 1. En la Ilustración 14 se muestra el resultado obtenido:

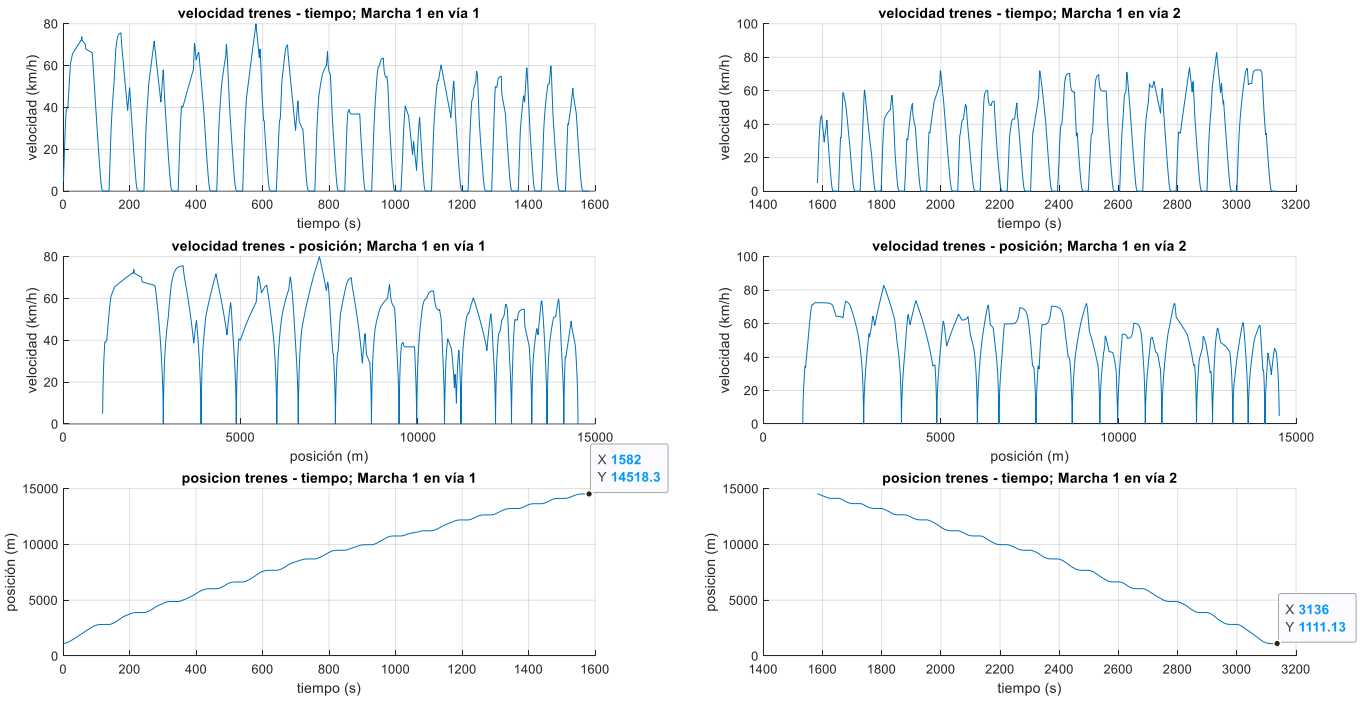

Ilustración 14: simulación con M1 para calcular el tiempo 'T'

Según se ha obtenido en la simulación, el tiempo que necesita un tren en recorrer la vía 1 y 2 a marcha 1 (M1) y con un tiempo de parada de 15 s es de 3136 s.

Consecuente,

$$N = \frac{T}{I} = \frac{3136}{210} = 14.93\ trenes$$

Como es lógico, es necesario que el número de trenes sea un valor entero, por lo que se aproximará a 15. Hay que tener en cuenta que, si se quiere que el intervalo entre trenes sea de 210 s y que se empleen 15 trenes, el tiempo total T cambiará de la siguiente manera:

$$T' = N * I = 15 * 210 = 3150\ s$$

En las líneas de metro es común que el tiempo de parada de la primera y la última estación de las vías sean algo mayor consiguiendo así compensar el exceso de tiempo al aproximar a

un valor entero el número de trenes empleados, además, permiten tener un colchón de tiempo para compensar retrasos. A esas dos estaciones se les conoce como cabeceras.

Entonces, el tiempo total *T'* se puede expresar de la siguiente forma:

$$T' = \sum R_i + \sum P_i + T_{c1} + T_{c2} \rightarrow T_{c1} + T_{c2} = T - \sum R_i + \sum P_i \rightarrow$$

$$\rightarrow T_{c1} + T_{c2} = 3150 - 3136 = 14\ s$$

Donde: *Ri* es el tiempo que el tren se está moviendo, expresado en segundos; *Pi* es el tiempo que el tren está parado en estaciones que no son cabecera, expresado en segundos; *Tc1* es el tiempo de colchón en la estación inicial, expresado en segundos; *Tc2* es el tiempo de colchón en la estación final, expresado en segundos.

De acuerdo con lo razonado anteriormente, si la suma de tiempos de parada adicionales en la primera y última estación es 14 s, es preferible que se distribuyan de manera uniforme entre las dos, repartiendo así 7 segundos a cada una y obteniendo los tiempos de cabecera:

$$T_{parada_1} = T_{parada_{18}} = 15 + 7 = 22\ s$$

Finalmente, se concluye entonces que en la vía se emplearán 15 trenes y que los tiempos de parada en estación serán de 15 s excepto en las paradas de cabecera, donde habrá un tiempo de parada de 22 s.

7.2 CÁLCULO DEL HORARIO

El horario marca los tiempos en los que los trenes tienen que entrar y salir de cada estación, así como el tiempo de recorrido entre estaciones; cada tren tiene asignado el suyo propio.

El horario vendrá definido por los tiempos de salida de cada estación que marca el primer tren al recorrer las vías a su marcha nominal, es decir, aplicando la marcha 1 (M1). El horario de dicho tren son los tiempos que marca él mismo, sin embargo, el horario del tren *j* será:

$$H_j = H_1 + (j-1) * 210$$

Por ejemplo, el tren 1 sale de la primera estación en t = 0 y sale de la segunda estación en t = 100. De la misma forma, el tren 2 sale de la estación 1 en t = 210 y tendrá que estar saliendo de la segunda estación en t = 210+100 = 300 s. El mismo razonamiento se aplica a todos y cada uno de los trenes.

Consecuentemente, ha sido necesario crear una nueva propiedad del objeto *Tren* a la que se le ha denominado *vector_horario* la cual recoge en un vector los tiempos de salida de cada estación de cada tren.

Para recoger los datos del horario del primer tren y así poder calcular los horarios de los demás trenes, se ha realizado una simulación para un solo tren recorriendo las 2 vías con marcha 1 (M1) recogiendo en un vector el momento temporal en el que efectúa la salida de cada estación. La Ilustración 15 muestra dicha simulación:

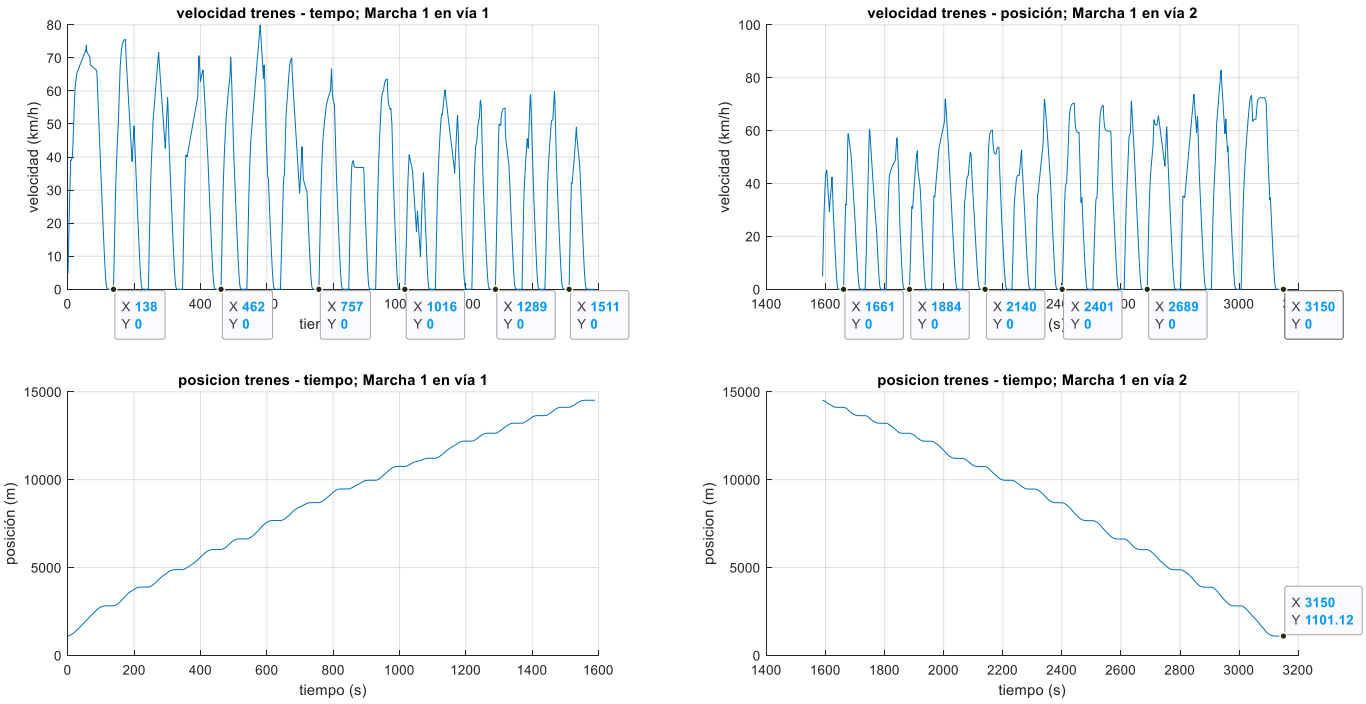

Ilustración 15: cálculo del horario con M1

En la figura se muestran algunos de los momentos en los que el tren abandona las estaciones y que forman parte del horario. Una vez calculado el horario del tren 1, es posible calcular el horario del resto de los trenes. Los resultados obtenidos para el horario de cada uno se muestran en la Tabla 8 de la página siguiente:

	Horarios: tiempo de salida de estación (s)														
Estación	Tren 1	Tren 2	Tren 3	Tren 4	Tren 5	Tren 6	Tren 7	Tren 8	Tren 9	Tren 10	Tren 11	Tren 12	Tren 13	Tren 14	Tren 15
E. 1	0	210	420	630	840	1050	1260	1470	1680	1890	2100	2310	2520	2730	2940
E. 2	138	348	558	768	978	1188	1398	1608	1818	2028	2238	2448	2658	2868	3078
E. 3	243	453	663	873	1083	1293	1503	1713	1923	2133	2343	2553	2763	2973	3183
E. 4	346	556	766	976	1186	1396	1606	1816	2026	2236	2446	2656	2866	3076	3286
E. 5	462	672	882	1092	1302	1512	1722	1932	2142	2352	2562	2772	2982	3192	3402
E. 6	540	750	960	1170	1380	1590	1800	2010	2220	2430	2640	2850	3060	3270	3480
E. 7	641	851	1061	1271	1481	1691	1901	2111	2321	2531	2741	2951	3161	3371	3581
E. 8	757	967	1177	1387	1597	1807	2017	2227	2437	2647	2857	3067	3277	3487	3697
E. 9	847	1057	1267	1477	1687	1897	2107	2317	2527	2737	2947	3157	3367	3577	3787
E. 10	928	1138	1348	1558	1768	1978	2188	2398	2608	2818	3028	3238	3448	3658	3868
E. 11	1016	1226	1436	1646	1856	2066	2276	2486	2696	2906	3116	3326	3536	3746	3956
E. 12	1108	1318	1528	1738	1948	2158	2368	2578	2788	2998	3208	3418	3628	3838	4048
E. 13	1218	1428	1638	1848	2058	2268	2478	2688	2898	3108	3318	3528	3738	3948	4158
E. 14	1289	1499	1709	1919	2129	2339	2549	2759	2969	3179	3389	3599	3809	4019	4229
E. 15	1369	1579	1789	1999	2209	2419	2629	2839	3049	3259	3469	3679	3889	4099	4309
E. 16	1438	1648	1858	2068	2278	2488	2698	2908	3118	3328	3538	3748	3958	4168	4378
E. 17	1511	1721	1931	2141	2351	2561	2771	2981	3191	3401	3611	3821	4031	4241	4451
E. 18	1589	1799	2009	2219	2429	2639	2849	3059	3269	3479	3689	3899	4109	4319	4529
Paso de la vía 1 a la vía 2 ↓ **E. 17**	1661	1871	2081	2291	2501	2711	2921	3131	3341	3551	3761	3971	4181	4391	4601
E. 16	1734	1944	2154	2364	2574	2784	2994	3204	3414	3624	3834	4044	4254	4464	4674
E. 15	1805	2015	2225	2435	2645	2855	3065	3275	3485	3695	3905	4115	4325	4535	4745
E. 14	1884	2094	2304	2514	2724	2934	3144	3354	3564	3774	3984	4194	4404	4614	4824
E. 13	1959	2169	2379	2589	2799	3009	3219	3429	3639	3849	4059	4269	4479	4689	4899

E. 12	2064	2274	2484	2694	2904	3114	3324	3534	3744	3954	4164	4374	4584	4794	5004
E. 11	2140	2350	2560	2770	2980	3190	3400	3610	3820	4030	4240	4450	4660	4870	5080
E. 10	2231	2441	2651	2861	3071	3281	3491	3701	3911	4121	4331	4541	4751	4961	5171
E. 9	2310	2520	2730	2940	3150	3360	3570	3780	3990	4200	4410	4620	4830	5040	5250
E. 8	2401	2611	2821	3031	3241	3451	3661	3871	4081	4291	4501	4711	4921	5131	5341
E. 7	2505	2715	2925	3135	3345	3555	3765	3975	4185	4395	4605	4815	5025	5235	5445
E. 6	2611	2821	3031	3241	3451	3661	3871	4081	4291	4501	4711	4921	5131	5341	5551
E. 5	2689	2899	3109	3319	3529	3739	3949	4159	4369	4579	4789	4999	5209	5419	5629
E. 4	2802	3012	3222	3432	3642	3852	4062	4272	4482	4692	4902	5112	5322	5532	5742
E. 3	2906	3116	3326	3536	3746	3956	4166	4376	4586	4796	5006	5216	5426	5636	5846
E. 2	3007	3217	3427	3637	3847	4057	4267	4477	4687	4897	5107	5317	5527	5737	5947
E. 1	3150	3360	3570	3780	3990	4200	4410	4620	4830	5040	5250	5460	5670	5880	6090

Tabla 8: resultado de los horarios para cada tren

Cabe destacar que los horarios mostrados en la Tabla 8 son válidos cuando los trenes se encuentran recorriendo por primera vez las vías 1 y 2. Si por el contrario se encontraran en, por ejemplo, la segunda vuelta a las vías 1 y 2, habría que sumar a sus horarios el tiempo total de recorrido *T'* (*T'* = 3150 s). De esta forma:

$$H_{ji} = H_{j0} + (i - 1) * T'$$

Donde: *Hji* es el horario del tren *j* en la vuelta *i*; *Hj0* es el horario del tren *j* en su primera vuelta; *T'* es el tiempo total de recorrida de una vuelta a las vías 1 y 2, expresado en s.

7.3 SALIDA ESCALONADA DE LOS TRENES

Una vez que es conocida la cantidad de trenes que se emplearán y el intervalo de tiempo (*I* = 210 s) entre ellos, es necesario diseñar un algoritmo que consiga que los trenes salgan de la primera estación de la vía 1 escalonadamente cada 3.5 minutos.

Para ello, se ha decidido crear una propiedad constante del objeto *Tren* llamada *intervalo* la cual recoge el tiempo que tiene que esperar cada tren para salir desde que ha iniciado el recorrido el de delante y que es igual al valor del intervalo, 210 s.

El algoritmo desarrollado para este objetivo se encuentra en la función *SimulaTrafico* la cual figura en el Anexo I. Función de Simulación de Tráfico. Dentro de dicha función se han creado las siguientes variables:

- **Variable *trenes_circulando***. Indica la cantidad de trenes que se encuentran recorriendo la vía.
- **Variable *aux_intervalo***. Variable auxiliar que servirá para compararla con el valor de tiempo de intervalo.

Como se había descrito anteriormente, en la función *SimulaTrafico* existen dos bucles: el referente al tiempo que dura la simulación (bucle tipo *while*) y el referente a que se ejecute cada iteración de tiempo para cada tren (bucle tipo *for*).

Habiendo inicializado la variable *trenes_circulando* a 1 y la variable *aux_intervalo* a 1, se impondrá que el bucle tipo *for* se realice desde el tren número 1 hasta la cantidad de trenes que estén circulando, es decir, hasta el valor que tome la variable *trenes_circulando*. De esta forma se consigue que las iteraciones de tiempo no se simulen para trenes que todavía no están en vía.

Posteriormente, en cada iteración de tiempo se añadirá una unidad a la variable *aux_intervalo* y será con una condición de tipo *if* con la que se dará orden de salir al siguiente tren si *aux_intervalo* es igual al valor del intervalo (I =210 s), lo que se traduce en aumentar en una unidad la variable *trenes_circulando*.

En el caso de que ya todos los trenes se encuentren circulando, es decir, que la variable *trenes_circulando* haya alcanzado el valor de 15, se actualizará la variable *aux_intervalo* a 0 en cada iteración de tiempo para que así no haya manera de que supere los 210 s.

En la Ilustración 16 se muestra una simulación realizada para la comprobación de la salida escalonada de los trenes. Cabe de destacar que en dicha simulación ya se han establecido las condiciones que se concluyeron en el apartado 7.1 Cálculo del Número de Trenes Empleados.

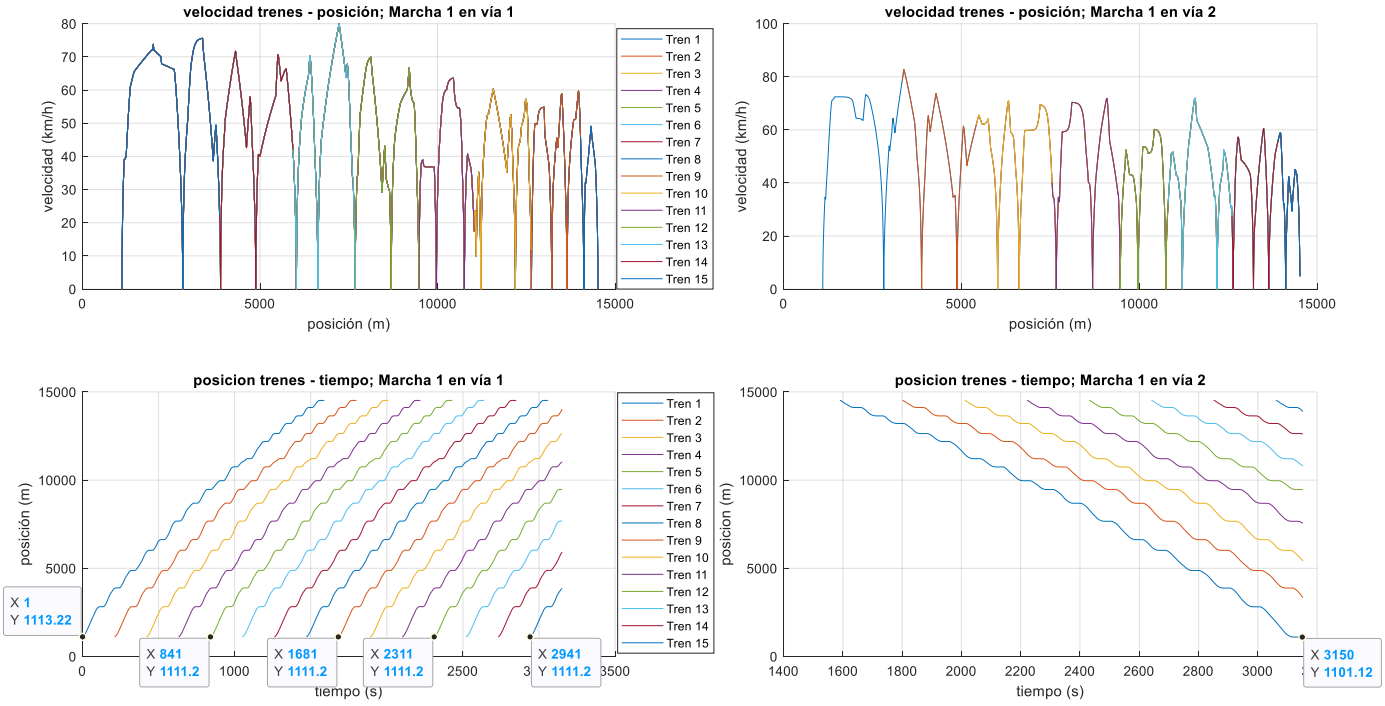

Ilustración 16: simulación de puesta en servicio de los 15 trenes

7.4 CÁLCULO DEL TIEMPO GANADO O PERDIDO CON M0, M2 Y M3

Es normal que en los trenes de metro se produzcan retrasos o adelantos respecto del horario. Cuando eso sucede, es necesario que el tren aplique una marcha distinta de la nominal (M1) para corregir sus tiempos: si el tren sufre un retraso, se aplicará la única marcha rápida, que es M0; si por el contrario sufre un adelanto, se aplicará una de las dos marchas lentas, donde M2 es más rápida que M3.

En el apartado 7.5 Modelo del ATR: Cálculo de la Corrección y Consigna de Aplicación de Marcha, se explicará cómo se compara el retraso que lleva el tren respecto de su horario y cómo se realiza la consigna de aplicar una marcha u otra, pero para conseguir eso, primero hay que saber los segundos que se ganan o se pierden entre estaciones respecto de la marcha

1 (M1) con cada marcha (M0, M2, M3). Por ejemplo, si el tren sufre un adelanto y necesita perder 6 s entre la estación *i* y la estación *j*, se tiene que saber qué marcha ofrece 6 s de retraso entre dichas estaciones o cuál es la que se queda más próxima.

Para ello se ha hecho circular un solo tren por las vías 1 y 2 a cada una de las marchas, se han recogido los tiempos de salida de estación y se han comparado con los tiempos de salida de estación que marcaba el tren cuando utilizaba la marcha 1 (M1). De esta manera se obtienen los segundos ganados o perdidos entre estaciones y están los datos preparados para compararlos con los segundos que necesita ganar o perder el tren entre estaciones ejecutando así la consigna de aplicación de marcha pertinente.

En las ilustraciones Ilustración 17, Ilustración 18 e Ilustración 19 se muestran las simulaciones de los recorridos del tren a las vías 1 y 2 en marchas M0, M2 y M3 respectivamente. Se han marchado algunos de los valores temporales en los que el tren efectúa la salida de estación:

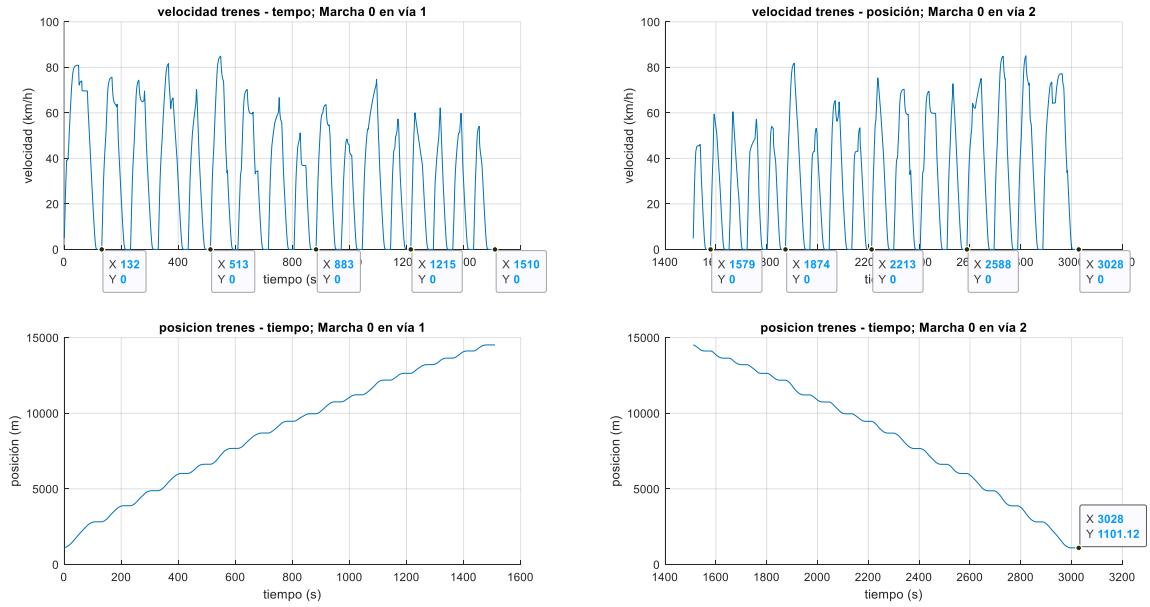

Ilustración 17: cálculo de los tiempos del tren con M0

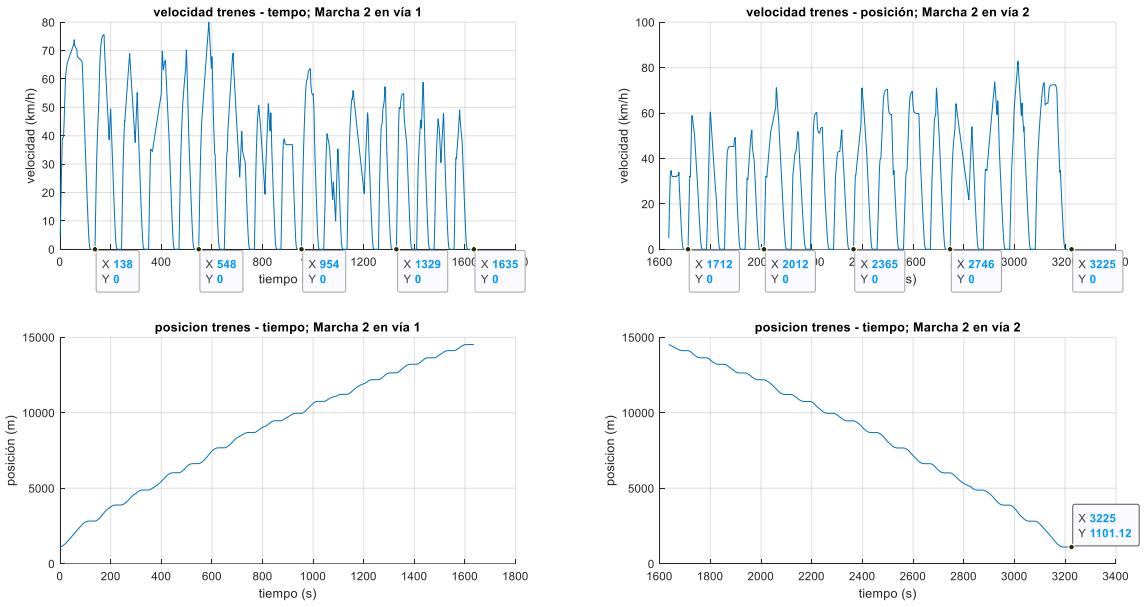

Ilustración 18: cálculo de los tiempos del tren con M2

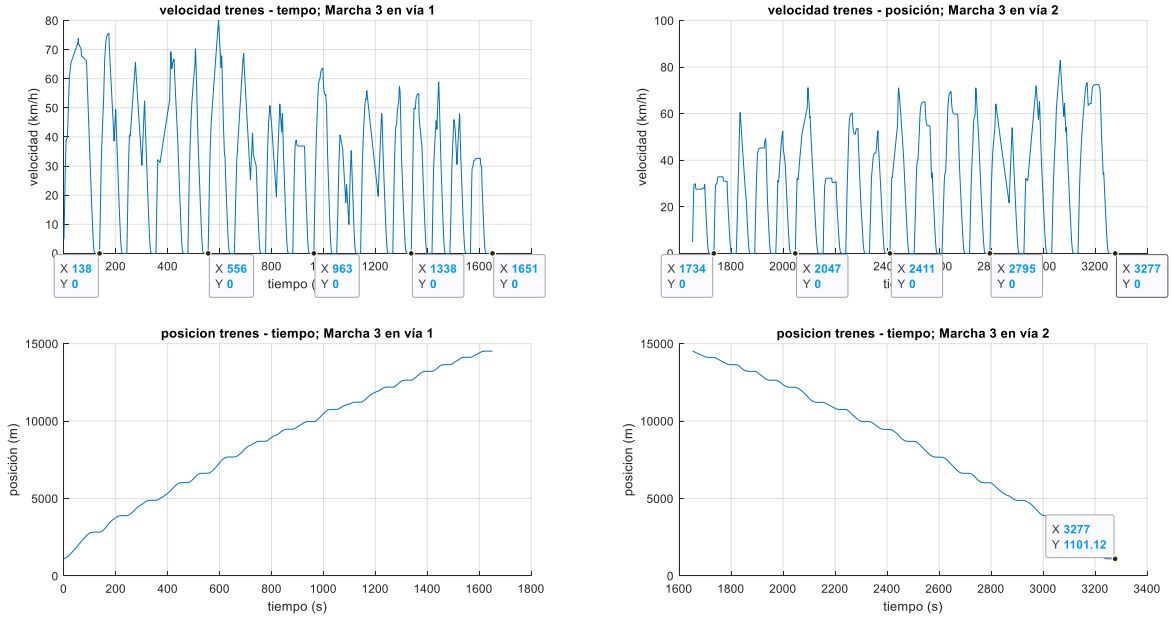

Ilustración 19: cálculo de los tiempos del tren con M3

A modo de resumen, en la Tabla 9 de la siguiente página se muestran los tiempos realizados a marchas M0, M1 y M2 comparados con los realizados a marcha M1, así como los segundos ganados o perdidos entre estaciones:

	Horario (s)	**En cada marcha, col1: tiempo (s); col2 tiempo ganado o perdido (s)**					
Estación	Tren 1	Marcha 0		Marcha 2		Marcha 3	
E. 1	0	0	0	0	0	0	0
E. 2	138	132	-6	138	0	138	0
E. 3	243	233	-4	243	0	243	0
E. 4	346	330	-6	350	4	355	9
E. 5	462	435	-11	470	4	478	7
E. 6	540	513	0	548	0	556	0
E. 7	641	609	-5	649	0	657	0
E. 8	757	716	-9	768	3	777	4
E. 9	847	806	0	873	15	882	15
E. 10	928	883	-4	954	0	963	0
E. 11	1016	971	0	1042	0	1051	0
E. 12	1108	1045	-18	1134	0	1143	0
E. 13	1218	1144	-11	1258	14	1267	14
E. 14	1289	1215	0	1329	0	1338	0
E. 15	1369	1294	-1	1409	0	1418	0
E. 16	1438	1362	-1	1478	0	1487	0
E. 17	1511	1435	0	1557	6	1566	6
E. 18	1589	1510	-3	1635	0	1651	7
Paso de la vía 1 a la vía 2 ↓ **E. 17**	1661	1579	-3	1712	5	1734	11
E. 16	1734	1652	0	1785	0	1820	13
E. 15	1805	1723	0	1856	0	1891	0
E. 14	1884	1802	0	1937	2	1972	2
E. 13	1959	1874	-3	2012	0	2047	0
E. 12	2064	1970	-9	2119	2	2154	2
E. 11	2140	2046	0	2195	0	2241	11
E. 10	2231	2135	-2	2286	0	2332	0
E. 9	2310	2213	-1	2365	0	2411	0
E. 8	2401	2300	-4	2458	2	2504	2
E. 7	2505	2404	0	2562	0	2611	3
E. 6	2611	2510	0	2668	0	2717	0
E. 5	2689	2588	0	2746	0	2795	0
E. 4	2802	2695	-6	2877	18	2926	18
E. 3	2906	2788	-11	2981	0	3033	3
E. 2	3007	2887	-2	3082	0	3134	0
E. 1	3150	3028	-2	3225	0	3277	0

Tabla 9: tiempos ganados o perdidos entre estaciones con cada marcha

Cabe destacar que el tiempo que se gana o se pierde entre estaciones con cada marcha es discreto, por ese motivo no siempre existirá una marcha que corrija el retraso o adelanto a la perfección, pero se escogerá la que más se adecúe.

7.5 MODELO DEL ATR: CÁLCULO DE LA CORRECCIÓN Y CONSIGNA DE APLICACIÓN DE MARCHA.

El regulador implementado es un algoritmo heurístico propuesto por Araya, S. [5]. Este algoritmo, además de los retrasos que puedan sufrir los trenes, o desviación de horario, también tiene en cuenta que pueden existir errores en el intervalo entre ellos, es decir que, aunque el tren siga su horario, es posible que el tren que lleva delante o detrás haya sufrido un retraso y que no se respete el intervalo de tiempo. A esto se le conoce como error de centrado de intervalo.

La corrección es un valor que indica el tiempo que tiene que ganar o perder cada tren entre dos estaciones, la cual tiene en cuenta el retraso horario y el error de centrado de intervalo. Se puede obtener con la siguiente expresión:

$$corrección = \max\left[-dh,; -dIc - \frac{h}{2}\right]$$

Donde: *dh* es el desvío de horario, expresado en segundos; *dIc* es el error de centrado de intervalo; *h* es el parámetro de ajuste. Si *h* es grande, la regulación es más fiel al horario que al intervalo y es más estable.

Cabe destacar que, si la corrección es positiva, la consigna es de perder tiempo entre estaciones y que, si es negativa, la consigna será de ganar tiempo entre estaciones.

Una vez calculada la corrección, se traducirá su valor en una orden de marcha. Por ejemplo, si en una iteración de tiempo se obtiene una corrección de -5 s, el tren debe de ganar ese tiempo escogiendo la marcha que lo consiga.

Para obtener dicha corrección, inicialmente habrá que calcular ambos errores, los cuales se conocen como indicadores de calidad del servicio:

- **Desviación de horario**. Es la diferencia entre los tiempos que está marcando el tren al recorrer la vía y los que tendría que estar marcando según su horario. Se calcula de la siguiente manera:

$$dh_j = t_j - H_j$$

  Donde: dh_j es el desvío horario del tren *j*, expresado en segundos; t_j es el tiempo que está marcando el tren *j* en el recorrido, expresado en segundos; H_j es el horario del tren *j*, expresado en segundos.

- **Error de centrado de intervalo.** Es el error que existe entre dos trenes respecto de del intervalo. Se calcula con la siguiente expresión;

$$dIc_j = dh_j - \frac{1}{2}(dh_{j+1} + dh_{j-1})$$

  Donde: dIc_j es el error de centrado de intervalo del tren *j*, expresado en segundos; dh_j es el desvío horario del tren *j*; dh_{j+1} es el desvío horario del tren que va detrás del *j*; expresado en segundos; dh_{j-1} es el desvío horario del tren que va delante del *j*, expresado en segundos.

En cada iteración de tiempo se calculará la corrección mediante la llamada a una función la cual recibe el objeto *Tren* y el parámetro *h y* devuelve la corrección. Dicha función recibe el nombre *CalculaCorreccion* en el código y figura en el Anexo I. Función para el Cálculo de la Corrección.

Por lo tanto, para calcular el desvío de horario y el error de centrado de intervalo para cada tren y poder operar con ellos, se definen las siguientes propiedades variables del objeto *Tren*:

- **Propiedades *RH_delante; RH; RH_detras***. Guardan respectivamente el desvío horario del tren que va delante, el desvío horario del propio tren y el desvío horario del tren que va detrás.
- **Propiedad *RCI***. Guarda el error de centrado de intervalo de cada tren.

Los desvíos de horario y errores de centrado de intervalo se calculan en cada iteración de tiempo en las que el tren se encuentra realizando su parada en cada estación. Dichos cálculos figuran en la función *movimientoacelerando* en el Anexo I. Clase Tren.

Una vez se ha calculado la corrección, es necesario crear una función que reciba los objetos *Tren*, *Linea*, *Linea2* y que devuelva la marcha que se le va a dar al tren para corregir los tiempos, disparando en el momento en el que el tren sale de la estación tras una parada. Dicha función se ha nombrado como *AplicarMarcha* y figura en el Anexo I. Aplicación de Marcha.

La dinámica interna de la función es el siguiente: la corrección es el tiempo que el tren tiene que llevar a cero y para eso, hay que aplicar la marcha que más se aproxime a ella saturándola por los lados. Es decir, si se requiere una marcha muy lenta que no se puede dar, se aplicará la marcha más lenta (M3). Si por el contrario se exige una marcha demasiado rápida, se aplicará la más rápida (M0). En el caso de requerir un tiempo de corrección entre estos dos extremos, se aplicará la marcha más rápida que proporciona el tiempo deseado, proporcionándose nunca así una marcha más lenta que la corrección deseada para no provocar retrasos adicionales.

Un ejemplo de la explicación anterior podría ser el siguiente: en un punto de recorrido se tienen las marchas M0: -2 s; M1: 0 s; M2: 7 s; M3: 14 s y una corrección de 12 s. Aunque se necesiten perder 12 s y la marcha M3 sea más próxima de dicha corrección, se aplicará la marcha M2 de 7 s para no provocar retrasos.

Otro factor importante para anular la corrección es el tiempo de parada extra. Es probable que no se consiga corregir la totalidad del tiempo exigido. Es por ello por lo que será necesario aplicar un tiempo de parada extra en la estación para alcanzar la corrección

deseada. El tiempo de parada extra máximo se ha fijado a un valor de 20 s. Por el contrario, si el tren necesita perder tiempo, no se podrá modificar el tiempo de parada para hacer que permanezca menos tiempo en la estación ya que los 15 segundos de estacionamiento son mínimos.

Para afianzar el concepto anterior, se presenta el siguiente ejemplo: se obtiene una corrección de 16 s en un tramo de la vía donde se tienen las marchas M0: -2 s; M1: 0 s; M2: 7 s; M3: 14 s. Consecuentemente, se aplicará la marcha M3 y un tiempo de parada extra de 2 segundos para llevar a cero el valor de la corrección.

El cálculo de tiempo de parada extra explicado anteriormente se realiza en la función *TiempoParadaExtra* la cual recibe los objetos del modelo y devuelve el tiempo de parada extra y el objeto *Tren.* Dicha función figura en el Anexo I. Tiempo de Parada Extra.

La función se ejecutará cuando el tren llega a una estación ya que, cuando realiza su parada, se sabe cuándo va a salir y, por lo tanto, se puede calcular la corrección entre el tramo de la parada en la que está y la siguiente. De esta manera, el tren efectuará su salida de estación cuando la corrección pueda ser anulada solamente con las marchas o cuando ha llegado al tiempo de parada máximo.

Es importante destacar que sólo se aplicará tiempo de parada extra cuando solo con la marcha no se realice la corrección, es decir, cuando se selecciona M3 (marcha más lenta) y ésta no es capaz de anular el tiempo de corrección. Por ejemplo, se presenta un caso en el que la corrección es de 20 s en un tramo de la vía donde se tienen las marchas M0: -2 s; M1: 0 s; M2: 7 s; M3: 14 s. En ese caso se aplicará la marcha M3 y un tiempo de parada extra de 6 s. Si por el contrario se tiene una corrección de 10 s para el mismo tramo, no existirá tiempo de parada extra y se perderán los 7 segundos de la marcha M2.

A modo de resumen, para llevar a cero la corrección existen dos tiempos: el que proporciona la marcha y el que proporciona el tiempo de parada extra. Consecuentemente, el momento de salida del tren de una estación es cuando el tiempo de parada extra sea 0 segundos o cuando se realice la corrección con las marchas.

Una vez diseñado el modelo de regulación del tráfico, cuando se produce una incidencia de retraso en uno de los trenes, se conseguirá que el propio tren vuelva a su horario y que además los trenes que van delante de él reduzcan su marcha para que el error de centrado de intervalo no sea excesivo. En la Ilustración 20 se presenta una figura esquema de la situación:

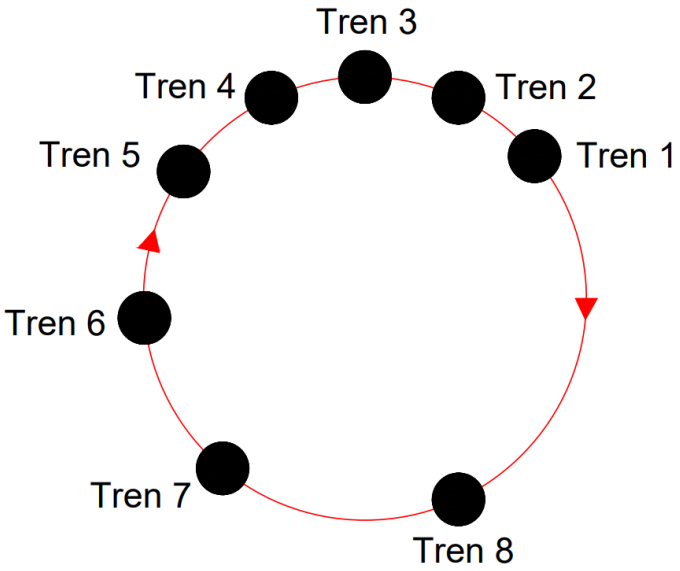

Ilustración 20: esquema de regulación del tráfico

Según se puede observar, el Tren 1 sufre un importante retraso y el intervalo entre dicho tren y el Tren 8 aumenta. Será necesario que el Tren 8 decelere su marcha para estabilizar el intervalo e incluso puede que suceda lo mismo con el Tren 7. Una vez que el Tren 1 reanude su marcha, éste deberá volver a su horario nominal, así como los trenes que han reducido su marcha para regularizar el intervalo, siendo entonces nulo el error de horario y el error de centrado de intervalo de todos los trenes.

El procedimiento descrito anteriormente se simulará y analizará en el Capítulo 8. Análisis de Resultados para comprobar que el simulador de tráfico funciona correctamente.

7.6 RETRASOS EN ESTACIÓN POR SUBIDA Y BAJADA DE PASAJEROS

Adicionalmente, hay que tener en cuenta la posibilidad de que en las paradas en estación se produzcan incidencias aleatorias y retrasos debido a la subida y bajada de pasajeros. Dicho fenómeno se simulará con una distribución lognormal con $\mu = 15$ (parada nominal del tren en la estación) y $\sigma = 3$. De esta forma se podrá comprobar que el tren también modifica su marcha para llevar a cero los retrasos producidos por dichas incidencias. Es necesario puntualizar que se empleará una distribución estadística lognormal puesto que en el artículo *Statistical dwell time model for metro lines* [6] se realizó un análisis de tiempos de parada de Metro Madrid llegando a la conclusión de que es la mejor distribución para este suceso.

Para llevar a cabo lo anterior, se ha creado una función llamada *ParadaLognormal* la cual recibe el objeto *Tren*, genera un número aleatorio siguiendo la distribución lognormal y devuelve el tiempo que se le debe de añadir a la parada nominal en segundos. Cabe destacar que no es posible que el tiempo de parada sea menor que 15 s, por ese motivo, se impone que, si el número aleatorio generado es menor que 15, se devuelvan 0 segundos para añadir al tiempo de parada nominal. Esta función figura en el Anexo I. Distribución Lognormal.

Para comprobar el correcto funcionamiento de los retrasos por incidencias de pasajeros se realiza una simulación del recorrido de un tren a lo largo de la vía 1. En la Ilustración 21 se representa el retraso horario frente al tiempo y se puede apreciar cómo los tiempos de parada son variables dependiendo de la distribución lognormal, provocando así disparidades con su horario. Consecuentemente, el tren modificará su marcha para intentar llevar los retrasos a cero.

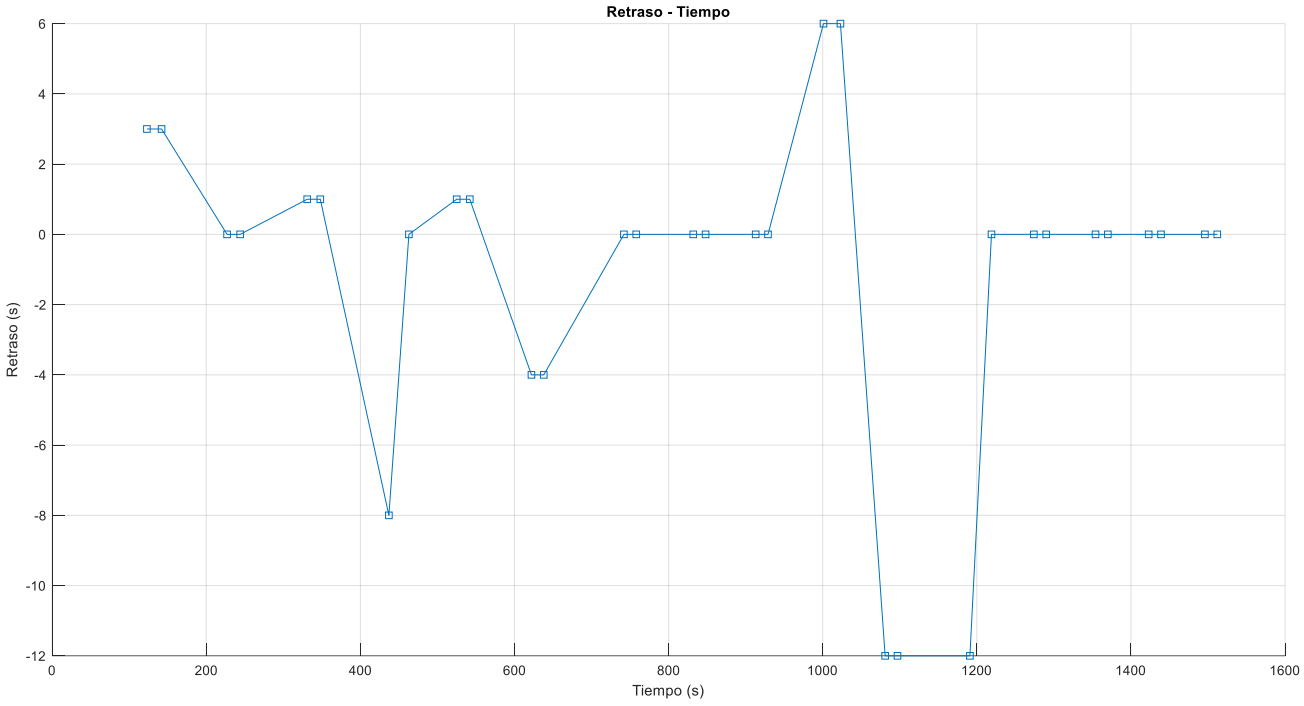

Ilustración 21: retraso producido por incidencias de pasajeros

Para entender mejor el comportamiento del tren, en la Ilustración 22 se representa la velocidad de los trenes respecto del tiempo, donde se puede ver los tiempos de parada que realiza en estación.

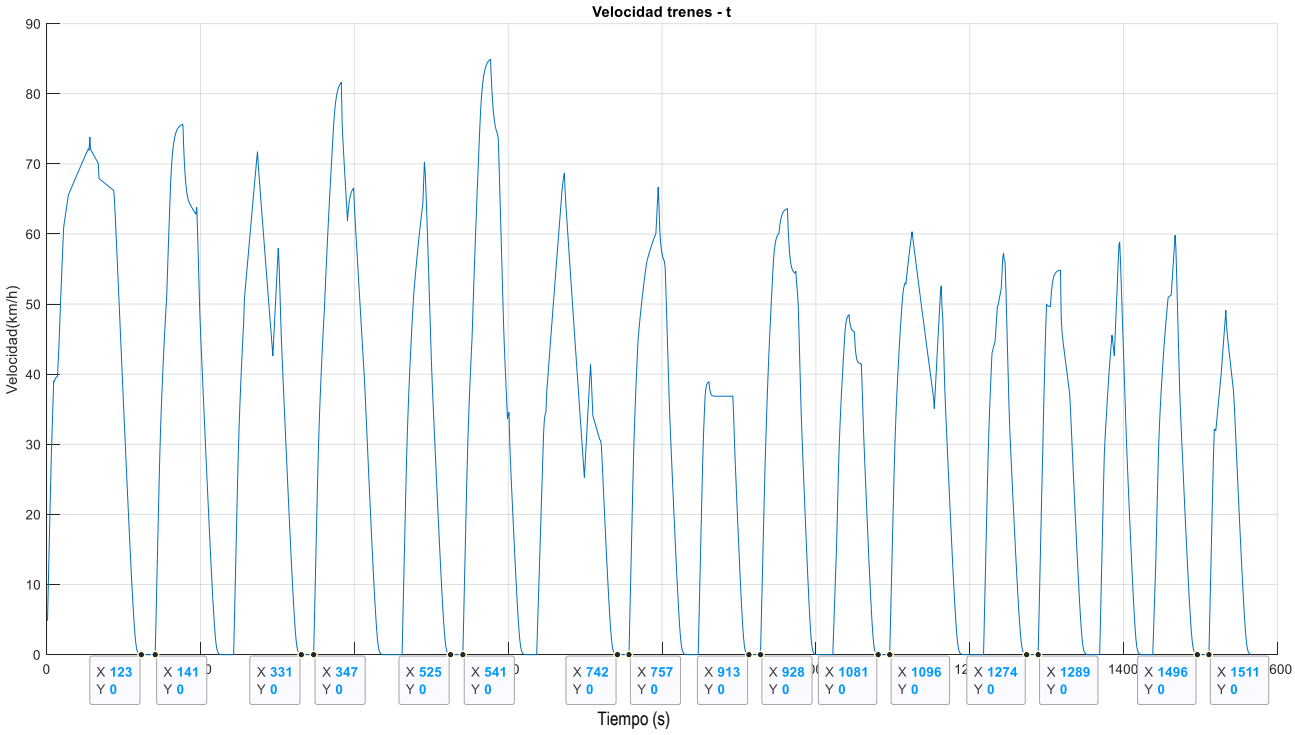

Ilustración 22: velocidad frente al tiempo; simulación de incidencia

A modo de resumen, en la Tabla 10 se muestran los tiempos de parada para la misma simulación:

Estación	Tiempo de parada (s)
Est. 2	18
Est. 3	16
Est. 4	16
Est. 5	25
Est. 6	16
Est. 7	15
Est. 8	15
Est. 9	15
Est. 10	15

Est. 11	21
Est. 12	15
Est. 13	27
Est. 14	15
Est. 15	15
Est. 16	15
Est. 17	15

Tabla 10: tiempos de parada; simulación de incidencia

Capítulo 8. ANÁLISIS DE RESULTADOS Y CONCLUSIONES

Una vez que el modelo de regulación automática del tráfico ha sido diseñado completamente, se procede a comprobar los resultados y correcto funcionamiento. Para ello se simulará la siguiente incidencia:

Una vez que todos los trenes se hayan puesto en vía, se simulará un retraso de 5 minutos en el tren 2. Para ello, se impondrá un tiempo de parada de 5 minutos en la estación 2 del segundo recorrido de la vía 1, ya que el primer recorrido de las líneas 1 y 2 se realiza para la puesta en circulación de los trenes.

Para la misma simulación se van a estudiar dos situaciones:

1. Sin tener en cuenta las incidencias producidas por los pasajeros en la parada en estación.
2. Teniendo en cuenta las incidencias aleatorias producidas por los pasajeros en la parada en estación.

8.1 SIMULACIÓN SIN INCIDENCIAS POR PASAJEROS

Lo primero que se va a comprobar y demostrar es que cuando el tren realiza su parada de 5 minutos, los trenes que circulan por detrás se detienen para evitar la colisión. En la Ilustración 23 se muestra la posición frente al tiempo en detalle cuando se produce el retraso de 5 minutos en la estación 2 de la vía 1:

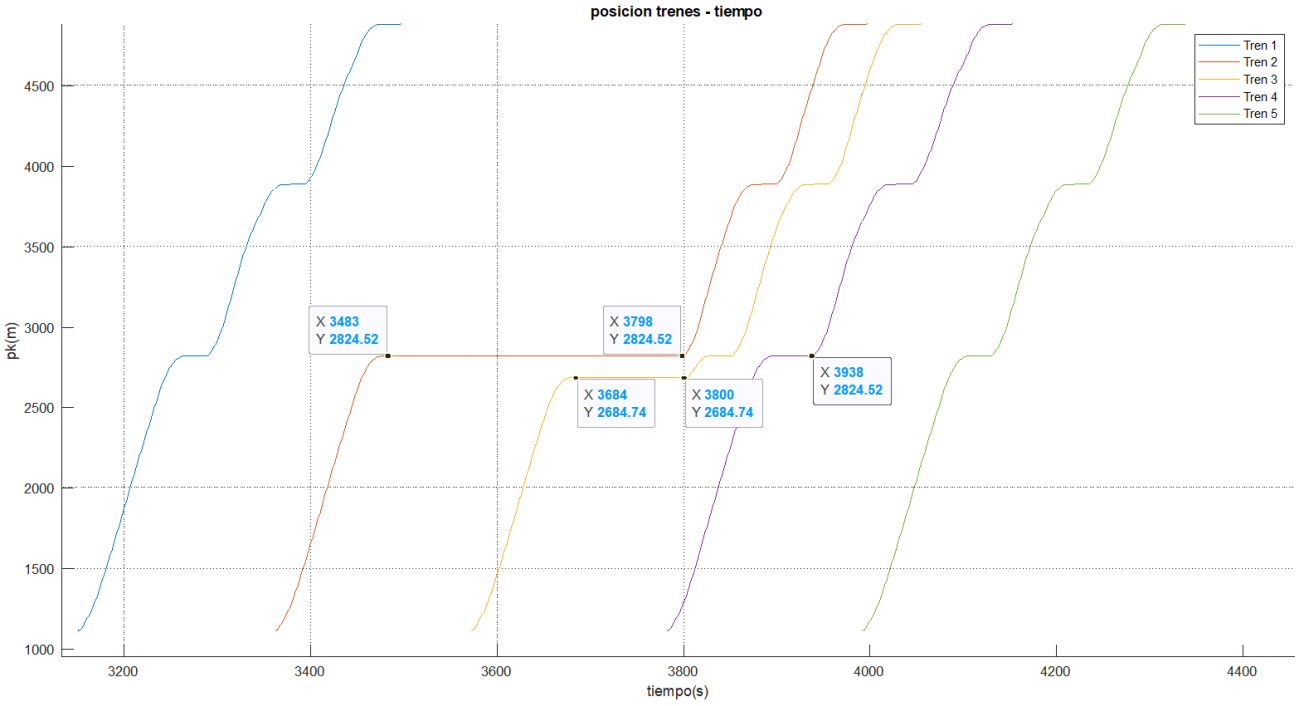

Ilustración 23: frenado de trenes

El tren 2 realiza un tiempo de parada en la estación 2 de 315 segundos, es decir, los 15 segundos de parada nominal más los 300 segundos del retraso. En ese momento el tren 3, que se encuentra circulando detrás del tren 2, alcanza la distancia mínima de seguridad y se ve obligado a detenerse durante 116 segundos hasta que el tren 2 continúa su marcha. Se recuerda que la distancia de seguridad se ha impuesto a 50 m desde la cola del tren de delante y, como la longitud del tren es de 89.16 m, la distancia mínima entre trenes debe de ser de 139.16 m. En este caso el tren 3 se detiene a 139.78 m, respetando así la distancia mínima de seguridad entre trenes.

Cabe destacar que el tren 4 no se detiene ya que no alcanza dicha distancia, sin embargo, se aproxima al tren 3, lo que produce un error de centrado de intervalo.

Debido a la perturbación impuesta, el tren 2, tren 3 y tren 4 sufren retrasos en sus horarios y, adicionalmente, se producen errores de centrado de intervalo ya que el intervalo de 210 segundos entre trenes no se respeta.

Lo que entonces hará el simulador de regulación del tráfico será:

a. Provocar una retención en los trenes que circulan por delante del que sufre la perturbación para intentar respetar el intervalo entre trenes. Es decir, cuando el tren 2 sufre el retraso de 5 minutos, el intervalo entre él y el tren 1 es de 210 + 300 = 510 segundos. Es necesario que el tren 1 reduzca su marcha para reducir ese tiempo y dar buen servicio a los pasajeros.
b. Devolver a los trenes retrasados a su horario nominal.

Es importante destacar que para esta simulación se ha impuesto un parámetro de ajuste *h*=80.

Por un lado, en la Ilustración 24 se muestra la evolución de los retrasos horarios del tren sujeto (tren 2) y de los trenes que circulan delante de él:

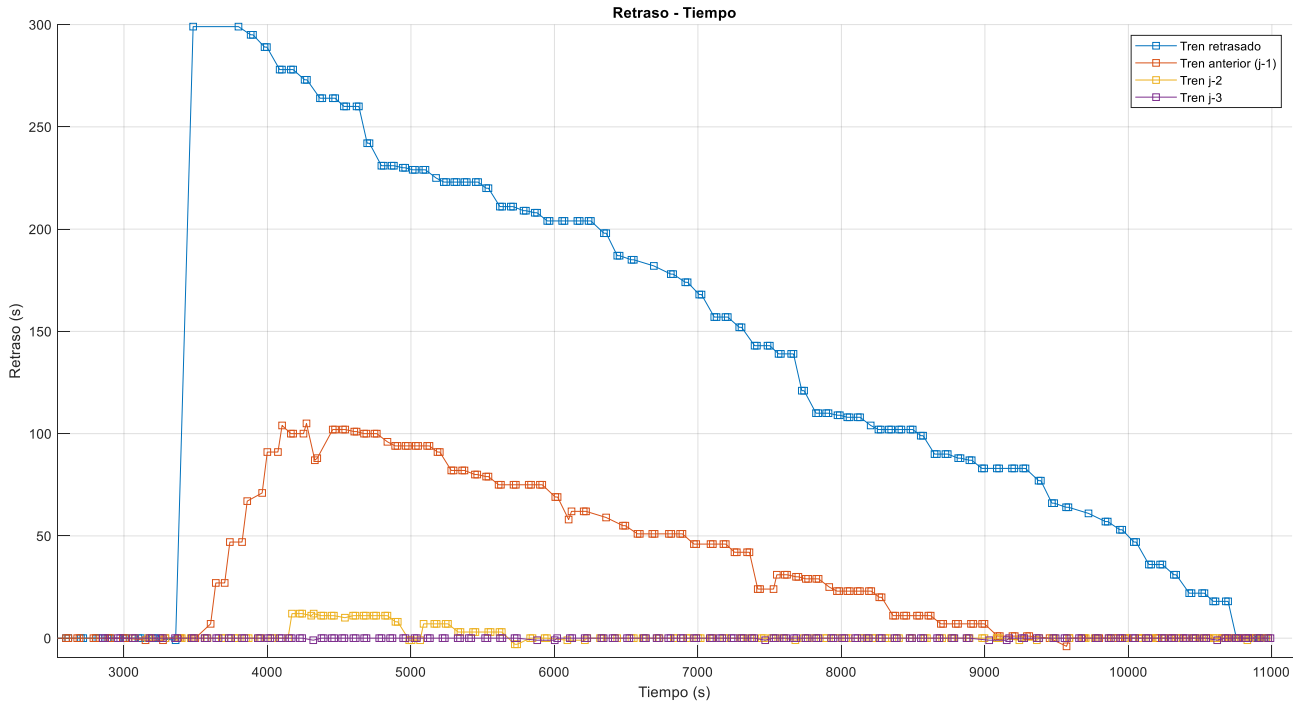

Ilustración 24: evolución del RH; trenes j, j-1, j-2, j-3

Cuando el intervalo entre trenes aumenta, el regulador de tráfico automáticamente impone una reducción de la marcha de los trenes de delante hasta que el retraso horario tiene más

peso en la corrección que el error de centrado de intervalo. Así mismo, transcurrido el periodo de tiempo necesario, los retrasos en el horario son eliminados.

Por otro lado, en la Ilustración 25 se muestra la evolución de los retrasos horarios en los trenes que circulan detrás del tren sujeto:

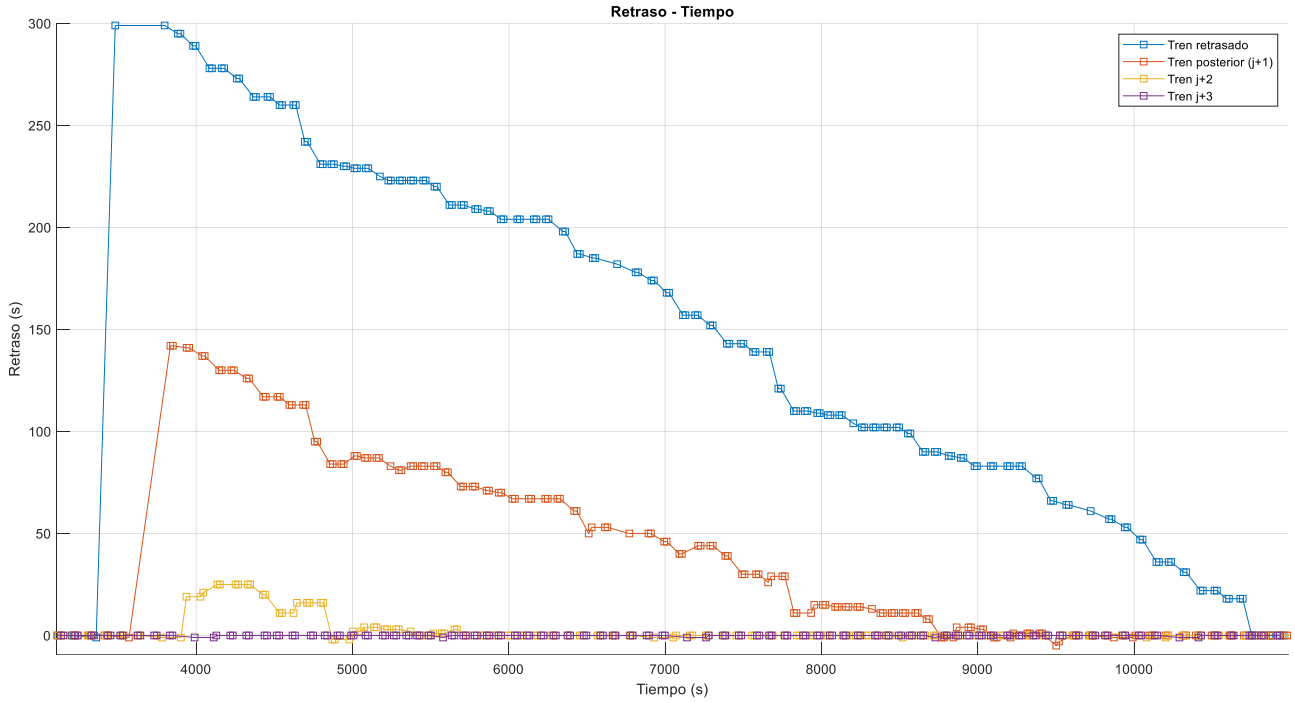

Ilustración 25: retraso de los trenes j, j+1, j+2, j+3

De la misma manera, tras el tiempo necesario transcurrido, los trenes vuelven a su horario nominal.

Finalmente, en la Ilustración 26 se representan la evolución de los retrasos horarios del tren sujeto, de los tres trenes anteriores y de los tres trenes posteriores:

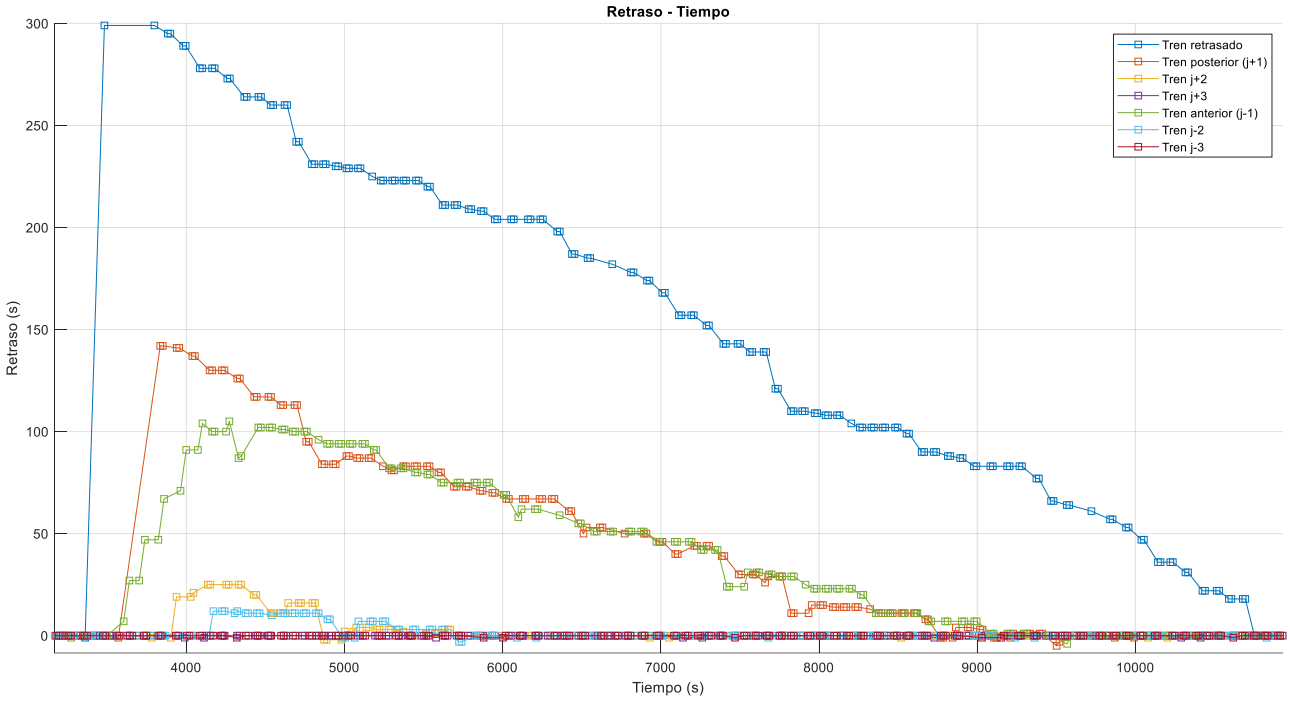

Ilustración 26: resultados conjuntos para h=80

De la misma forma, es interesante comprobar el efecto que el parámetro de ajuste *h* tiene en la regulación del tráfico. Para ello, se realizará la misma simulación, en este caso con un parámetro de ajuste *h* = 45. En la Ilustración 27 se muestran comparadas las evoluciones de los retrasos horarios de los trenes que circulan delante del tren retrasado con *h* = 80 y *h* = 45:

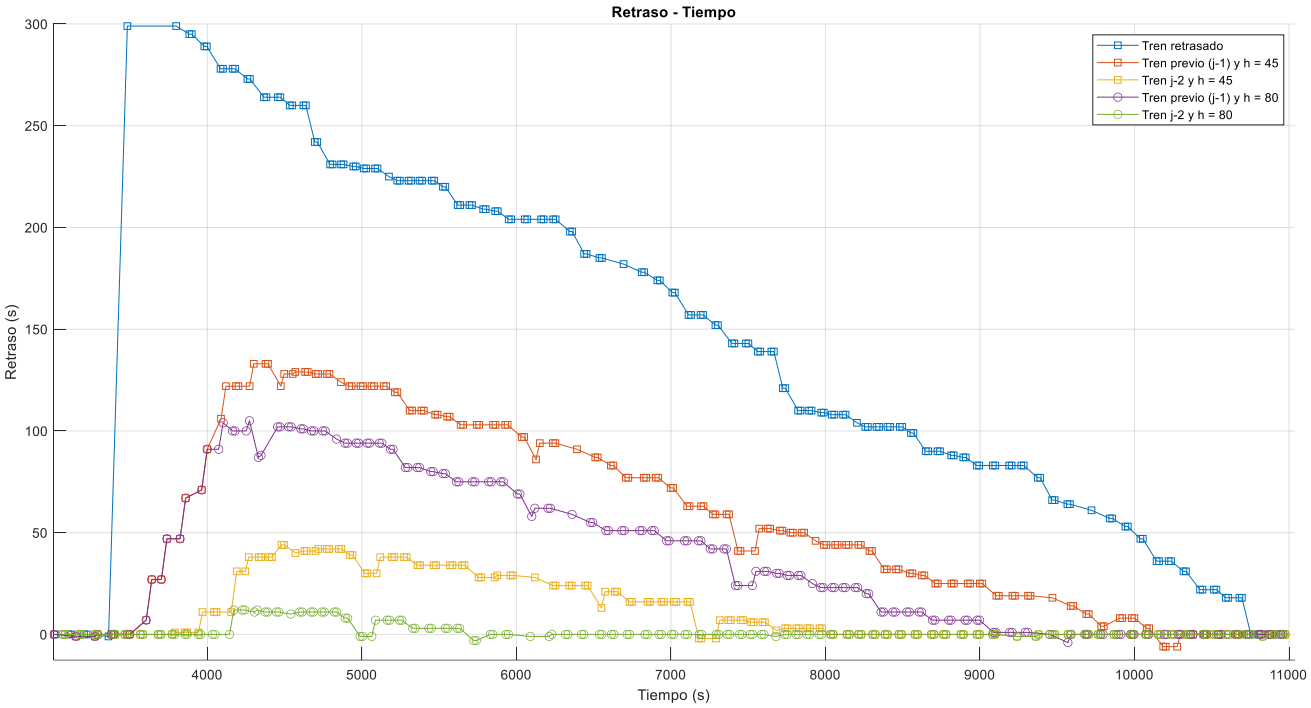

Ilustración 27: comparación h=80 vs h=45

Cuando el parámetro de ajuste *h* es menor, se produce una mayor regulación por intervalo y menor regulación por horario y, consecuentemente, los trenes tardan más tiempo en volver a sus horarios nominales.

8.2 EFECTO DE INCIDENCIAS ALEATORIAS POR PASAJEROS EN LA REGULACIÓN DEL TRÁFICO

En este apartado se quiere estudiar el efecto que tiene sobre la regulación del tráfico las incidencias aleatoriasy retrasos que producen los pasajeros al subir y bajar de los trenes en estación. Para ello, se ha realizado la misma simulación que en el apartado 8.1, con un parámetro de ajuste $h = 80$.

Por un lado, en la Ilustración 28 se observa la evolución de los retrasos horarios del tren sujeto y de los tres que lleva delante y de los horarios del tren sujeto y los tres que lleva detrás:

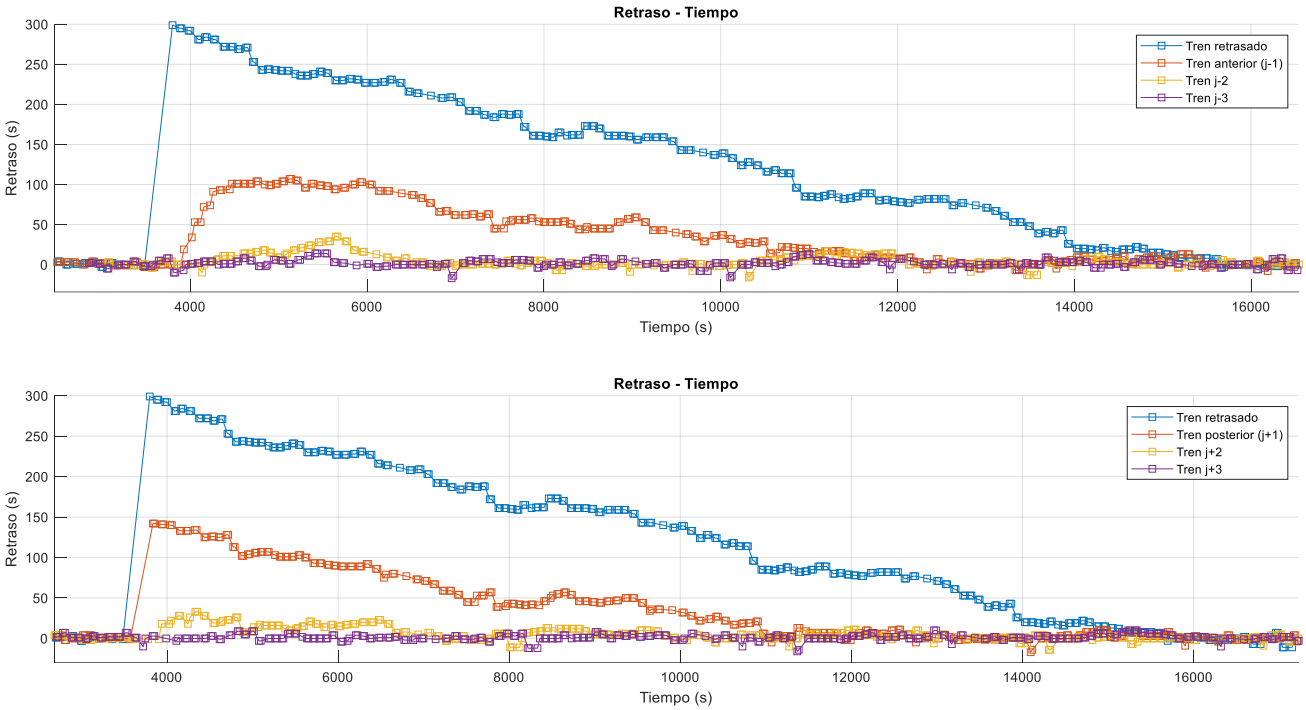

Ilustración 28: evolución de los retrasos con incidencias provocadas por pasajeros

Finalmente, a modo de comparación, en la Ilustración 29 se muestra la evolución de los retrasos horarios del tren sujeto y los dos que lleva delante cuando existe incidencia por parte de los pasajeros y cuando no:

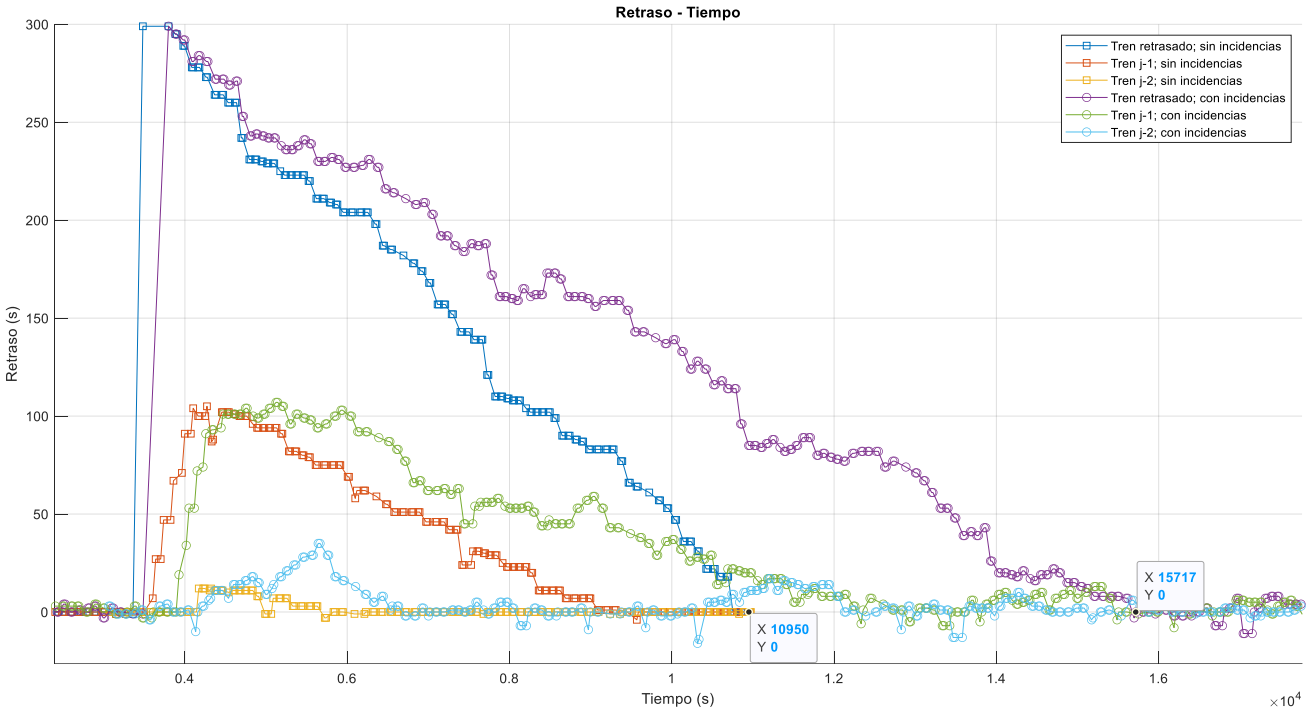

Ilustración 29: comparación de la evolución en los retrasos cuando hay incidencias por pasajeros y cuando no

Cuando existen incidencias por parte de los pasajeros, el tiempo requerido para corregir los errores de horario es considerablemente mayor, concretamente en esta simulación y para el tren al que se le ha impuesto el retraso de 5 minutos, se necesitan 4767 segundos más, o lo que es lo mismo, 1 hora y 20 minutos aproximadamente.

Cuando no existían incidencias, todo el tiempo que se ganaba al aplicar la marcha rápida (M0), se empleaba para corregir el error de horario, sin embargo, al considerar dichas incidencias, los segundos ganados entre estaciones de la marcha rápida también tienen que ser empleados para corregir el retraso producido por los pasajeros al subir y bajar del tren en cada estación.

8.3 CONCLUSIONES

Según los resultados expuestos a lo largo de este capítulo, se puede justificar que se ha obtenido un modelo de regulación automática del tráfico eficaz y funcional. Cuando se producen retrasos en las vías, el regulador es capaz de llevar los errores de horario a cero ejecutando las consignas oportunas y obedeciendo las limitaciones y restricciones de velocidad impuestas por la vía y/o por la marcha que se esté utilizando. De la misma forma, el regulador también emite consigna a los trenes de modificación de marcha para disminuir el tiempo de intervalo entre ellos si éste ha aumentado demasiado debido a incidencias que hayan podido ocurrir en la circulación. Se recuerda que para ello se han tenido que desarrollar los siguientes modelos:

- **Modelo de conducción automática ATO** (*Automatic Train Operation*). Impone la consigna de velocidad que debe de llevar el tren para que acelere o frene. Para ello previamente se ha tenido que desarrollar el modelo que calcule la fuerza que tiene que ejercer el motor cuando el tren acelera y decelera, así como la velocidad, posición y aceleración.
- **Modelo del ATP** (*Automatic Train Protection*). Controla que el tren no supere la velocidad máxima de la vía y evita colisiones entre trenes circulantes por la misma línea.
- **Modelo de cambio de sentido en vía de topología circular**. Permite que el tren cambie de la vía 1 a la vía 2 y viceversa, cuando ha llegado al final de las mismas.
- **Modelo de marchas de regulación**. Implementación de la regulación por velocidad y deriva – remotor, así como sus diferentes marchas (M0, M1, M2, M3), para que el tren pueda hacer frente a los retrasos ocasionados.
- **Modelo de circulación de varios trenes en la misma vía en condiciones de seguridad**.
- **Modelo de regulador automático del tráfico, ATR** (*Automatic Train Regulation*). A través del cálculo de errores de centrado de intervalo y errores de horario, se ejecuta una orden de marcha para devolver al tren a su horario nominal y conseguir

que el tiempo de intervalo entre trenes no sea demasiado elevado. También se modificarán los tiempos de parada en estación cuando esto sea necesario para combatir los horarios.

Por todo lo anterior, se puede concluir que el simulador multi – tren desarrollado modela todos los aspectos necesarios para ser utilizado como banco de pruebas de nuevos algoritmos de regulación de tráfico, por ejemplo, adaptados a las nuevas posibilidades que ofrecen los nuevos sistemas de señalización vía radio, que permiten actualizar las órdenes de conducción en cualquier momento y no solo en las estaciones.

Capítulo 9. BIBLIOGRAFÍA

[1] Alamys. "Metro de Sevilla incorpora la regulación automática de trenes a su sistema integral de gestión del tráfico". Asociación Latinoamericana de Metros y Ssubterráneos, septiembre 2013.

[2] Laquidaín, J. *La Universidad Comillas desarrolla sistemas de control ferroviario utilizados por Metro de Madrid y de Barcelona.* Source: Diálogo Iberoamericano núm 16 / julio-agosto 1998. Pág. 21.

[3] M. Domínguez, A. Fernández-Cardador, A.P. Cucala, P. Lukaszewicz. Optimal design of metro automatic train operation speed profiles for reducing energy consumption. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit. Vol. 225, nº. 5, pp. 463 - 474, Septiembre 2011. [Online: Agosto 2011] JCR JCR: 0.436 Q3 (2011); 2.359 Q2 (2020) - SJR: Q2 (2011); 0.659 Q2 (2020).

[4] A. Fernández-Cardador, A.P. Cucala, B. Vitoriano, F. de Cuadra. Predictive traffic regulation for metro loop lines based on quadratic programming. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit. Vol. 220, nº. 2, pp. 79 - 89, Julio 2006. JCR JCR: 0.333 (2006); 2.359 Q2 (2020) - SJR: 0.659 Q2 (2020).

[5] Araya, S. and Sone, S. Traffic dynamics of automated transit systems with pre-established schedules. IEEE Trans. Syst., Man Cybern., 1984, 14, 677·687.

[6] I. Martínez, B. Vitoriano, A. Fernández & A. P. Cucala. *Statistical dwell time model for metro lines.* Source: WIT Transactions on The Built Environment, Vol 96, 2007 WIT Press.

ANEXO I

En este anexo figurarán las funciones que se han diseñado para el funcionamiento del regulador automático del tráfico.

PROGRAMA PRINCIPAL Y CLASES

PROGRAMA PRINCIPAL *MAIN*

```
%% SIMULADOR MULTI - TREN DE UNA LÍNEA DE METRO PARA ENSAYOS DE REGULACIÓN
AUTOMÁTICA DEL TRÁFICO
% Autor: Álvaro Cidoncha González.
% Directores: Adrián Fernández Rodríguez y Antonio Fernández Cardador.
% Junio 2022

clear all
clc

%Definición de variables
At = 1; %s
k = ; %Parámetro del ATO
kfrenado =; %Parámetro del ATO
decel =; %m/s^2
h = 80; %Parámetro de ajuste

%Se define la variable de la clase
Tren = ClaseTren;
Vector_Trenes(1:15) = Tren;
Linea = ClaseLinea;
Linea2 = ClaseLinea2;

%Se valida el estado inicial del Tren
EstadoInicial(Tren);
velocidad = Tren.v;

%Inicio del movimento
[Tren.parado, Tren.acelerando] = arrancanoarranca(Tren)

%Se inicializa la velocidad y posición del tren
[Vector_Trenes] = InicializaVariosTrenes(Vector_Trenes, Linea, Linea2);

%Movimiento del Tren
if Tren.acelerando == 1
```

```
    [Vector_Trenes, vkmh, posicion] = SimulaTrafico(Vector_Trenes, Linea, Linea2,
At, k, kfrenado, decel, v_regulacion, v_deriva, v_remotor, h);

    %Se representan los datos
    RepresentaTrafico(Vector_Trenes, Linea, Linea2);
else
   disp('Fin del programa');
end

% save Vector_Trenes Vector_Trenes
```

CLASE TREN

```
classdef ClaseTren;

    properties

        longitud
        masa
        a
        b
        c
        fmax
        v
        posicion
        pk_parada %Recoge los puntos kilométricos de las estaciones
        sTren %Variable que indica la posición que tiene delante el tren
        tiempo
        t_total
        parada
        tiempo_parada %Tiempo que el tren va a estar parado en la estación
        tiempo_parada_0 %Tiempo nominal de parada en estación
        flag_parada_extra
        aux_tiempo_parada
        flag_ultima_estacion
        linea_simulador

        %Variables para la representación de gráficos
        vector_velocidad
        vector_posicion
        vector_tiempo
        vector_vmax

        %Variables flag que indican la situación del tren
        parado
        frenando
        acelerando
        derivando

        intervalo
        indice %Variable que indica el número del tren
```

```
        vector_horario %Vector que recoge el horario del tren (salidas de
estación)
        vector_marcha %Vector que recoge los tiempos del tren en la marcha i
        RH_delante
        RH %Valor del retraso horario
        RH_detras
        flag_calculo_RI %Variable que indica cuándo se tiene que calcular el RI
        RI %Valor del retraso de intervalo
        RCI %Retraso de centrado
        correccion %correccion a aplicar
        almacena_RH %Almacena los retrasos
        almacena_pk %Almacena los pk de los retrasos
        almacena_t %Almacena el tiempo de los retrasos


        reg_vel %Datos de regulación por velocidad que sigue el tren
        deriva %Datos de velocidades de deriva que sigue el tren
        remotor %Datos de velocidades de remotor que sigue el tren

        %Variables para representar la velocidad en las vías 1 y 2 a la
        %hora del cálculo del tiempo total 'T'
        vector_t_1
        vector_v_1
        vector_p_1
        vector_t_2
        vector_v_2
        vector_p_2

        flag_fin %flag para terminar la simulacion


    end

    methods

        function tren = ClaseTren()
            tren.longitud = xlsread('Datos simulador', 'Hoja1', 'B3');
            tren.masa = xlsread('Datos simulador', 'Hoja1', 'B4')*1000;
            tren.a = xlsread('Datos simulador', 'Hoja1', 'B5');
            tren.b = xlsread('Datos simulador', 'Hoja1', 'B6');
            tren.c = xlsread('Datos simulador', 'Hoja1', 'B7');
            tren.fmax = [xlsread('Datos simulador', 'Hoja1', 'A14:A28')
xlsread('Datos simulador', 'Hoja1', 'B14:B28')]
            tren.v = 0;
            tren.posicion = 0;
            tren.pk_parada = 0;
            tren.sTren = 0;
            tren.tiempo = 1;
            tren.t_total = 1;
            tren.parada = 0;
            tiempo_parada = 0;
            tren.tiempo_parada_0 = 0;
            flag_parada_extra = 0;
```

```
            aux_tiempo_parada = 1;
            flag_ultima_estacion = 0;
            tren.linea_simulador = [];
            tren.vector_velocidad = [];
            tren.vector_posicion = [];
            tren.vector_tiempo = [];
            tren.vector_vmax = []
            tren.parado = 1;
            tren.frenando = 0;
            tren.acelerando = 0;
            tren.derivando = 0;
            tren.intervalo = 0;
            tren.indice = 0;
            tren.vector_horario = 0;
            tren.vector_marcha = 0;
            tren.RH_delante = 0;
            tren.RH = 0;
            tren.RH_detras = 0;
            tren.flag_calculo_RI = 0;
            tren.RI = 0;
            tren.RCI = 0;
            tren.correccion = 0;
            tren.almacena_RH = [];
            tren.almacena_pk = [];%Almacena los pk de los retrasos
            tren.almacena_t = []; %Almacena el tiempo de los retrasos

            tren.reg_vel = [];
            tren.deriva = [];
            tren.remotor = [];
            vector_t_1 = [];
            vector_v_1 = [];
            vector_p_1 = [];
            vector_t_2 = [];
            vector_v_2 = [];
            vector_p_2 = [];
            tren.flag_fin = 0;
        end

        function EstadoInicial(tren)
            if tren.parado == 1
               disp('El tren está parado');
            else
               disp('El tren está en marcha');
            end

        end

        function [quieto acelerado] = arrancanoarranca(tren)
            if tren.parado == 1
               aux = input('Indique si desea ponerlo en marcha (Si/No): ', 's');
                if aux == 'Si'
                    disp('El tren ha empezado a moverse');
                     quieto = 0;
```

```
                    acelerado = 1;
                else
                    disp('El tren permanece quieto')
                    quieto = 1;
                    acelerado = 0;
                end
            end
        end


        %Función que inicializa varios trenes
        function [Vector_Trenes] = InicializaVariosTrenes(Vector_Trenes, Linea,
Linea2)
            index = 1;
            %Vector que contiene las posiciones iniciales
            posiciones_iniciales = [xlsread('Datos simulador', 'Hoja1',
'B34:B48')];
            via1o2 = [xlsread('Datos simulador', 'Hoja1', 'C34:C48')];

            %Se inicializa la vía en la que está el tren
            for index = 1:1:length(via1o2)
                if via1o2(index) == 1
                    Vector_Trenes(index).linea_simulador = Linea;
                else
                    Vector_Trenes(index).linea_simulador = Linea2;
                end
            end

            %Se inicializan los datos de regulación de marchas con los que
            %empieza el tren
            for index = 1:1:length(posiciones_iniciales)
                Vector_Trenes(index).reg_vel = Linea.marcha1_reg_velocidad;
                Vector_Trenes(index).deriva = Linea.marcha1_deriva;
                Vector_Trenes(index).remotor = Linea.marcha1_remotor;
            end

            %Se inicializan velocidades, posiciones y el índice de la
            %parada
            for index = 1:1:length(posiciones_iniciales)
                Vector_Trenes(index).v = 0;
                Vector_Trenes(index).posicion = posiciones_iniciales(index);
                Vector_Trenes(index).parada =
find(Vector_Trenes(index).linea_simulador.estacion >=
Vector_Trenes(index).posicion, 1);
            end

            %Se inicializa el tiempo de parada de cada tren
            for index = 1:1:length(posiciones_iniciales)
                if index == 1
                    Vector_Trenes(index).tiempo_parada = 16;
                else
                    Vector_Trenes(index).tiempo_parada = 16;
                end
```

```
            end

            %Se inicializa la variable flag que indica si los trenes están
            %quietos o no y la variable auxiliar para el tiempo de parada
            for index = 1:1:length(posiciones_iniciales)
                Vector_Trenes(index).parado = 0;
                Vector_Trenes(index).aux_tiempo_parada = 1;
                Vector_Trenes(index).flag_ultima_estacion = 0;
            end

            %Se inicializa la variable sTren de cada tren
            for index = 1:1:length(posiciones_iniciales)
                if index == 1
                    Vector_Trenes(index).sTren = inf;
                else
                    Vector_Trenes(index).sTren = Vector_Trenes(index-1).posicion;
                end
            end

            %Se inicializa el pk externo (distancia de seguridad: 50 m
            %dedsde la cola del tren).
            for index = 1:1:length(posiciones_iniciales)
                if index == 1
                    Vector_Trenes(index).pk_parada = Vector_Trenes(index).sTren -
Vector_Trenes(index).longitud - 50;
                else
                    Vector_Trenes(index).pk_parada = Vector_Trenes(index).sTren -
50; %50 m de seguridad desde la cola del tren de delante
                end
            end

            %Se inicializa el intervalo
            for index = 1:1:length(posiciones_iniciales)
                if index == 1
                    Vector_Trenes(index).intervalo = 209;
                else
                    Vector_Trenes(index).intervalo = 210;
                end
            end

            %Se inicializa el indice del tren
            for index = 1:1:length(posiciones_iniciales)
                Vector_Trenes(index).indice = index;
            end

            %Se inicializan los horarios de los trenes
            Vector_Trenes(1).vector_horario = [xlsread('Horarios', 'Hoja1',
'B3:B37')];
            Vector_Trenes(2).vector_horario = [xlsread('Horarios', 'Hoja1',
'C3:C37')];
            Vector_Trenes(3).vector_horario = [xlsread('Horarios', 'Hoja1',
'D3:D37')];
```

```
            Vector_Trenes(4).vector_horario = [xlsread('Horarios', 'Hoja1',
'E3:E37')];
            Vector_Trenes(5).vector_horario = [xlsread('Horarios', 'Hoja1',
'F3:F37')];
            Vector_Trenes(6).vector_horario = [xlsread('Horarios', 'Hoja1',
'G3:G37')];
            Vector_Trenes(7).vector_horario = [xlsread('Horarios', 'Hoja1',
'H3:H37')];
            Vector_Trenes(8).vector_horario = [xlsread('Horarios', 'Hoja1',
'I3:I37')];
            Vector_Trenes(9).vector_horario = [xlsread('Horarios', 'Hoja1',
'J3:J37')];
            Vector_Trenes(10).vector_horario = [xlsread('Horarios', 'Hoja1',
'K3:K37')];
            Vector_Trenes(11).vector_horario = [xlsread('Horarios', 'Hoja1',
'L3:L37')];
            Vector_Trenes(12).vector_horario = [xlsread('Horarios', 'Hoja1',
'M3:M37')];
            Vector_Trenes(13).vector_horario = [xlsread('Horarios', 'Hoja1',
'N3:N37')];
            Vector_Trenes(14).vector_horario = [xlsread('Horarios', 'Hoja1',
'O3:O37')];
            Vector_Trenes(15).vector_horario = [xlsread('Horarios', 'Hoja1',
'P3:P37')];
        end



        %Función que contiene los cálculos
        function [Tren, pk_parada, linea_simulador] = MovimientoAcelerando(Tren,
Linea, Linea2, linea_simulador, At, k, kfrenado, decel, tiempo_parada,
v_regulacion, v_deriva, v_remotor, velocidad, posicion, vkmh, kmhvmax,
factor_reg, factor_deriva_remotor, pk_parada, tiempo_pk_parada, tiempo_total, h);

%             %Para provocar retrasos
            if Tren.indice == 2 && Tren.posicion > 2000 && Tren.posicion < 3000
&& Tren.tiempo > 3150 && Tren.tiempo < 4000
                tiempo_parada = 316; %5 minutos de retraso
            end

            %Se indica si el Tren ha llegado a la velocidad de deriva o remotor

            %Cálculo de la velocidad de deriva
            ipos_vderiva = find(v_deriva(:,1) <= posicion);
            kmh_vel_deriva = v_deriva(max(ipos_vderiva),2)*factor_deriva_remotor;
            vel_deriva = kmh_vel_deriva*1000/3600;

            %Cálculo de la velocidad de remotor
            ipos_vremotor = find(v_remotor(:,1) <= posicion);
            kmh_vel_remotor = v_remotor(max(ipos_vremotor),
2)*factor_deriva_remotor;
            vel_remotor = kmh_vel_remotor*1000/3600;
```

```
            if velocidad >= vel_deriva
                Tren.derivando = 1;
                Tren.acelerando = 0;
            elseif velocidad <= vel_remotor
                Tren.derivando = 0;
                Tren.acelerando = 1;
            end


            %Búsqueda de la pendiente
            [pte] = BuscarPendiente(Tren, linea_simulador);

            %Se identifica la fuerza máxima (Fmax)
            if velocidad > 0 && velocidad < 80*1000/3600
                idebajo = find(Tren.fmax(:,1) < vkmh);
                vdebajo = Tren.fmax(max(idebajo));
                iarriba = find(Tren.fmax(:,1) >= vkmh, 1);
                varriba = Tren.fmax(iarriba);
                %Posiciones de fuerza máxima
                fdebajo = Tren.fmax(max(idebajo), 2);
                farriba = Tren.fmax(iarriba, 2);
                %Se calcula la interpolación
                Fmax = fdebajo+((farriba-fdebajo)/(varriba-vdebajo)*(vkmh-
vdebajo));
            elseif velocidad == 0
                Fmax = Tren.fmax(1,2);
            elseif velocidad >= 80*1000/3600
                Fmax = Tren.fmax(15,2);
            end

            %En este paso se va a indicar si se emplea regulación
            %por velocidad o no
            %Se calcula velocidad máxima de la vía
            ipos = find(linea_simulador.vmax(:,1) <= posicion);
            kmhvmax = linea_simulador.vmax(max(ipos), 2); %Queda definida vmax
para la zona en donde está el Tren
            %Se calcula la velocidad de regulación
            ipos_vreg = find(v_regulacion(:,1) <= posicion);
            kmh_vel_regulacion = v_regulacion(max(ipos_vreg), 2);
            %Se decide cuál es la velocidad consigna
            if kmh_vel_regulacion*factor_reg < kmhvmax
               velmax = kmh_vel_regulacion*1000/3600; %m/s
            else
                velmax = kmhvmax*1000/3600; %m/s
            end

            %FUNCIONES PARA FRENADO

            %Se busca la siguiente reduccion de velocidad
            [pos_sig_reduccion, vel_sig_reduccion] =
BuscarSiguienteReduccionVel(Tren, linea_simulador, Tren.posicion);

            Tren.flag_calculo_RI = 0;
```

```
            %Se calcula la velocidad no superable. También se
            %contemplan las estaciones
            if (Tren.posicion > linea_simulador.estacion(Tren.parada) || (Tren.v
< 0.05 && Tren.posicion > linea_simulador.estacion(Tren.parada)-4)) &&
Tren.flag_ultima_estacion == 0
                Tren.parado = 1;
                %Se guarda el tiempo original de parada
                if Tren.aux_tiempo_parada == 1
                    %Se calcula el tiempo de parada según la
                    %distribución lognormal
                    [t_add_logn] = ParadaLognormal(Tren);
                    Tren.tiempo_parada_0 = Tren.tiempo_parada + t_add_logn;
                    tiempo_parada = Tren.tiempo_parada_0;
                    Tren.tiempo_parada = tiempo_parada;
                end
                if Tren.aux_tiempo_parada <= tiempo_parada && Tren.parada ~= 1
                    Tren.v = 0;
                    vkmh = 0;
                    Tren.posicion = linea_simulador.estacion(Tren.parada);
                    posicion = Tren.posicion;

                    if Tren.aux_tiempo_parada == 1
                        %Se calcula la correccion a la llegada a la estación y si
                        %hay que introducir un tiempo de parada extra:
                        pk_horarios = [xlsread('Horarios', 'Hoja1', 'Q3:Q37')];
                        index_RH = find(pk_horarios == Tren.posicion);
                        Tren.RH = (tiempo_total-1+tiempo_parada) -
Tren.vector_horario(index_RH);
                        Tren.RCI = Tren.RH -
1/2*(Tren.RH_detras+Tren.RH_delante);

                        %Cálculo de la corrección
                        [correccion] = CalculaCorreccion(Tren, h);
                        Tren.correccion = correccion;

                        %Función que devuelve el tiempo de parada extra
                        [tiempo_parada_extra, Tren] = TiempoParadaExtra(Tren,
Linea, Linea2);

                        tiempo_parada = Tren.tiempo_parada_0 +
tiempo_parada_extra;
                        Tren.tiempo_parada = tiempo_parada;

                        %Si debido a la probabilidad de la lognormal se
                        %superan los 35 segundos máximos de parada (15 s de
                        %parada + 20 s de parada extra), se satura el
                        %tiempo de parada a 35 s
                        if Tren.tiempo_parada > 35
                            Tren.tiempo_parada = 35;
                        end

                        %Se almacena el delay para representarlo luego
                        if Tren.flag_ultima_estacion == 0
```

```
                        Tren.almacena_RH(end+1) = [Tren.RH];
                        Tren.almacena_pk(end+1)= Tren.posicion;
                        Tren.almacena_t(end+1) = Tren.t_total;
                    end
                end

                Tren.aux_tiempo_parada = Tren.aux_tiempo_parada + 1;
            else

                Tren.parado = 0;

                %Para calcular el horario
                %Tren.vector_horario(end+1) = Tren.tiempo - 1;

                %Para calcular el tiempo de cada marcha
                %Tren.vector_marcha(end+1) = Tren.tiempo - 1;

                %Se calcula el retraso horario (RH)
                pk_horarios = [xlsread('Horarios', 'Hoja1', 'Q3:Q37')];
                index_RH = find(pk_horarios == Tren.posicion);
                Tren.RH = (tiempo_total-1) - Tren.vector_horario(index_RH);
                Tren.flag_calculo_RI = 1;
                %Se calcula el Retraso de intervalo (RH)
                Tren.RI = Tren.RH - Tren.RH_detras;
                %Se calcula el Retraso de centrado (RCI)
                Tren.RCI = Tren.RH - 1/2*(Tren.RH_detras+Tren.RH_delante);

                %Se calcula la corrección
                [correccion] = CalculaCorreccion(Tren, h);
                Tren.correccion = correccion;

                %Se llama a la función de aplicar las marchas para
                %corregir la correccion
                [Tren] = AplicarMarcha(Tren, Linea, Linea2);

                %Se almacena el delay para representarlo luego
                if Tren.posicion ~= Tren.linea_simulador.estacion(1)
                    Tren.almacena_RH(end+1) = Tren.RH;
                    Tren.almacena_pk(end+1)= Tren.posicion;
                    Tren.almacena_t(end+1) = Tren.t_total;
                end

                Tren.aux_tiempo_parada = 1;
                Tren.parada = Tren.parada + 1;
                Tren.flag_parada_extra = 0;

            end

        end


        %Para el caso de las paradas no programadas
        %(arbitrarias)
```

```
            flag_pos_aux = 0;
            if Tren.posicion > pk_parada || (Tren.v < 0.1 && Tren.posicion >
pk_parada - 5)
                Tren.parado = 1;
                if Tren.aux_tiempo_parada <= tiempo_pk_parada
                    Tren.v = 0;
                    vkmh = 0;
                    Tren.posicion = pk_parada;
                    posicion = Tren.posicion;
                    Tren.aux_tiempo_parada = Tren.aux_tiempo_parada + 1;
                else
                    flag_pos_aux = 1;
                    pos_aux = Tren.posicion;
                    Tren.parado = 0;
                    Tren.aux_tiempo_parada = 1;
                    Tren.pk_parada = inf;
                    Tren.v = 0;
                    vkmh = 0;
                    Tren.posicion = pk_parada;
                    posicion = Tren.posicion;
                end

            end

            if Tren.parado == 0
                %Se tienen en cuenta la velocidad no superable por
                %parte de la estación y por parte de las restricciones
                %de las velocidades máximas de las vías
                [vel_no_superable_ms] = VelocidadNoSuperable(Tren,
linea_simulador, pos_sig_reduccion, vel_sig_reduccion, decel, Tren.posicion);
                [vel_no_superable_estacion_ms] =
VelocidadNoSuperableEstacion(Tren, linea_simulador, decel, Tren.posicion,
Tren.parada);
                %En caso de haber una parada arbitraria
                if pk_parada ~= inf
                    [vel_no_superable_arbitraria_ms] =
VelocidadNoSuperableArbitraria(Tren, linea_simulador, decel, Tren.posicion,
pk_parada);
                else
                vel_no_superable_arbitraria_ms = inf;
                end

                %Se escoge la velocidad más baja
                vel_no_superable_1_ms = min(vel_no_superable_ms,
vel_no_superable_estacion_ms);
                vel_no_superable_total_ms = min(vel_no_superable_1_ms,
vel_no_superable_arbitraria_ms);

                %Se comprueba si se supera o no la vmax y se escoge
                %Fmotor
                if vel_no_superable_total_ms <= velmax
```

```
                    [Fmotor_freno] = FuncionFrenado(Tren, linea_simulador,
kfrenado, vel_no_superable_total_ms, At, decel, pte);
                    Fmotor = Fmotor_freno;

                else
                %Cálculo a0 y saturación

                %Para saturar acero: si acero > 1, acersoat = 1. Si acero < -1,
acerosat = -1; de cualquier otra forma, acerosat = acero.
                acero = k*(velmax-velocidad); %En m/s
                acerosat = acero;
                   if acero > 1
                       acerosat = 1;
                   end
                   if Tren.derivando == 1
                       acerosat = min(acerosat, 0);
                   end
                   Fmotor = acerosat*Fmax;
                end

                %Cáluclos tracción o frenado
                Fresis = Tren.a+Tren.b*vkmh+Tren.c*vkmh^2;
                Fpte = Tren.masa*9.81*pte/1000;
                acel = (Fmotor-Fresis-Fpte)/Tren.masa; %en m/s^2
                velocidad = velocidad+acel*At; %en m/s
                posicion = posicion+velocidad*At+(1/2)*acel*At^2;

                if velocidad < 0 && flag_pos_aux == 1
                    velocidad = 0;
                    posicion = pos_aux;
                end

                vkmh = velocidad*3600/1000; %en Km/h
                Tren.posicion = posicion;
                Tren.v = vkmh;

            end %end del if Tren.parado == 0

            %Se guardan los datos en vectores
            Tren.vector_tiempo(Tren.tiempo) = Tren.tiempo;
            Tren.vector_velocidad(Tren.tiempo) = vkmh;

            %If para imponer que la vmax sea la v de regulacion
            if kmh_vel_regulacion*factor_reg < kmhvmax
            Tren.vector_vmax(Tren.tiempo) = kmh_vel_regulacion;
            else
                Tren.vector_vmax(Tren.tiempo) = kmhvmax;
            end
            Tren.vector_posicion(Tren.tiempo) = posicion;

            %Se guardan los vectores de velocidad, posición y tiempo en la clase
Tren
```

```
            Tren.vector_tiempo(1:length(Tren.vector_velocidad)) =
(1:1:length(Tren.vector_velocidad))+(Tren.indice-1)*Tren.intervalo;

            %Se guardan los datos de los vectores de cada vía para la
            %representación de la marcha 1 (cálculo tiempo total 'T')
            if Tren.linea_simulador.estacion(1) == 1111.17 &&
Tren.vector_tiempo(end) ~= 0
                if Tren.tiempo < 3000 %Tiempo para que no se sobrescriba
                    Tren.vector_t_1(end+1) = Tren.vector_tiempo(end);
                    Tren.vector_v_1(end+1) = Tren.v;
                    Tren.vector_p_1(end+1) = Tren.posicion;
                end
            else
                Tren.vector_t_2(end+1) = Tren.vector_tiempo(end);
                Tren.vector_v_2(end+1) = Tren.v;
                Tren.vector_p_2(end+1) = Tren.posicion;
            end

            %Última parada
            if Tren.posicion > linea_simulador.estacion(end-1) && Tren.v < 0.05

                Tren.parado = 1;
                Tren.flag_ultima_estacion = 1;

                if Tren.aux_tiempo_parada <= tiempo_parada
                    Tren.v = 0;
                    vkmh = 0;
                    Tren.posicion = linea_simulador.estacion(Tren.parada);
                    posicion = Tren.posicion;
                    Tren.aux_tiempo_parada = Tren.aux_tiempo_parada + 1;
                    flag_cambio_via = 0;
                else
                    %Se calcula el retraso horario (RH)
                    pk_horarios = [xlsread('Horarios', 'Hoja1', 'Q3:Q37')];
                    index_RH = find(pk_horarios == Tren.posicion);
                    Tren.RH = (tiempo_total-1) - Tren.vector_horario(index_RH);
                    Tren.flag_calculo_RI = 1;
                    %Se calcula el Retraso de intervalo (RH)
                    Tren.RI = Tren.RH - Tren.RH_detras;
                    %Se calcula el Retraso de centrado (RCI)
                    Tren.RCI = Tren.RH - 1/2*(Tren.RH_detras+Tren.RH_delante);

                    %Cálculo de la corrección
                    [correccion] = CalculaCorreccion(Tren, h);
                    Tren.correccion = correccion;

                    %Se aplica la marcha
                    [Tren] = AplicarMarcha(Tren, Linea, Linea2);

                    Tren.almacena_RH(end+1) = Tren.RH;
                    Tren.almacena_pk(end+1)= Tren.posicion;
                    Tren.almacena_t(end+1) = Tren.t_total
```

```
                    Tren.parado = 0;
                    Tren.aux_tiempo_parada = 1;
                    Tren.parada = Tren.parada + 1;
                    flag_cambio_via = 1;
                    Tren.flag_ultima_estacion = 0;

                end


                if flag_cambio_via == 1
                %Cambio de vía
                %Cambio de vía 1 a 2
                    if linea_simulador.estacion(1) == 1111.17
                        linea_simulador = Linea2;

                        Tren.parada = 1;
                        Tren.posicion = linea_simulador.estacion(Tren.parada);
                        Tren.v = 0;

                    else
                    %Cambio de vía 2 a vía 1
                        linea_simulador = Linea;

                        Tren.parada = 1;
                        Tren.posicion = linea_simulador.estacion(Tren.parada);
                        Tren.v = 0;

                    end
                end
            end
            Tren.tiempo = Tren.tiempo+At;


        end


        function RepresentaTrafico(Vector_Trenes, Linea, Linea2)
            %Transformación a pk decrecientes en vía 2
            pkRef = 0;
            for index = 1:1:length(Vector_Trenes)
                pkRef = Linea.estacion(18) + Linea2.estacion(1);
                Vector_Trenes(index).vector_p_2 = pkRef -
Vector_Trenes(index).vector_p_2;
            end


            %Se pintan las velocidades
            figure;
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_tiempo,
Vector_Trenes(index).vector_velocidad);
                grid on;
```

```
                title('Velocidad trenes - t; Marcha 1');
                xlabel('tiempo(s)');
                ylabel('Velocidad(km/h)');
            end
            hold off

            %Se pintan las posiciones
            figure;
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_tiempo,
Vector_Trenes(index).vector_posicion);
                xlim([0 3150])
                grid on;
                title('pk trenes - t; Marcha 1');
                xlabel('tiempo(s)');
                ylabel('pk(m)');
            end
            hold off

            %Se pintan las velocidades frente a las posiciones
            figure;
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_posicion,
Vector_Trenes(index).vector_velocidad);
                grid on;
                title('velocidad trenes - posicion; Marcha 1');
                xlabel('posicion (m)');
                ylabel('velocidad (m/s)');
            end
            hold off

            %Se pinta la velocidad frente a la posición y las posiciones a
            %la vez
            figure;
            subplot(2,1,1)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_posicion,
Vector_Trenes(index).vector_velocidad);
                grid on;
                title('velocidad trenes - posicion; Marcha 1');
                xlabel('posicion (m)');
                ylabel('velocidad (km/h)');
            end
            hold off

            subplot(2,1,2)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_tiempo,
Vector_Trenes(index).vector_posicion);
```

```
                grid on;
                title('posicion trenes - tiempo; Marcha 1');
                xlabel('tiempo(s)');
                ylabel('pk(m)');
            end
            hold off


            %Representación específica para el cálculo del tiempo 'T' (Marcha 1)
            figure;
            subplot(3,2,1)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_t_1,
Vector_Trenes(index).vector_v_1);
                grid on;
                title('velocidad trenes - tiempo; Marcha 1 en vía 1');
                xlabel('tiempo (s)');
                ylabel('velocidad (km/h)');
            end
            hold off

            subplot(3,2,2)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_t_2,
Vector_Trenes(index).vector_v_2);
                grid on;
                title('velocidad trenes - tiempo; Marcha 1 en vía 2');
                xlabel('tiempo (s)');
                ylabel('velocidad (km/h)');
            end
            hold off

            subplot(2,2,1)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_p_1,
Vector_Trenes(index).vector_v_1);
                grid on;
                title('velocidad trenes - posición; Marcha 1 en vía 1');
                xlabel('posición (m)');
                ylabel('velocidad (km/h)');
            end
            hold off

            subplot(2,2,2)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_p_2,
Vector_Trenes(index).vector_v_2);
                grid on;
                title('velocidad trenes - posición; Marcha 1 en vía 2');
```

```
                xlabel('posición (m)');
                ylabel('velocidad (km/h)');
            end
            hold off

            subplot(2,2,3)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_t_1,
Vector_Trenes(index).vector_p_1);
                grid on;
                title('posicion trenes - tiempo; Marcha 1 en vía 1');
                xlabel('tiempo (s)');
                ylabel('posición (m)');
            end
            hold off

            subplot(2,2,4)
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).vector_t_2,
Vector_Trenes(index).vector_p_2);
                grid on;
                title('posicion trenes - tiempo; Marcha 1 en vía 2');
                xlabel('tiempo (s)');
                ylabel('posicion (m)');
            end
            hold off

            %Se representa el retraso frente al tiempo
            figure;
            hold on
            for index = 1:1:length(Vector_Trenes)
                plot(Vector_Trenes(index).almacena_t,
Vector_Trenes(index).almacena_RH, '-s');
                grid on;
                title('Retraso - Tiempo');
                xlabel('Tiempo (s)');
                ylabel('Retraso (s)');
            end
            hold off

            %Se representa el retraso del tren y el de los 5 que van
            %delante
            figure;
            hold on
            plot(Vector_Trenes(2).almacena_t, Vector_Trenes(2).almacena_RH, '-
s');
            plot(Vector_Trenes(1).almacena_t, Vector_Trenes(1).almacena_RH, '-
s');
            plot(Vector_Trenes(15).almacena_t, Vector_Trenes(15).almacena_RH, '-
s');
```

```
            plot(Vector_Trenes(14).almacena_t, Vector_Trenes(14).almacena_RH, '-
s');
            grid on;
            title('Retraso - Tiempo');
            xlabel('Tiempo (s)');
            ylabel('Retraso (s)');
            hold off

        end
    end
end
```

CLASE LÍNEA

```
classdef ClaseLinea

    properties
        vmax %vmax de la vía 1
        pendiente %pendiente de la vía 2
        estacion %pk de las estaciones de la vía 1
        tiempo_parada %Recoge los tiempos de parada de cada estación

        marcha0_reg_velocidad %Datos regulacion por velocidad marcha 0
        marcha0_deriva
        marcha0_remotor

        marcha1_reg_velocidad %Datos regulacion por velocidad marcha 1
        marcha1_deriva
        marcha1_remotor

        marcha2_reg_velocidad %Datos regulacion por velocidad marcha 2
        marcha2_deriva
        marcha2_remotor

        marcha3_reg_velocidad %Datos regulacion por velocidad marcha 3
        marcha3_deriva
        marcha3_remotor

        tiempos_marchas_via1 %Tiempos ganados o perdidos en vía 1
    end

    methods
        function linea = ClaseLinea()
            linea.pendiente = [xlsread('Datos simulador', 'Hoja1', 'L6:L70')
xlsread('Datos simulador', 'Hoja1', 'M6:M70')];
            linea.vmax = [xlsread('Datos simulador', 'Hoja1', 'N6:N71')
xlsread('Datos simulador', 'Hoja1', 'O6:O71')];
            linea.estacion = [xlsread('Datos simulador', 'Hoja1', 'K6:K23')];
            linea.tiempo_parada = [xlsread('Datos simulador', 'Hoja1',
'E34:E51')];
```

```
            linea.marcha0_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'BA5:BA23') xlsread('Datos simulador', 'Hoja1', 'BB5:BB23')];
            linea.marcha0_deriva = [xlsread('Datos simulador', 'Hoja1',
'BC5:BC23') xlsread('Datos simulador', 'Hoja1', 'BD5:BD23')];
            linea.marcha0_remotor = [xlsread('Datos simulador', 'Hoja1',
'BE5:BE23') xlsread('Datos simulador', 'Hoja1', 'BF5:BF23')];

            linea.marcha1_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'AO5:AO23') xlsread('Datos simulador', 'Hoja1', 'AP5:AP23')];
            linea.marcha1_deriva = [xlsread('Datos simulador', 'Hoja1',
'AQ5:AQ23') xlsread('Datos simulador', 'Hoja1', 'AR5:AR23')];
            linea.marcha1_remotor = [xlsread('Datos simulador', 'Hoja1',
'AS5:AS23') xlsread('Datos simulador', 'Hoja1', 'AT5:AT23')];

            linea.marcha2_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'BM5:BM23') xlsread('Datos simulador', 'Hoja1', 'BN5:BN23')];
            linea.marcha2_deriva = [xlsread('Datos simulador', 'Hoja1',
'BO5:BO23') xlsread('Datos simulador', 'Hoja1', 'BP5:BP23')];
            linea.marcha2_remotor = [xlsread('Datos simulador', 'Hoja1',
'BQ5:BQ23') xlsread('Datos simulador', 'Hoja1', 'BR5:BR23')];

            linea.marcha3_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'BY5:BY23') xlsread('Datos simulador', 'Hoja1', 'BZ5:BZ23')];
            linea.marcha3_deriva = [xlsread('Datos simulador', 'Hoja1',
'CA5:CA23') xlsread('Datos simulador', 'Hoja1', 'CB5:CB23')];
            linea.marcha3_remotor = [xlsread('Datos simulador', 'Hoja1',
'CC5:CC23') xlsread('Datos simulador', 'Hoja1', 'CD5:CD23')];

            linea.tiempos_marchas_via1 = [xlsread('Horarios', 'Hoja1', 'Q3:Q21')
xlsread('Horarios', 'Hoja1', 'T3:T21') xlsread('Horarios', 'Hoja1', 'V3:V21')
xlsread('Horarios', 'Hoja1', 'X3:X21')];
        end
    end
end
```

CLASE LÍNEA 2

```
classdef ClaseLinea2

    properties
        vmax %vmax de la vía 2
        pendiente %pendiente de la vía 2
        estacion %pk de las estaciones de la vía 2
        tiempo_parada %Recoge los tiempos de parada de cada estación

        marcha0_reg_velocidad %Datos regulacion por velocidad marcha 0
        marcha0_deriva
        marcha0_remotor

        marcha1_reg_velocidad %Datos regulacion por velocidad marcha 1
        marcha1_deriva
```

```
        marcha1_remotor

        marcha2_reg_velocidad %Datos regulacion por velocidad marcha 2
        marcha2_deriva
        marcha2_remotor

        marcha3_reg_velocidad %Datos regulacion por velocidad marcha 3
        marcha3_deriva
        marcha3_remotor

        tiempos_marchas_via2 %Tiempos ganados o perdidos en vía 2
    end

    methods
        function linea2 = ClaseLinea2()
            linea2.pendiente = [xlsread('Datos simulador', 'Hoja1', 'R6:R70')
xlsread('Datos simulador', 'Hoja1', 'S6:S70')];
            linea2.vmax = [xlsread('Datos simulador', 'Hoja1', 'T6:T67')
xlsread('Datos simulador', 'Hoja1', 'U6:U71')];
            linea2.estacion = [xlsread('Datos simulador', 'Hoja1', 'Q6:Q23')];
            linea2.tiempo_parada = [xlsread('Datos simulador', 'Hoja1',
'E34:E51')];

            linea2.marcha0_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'BG5:BG23') xlsread('Datos simulador', 'Hoja1', 'BH5:BH23')];
            linea2.marcha0_deriva = [xlsread('Datos simulador', 'Hoja1',
'BI5:BI23') xlsread('Datos simulador', 'Hoja1', 'BJ5:BJ23')];
            linea2.marcha0_remotor = [xlsread('Datos simulador', 'Hoja1',
'BK5:BK23') xlsread('Datos simulador', 'Hoja1', 'BL5:BL23')];

            linea2.marcha1_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'AU5:AU23') xlsread('Datos simulador', 'Hoja1', 'AV5:AV23')];
            linea2.marcha1_deriva = [xlsread('Datos simulador', 'Hoja1',
'AW5:AW23') xlsread('Datos simulador', 'Hoja1', 'AX5:AX23')];
            linea2.marcha1_remotor = [xlsread('Datos simulador', 'Hoja1',
'AY5:AY23') xlsread('Datos simulador', 'Hoja1', 'AZ5:AZ23')];

            linea2.marcha2_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'BS5:BS23') xlsread('Datos simulador', 'Hoja1', 'BT5:BT23')];
            linea2.marcha2_deriva = [xlsread('Datos simulador', 'Hoja1',
'BU5:BU23') xlsread('Datos simulador', 'Hoja1', 'BV5:BV23')];
            linea2.marcha2_remotor = [xlsread('Datos simulador', 'Hoja1',
'BW5:BW23') xlsread('Datos simulador', 'Hoja1', 'BX5:BX23')];

            linea2.marcha3_reg_velocidad = [xlsread('Datos simulador', 'Hoja1',
'CE5:CE23') xlsread('Datos simulador', 'Hoja1', 'CF5:CF23')];
            linea2.marcha3_deriva = [xlsread('Datos simulador', 'Hoja1',
'CG5:CG23') xlsread('Datos simulador', 'Hoja1', 'CH5:CH23')];
            linea2.marcha3_remotor = [xlsread('Datos simulador', 'Hoja1',
'CI5:CI23') xlsread('Datos simulador', 'Hoja1', 'CJ5:CJ23')];
```

```
            linea2.tiempos_marchas_via2 = [xlsread('Horarios', 'Hoja1',
'Q21:Q38') xlsread('Horarios', 'Hoja1', 'T21:T38') xlsread('Horarios', 'Hoja1',
'V21:V38') xlsread('Horarios', 'Hoja1', 'X21:X38')];
        end
    end
end
```

FUNCIONES

BÚSQUEDA DE PENDIENTE

```
function [pte] = BuscarPendiente(Tren, Linea)
% El objetivo de la función es obtener la pendiente de la vía en el tramo
% por el que está pasando el tren

aux_pte = find(Linea.pendiente(:,1) <= Tren.posicion);
pte = Linea.pendiente(max(aux_pte),2);

end
```

BÚSQUEDA DE LA SIGUIENTE REDUCCIÓN DE VELOCIDAD

```
function [pos_sig_reduccion vel_sig_reduccion] =
BuscarSiguienteReduccionVel(Tren, Linea, pos_inicial)
%BuscarSiguienteReduccionVel busca el siguiente punto donde la velocidad va
%a ser menor a la actual del tren.
%Devuelve la posición y velocidad de dicho punto

aux = find(Linea.vmax(:,1) <= pos_inicial);
i_pos = max(aux);

%Bucle de búsqueda de la próxima velocidad más pequeña a la actual
a = i_pos+1;
%Variable bandera para salir del bucle
bandera = 0;
%Se busca la longitud del vector para introducir como condición en el bucle
%y que no se pase de la misma
x = length(Linea.vmax(:,2));

%El siguiente bucle devuelve la posición y velocidad del tramo más próximo
%donde se va a reducir la velocidad
while (bandera == 0 && a <= x)
    if Linea.vmax(a,2) < Linea.vmax(i_pos,2)
        vel_sig_reduccion = Linea.vmax(a,2);
        pos_sig_reduccion = Linea.vmax(a,1);
        bandera = 1;
```

```
    else
        vel_sig_reduccion = Linea.vmax(x,2);
        pos_sig_reduccion = Linea.vmax(x,1);
    end
    a = a+1;
end
```

CURVA DE FRENO

```
function [vel_no_superable_ms] = VelocidadNoSuperable(Tren, Linea,
pos_sig_reduccion, vel_sig_reduccion, decel, pos_inicial)
%VelocidadNoSuperable devuelve la velocidad máxima a la que puede ir el
%tren; si esta es superada, es necesario que el tren frene.

g = 9.81; %m/s^2

%Se consideran saltos de espacio Ae = 0.01
Ae = 0.01;

%Se inicializa la posición inicial
espacio = pos_sig_reduccion;
vel_no_superable = vel_sig_reduccion*1000/3600; %m/s

    while (espacio >= pos_inicial)
        %Se encuentra la pendiente del tramo
        idecel = find(Linea.pendiente(:,1) < espacio);
        pendiente = Linea.pendiente(max(idecel),2);

        %Cálculos
        a_pend = g*pendiente/1000;
        a_decel = 0.75 + a_pend;

        vel_no_superable = sqrt(vel_no_superable*vel_no_superable +
2*a_decel*Ae);

        espacio = espacio - Ae;

    end
    vel_no_superable_ms = vel_no_superable;
end
```

FUNCIÓN FRENADO

```
function [Fmotor_freno] = FuncionFrenado(Tren, Linea, kfrenado,
vel_no_superable_ms, At, decel, pte)
%FuncionFrenado Devuelve la velocidad y la posición del tren cuando la
%velocidad de la curva de freno es menor o igual que la vmax. Esta función
%irá dentro de un bucle que irá recalculando los valores de Tren.v y
%Tren.posicion.
```

```
%Cálculos
acero_freno = kfrenado*(vel_no_superable_ms - Tren.v*1000/3600) - decel +
pte*9.81/1000; %EN m/s

%Saturación de a0
acero_freno_sat = acero_freno;
    if acero_freno > 1
        acero_freno_sat = 1;
    end
      %No necesario hasta que no tengamos límite de
      %freno
%     if acero_freno < -1
%         acero_freno_sat = -1;
%     end

%Cálculos
Fmotor_freno = acero_freno_sat*Tren.masa;

end
```

VELOCIDAD NO SUPERABLE POR ESTACIÓN

```
function [vel_no_superable_estacion_ms] = VelocidadNoSuperableEstacion(Tren,
Linea, decel, pos_inicial, parada)
%VelocidadNoSuperable devuelve la velocidad máxima a la que puede ir el
%tren; si esta es superada, es necesario que el tren frene.

g = 9.81; %m/s^2
%Se consideran saltos de espacio Ae = 0.01
Ae = 0.01;

%Se inicializa la posición inicial
espacio = Linea.estacion(parada);
vel_no_superable = 0; %m/s

    while (espacio >= pos_inicial)
        %Encontramos la pendiente del tramo
        idecel = find(Linea.pendiente(:,1) < espacio);
        pendiente = Linea.pendiente(max(idecel),2);

        %Cálculos
        a_pend = g*pendiente/1000;
        a_decel = 0.75 + a_pend;

        vel_no_superable = sqrt(vel_no_superable*vel_no_superable +
2*a_decel*Ae);

        espacio = espacio - Ae;

    end
```

```
    vel_no_superable_estacion_ms = vel_no_superable;
end
```

FUNCIÓN DE SIMULACIÓN DE TRÁFICO

```
function [Vector_Trenes, vkmh, posicion] = SimulaTrafico(Vector_Trenes, Linea,
Linea2, At, k, kfrenado, decel, v_regulacion, v_deriva, v_remotor, h)

    %Inicialización de variables
    kmhvmax = 1;

    %Bucle de simulación

    pk_parada = inf; %inf para desactivar
    tiempo_pk_parada = 0;

    %Variable auxiliar que indica los trenes que están en la vía
    trenes_circulando = 1;

    %Variable auxiliar para compararla con el intervalo
    aux_intervalo = 1;

    %Tiempo total: T = 3150
    while (Vector_Trenes(1).tiempo <= 21000)

        %Otro tren inicia el movimiento si ha pasado un intervalo
        if aux_intervalo == Vector_Trenes(1).intervalo
            trenes_circulando = trenes_circulando + 1;
            aux_intervalo = -1;
        end

       %Si ya están todos los trenes circulando, se impone que no salgan
       %más
       if trenes_circulando == length(Vector_Trenes)
           aux_intervalo = 0;
       end

        for index = 1:1:trenes_circulando
            %Se calcula el tiempo total en cada tren
            Vector_Trenes(index).t_total = Vector_Trenes(1).tiempo;

       %Se actualizan los horarios en función de las vueltas:
           if Vector_Trenes(index).posicion == Linea.estacion(1)
                Vector_Trenes(index).vector_horario =
Vector_Trenes(index).vector_horario + 3150;
           end

            %Inicializa<ación de diversas variables
            Vector_Trenes(index).tiempo = Vector_Trenes(index).tiempo;
            velocidad = Vector_Trenes(index).v*1000/3600;
            posicion = Vector_Trenes(index).posicion;
```

```
            vkmh = Vector_Trenes(index).v;

            if  Vector_Trenes(index).flag_parada_extra == 0
                if Vector_Trenes(index).aux_tiempo_parada == 1
                    Vector_Trenes(index).tiempo_parada =
Vector_Trenes(index).linea_simulador.tiempo_parada(Vector_Trenes(index).parada);
                end
            end

            v_regulacion = Vector_Trenes(index).reg_vel;
            v_deriva = Vector_Trenes(index).deriva;
            v_remotor = Vector_Trenes(index).remotor;

            %Cálculo del factor de regulación (si se va a usar reg
            %velocidad o deriva-remotor)
            %Cálculo factor regvel
            ifac_regvel = find(v_regulacion(:,1) <= posicion);
            fac_regvel = v_regulacion(max(ifac_regvel), 2);

            %Cálculo factor deriva-remotor
            ifac_der = find(v_deriva(:,1) <= posicion);
            fac_der = v_deriva(max(ifac_der), 2);

            %Elección del factor
            if fac_regvel == 1 && fac_der == 1
                factor_reg = 100;
                factor_deriva_remotor = 100;
            elseif fac_regvel ~= 1 && fac_der == 1
                factor_reg = 1;
                factor_deriva_remotor = 100;
            elseif fac_regvel == 1 && fac_der ~= 1
                factor_reg = 100;
                factor_deriva_remotor = 1;
            end


            %LLAMADA A LA FUNCIÓN
            [Tren, pk_parada, linea_simulador] =
MovimientoAcelerando(Vector_Trenes(index), Linea, Linea2,
Vector_Trenes(index).linea_simulador, At, k, kfrenado, decel,
Vector_Trenes(index).tiempo_parada, v_regulacion, v_deriva, v_remotor, velocidad,
posicion, vkmh, kmhvmax, factor_reg, factor_deriva_remotor,
Vector_Trenes(index).pk_parada, tiempo_pk_parada, Vector_Trenes(1).tiempo, h);
            Vector_Trenes(index) = Tren;
            Vector_Trenes(index).linea_simulador = linea_simulador;

            %Se rellenan las variables RH_delante y RH_detras
            if index == 1
                Vector_Trenes(index).RH_delante = Vector_Trenes(15).RH;
                Vector_Trenes(index).RH_detras = Vector_Trenes(index+1).RH;
            elseif index ~= 1 && index ~= 15
                Vector_Trenes(index).RH_delante = Vector_Trenes(index-1).RH;
                Vector_Trenes(index).RH_detras = Vector_Trenes(index+1).RH;
```

```
            elseif index == 15
                Vector_Trenes(index).RH_delante = Vector_Trenes(index-1).RH;
                Vector_Trenes(index).RH_detras = Vector_Trenes(1).RH;
            end


            %Se actualiza el valor de la variable sTren y por tanto el de
            %pk_parada para todos los trenes
            if index == 1 && trenes_circulando == length(Vector_Trenes)
                Vector_Trenes(index).sTren = Vector_Trenes(15).posicion;
                Vector_Trenes(index).pk_parada = Vector_Trenes(index).sTren -
Vector_Trenes(index).longitud - 50;

                %Se anula la parada por proximidiad si el tren de delante
                %ha cambiado de vía y el de detrás no ya que sí o sí
                %el de detras va a tener la última estación en el medio
                %antes que el otro tren
                if Vector_Trenes(index).pk_parada < Vector_Trenes(index).posicion
                    Vector_Trenes(index).pk_parada = inf;
                end
            end

            if index >= 2
                Vector_Trenes(index).sTren = Vector_Trenes(index-1).posicion;
                Vector_Trenes(index).pk_parada = Vector_Trenes(index).sTren -
Vector_Trenes(index).longitud - 50;
                %Se anula la parada por proximidiad si el tren de delante
                %ha cambiado de vía y el de detrás no ya que sí o sí
                %el de detras va a tener la última estación en el medio
                %antes que el otro tren
                if Vector_Trenes(index).pk_parada < Vector_Trenes(index).posicion
                    Vector_Trenes(index).pk_parada = inf;
                end
            end
        end
    aux_intervalo = aux_intervalo + 1;
end
```

CURVA DE FRENO DEBIDO A TRENES PREVIOS

```
function [vel_no_superable_arbitraria_ms] = VelocidadNoSuperableArbitraria(Tren,
Linea, decel, pos_inicial, pk_parada)
%VelocidadNoSuperable devuelve la velocidad máxima a la que puede ir el
%tren; si esta es superada, es necesario que el tren frene.

g = 9.81; %m/s^2
%Se consideran saltos de espacio Ae = 0.01
Ae = 0.01;

%Se inicializa la posición inicial
espacio_a = pk_parada;
```

```
vel_no_superable = 0; %m/s

    while (espacio_a >= pos_inicial)
        %Se encuentra la pendiente del tramo
        idecel = find(Linea.pendiente(:,1) < espacio_a);
        pendiente = Linea.pendiente(max(idecel),2);

        %Cálculos
        a_pend = g*pendiente/1000;
        a_decel = 0.75 + a_pend;

        vel_no_superable = sqrt(vel_no_superable*vel_no_superable +
2*a_decel*Ae);

        espacio_a = espacio_a - Ae;

    end
    vel_no_superable_arbitraria_ms = vel_no_superable;
end
```

FUNCIÓN PARA EL CÁLCULO DE LA CORRECCIÓN

```
function [correccion] = CalculaCorreccion(Tren, h)

correccion = max(-Tren.RH, -Tren.RCI - h/2);

end
```

APLICACIÓN DE MARCHA

```
function [Tren] = AplicarMarcha(Tren, Linea, Linea2)

    %Se mira en la vía en la que está el tren
    if Tren.linea_simulador.estacion(1) == 1111.17
        vector_marchas = Linea.tiempos_marchas_via1;
    else
        vector_marchas = Linea2.tiempos_marchas_via2;
    end

    %Se busca los segundos que se ganan o se pierden para la posición en la
    %que está el tren
    auxiliar = find(vector_marchas(:,1) > Tren.posicion, 1);

    %Si está en la última vía
    if Tren.posicion == Tren.linea_simulador.estacion(end)
        auxiliar = find(vector_marchas(:,1) == Tren.posicion, 1);
        auxiliar = auxiliar +1;
    end
```

```
    dif_M2 = vector_marchas(auxiliar,3); %Segundos perdidos en marcha 2
    dif_M3 = vector_marchas(auxiliar,4); %Segundos perdidos eb marcha 3

    %Estando en la ultima estación y en la linea 1, se le aplique las
    %marchas de la linea 2 y viceversa
    if Tren.posicion == Tren.linea_simulador.estacion(end) &&
Tren.linea_simulador.estacion(1) == 1111.17
        Tren.linea_simulador = Linea2;
    elseif Tren.posicion == Tren.linea_simulador.estacion(end) &&
Tren.linea_simulador.estacion(1) == 285.72
        Tren.linea_simulador = Linea;
    end

    % Se ejecuta la marcha
    if Tren.correccion < 0
        Tren.reg_vel = Tren.linea_simulador.marcha0_reg_velocidad;
        Tren.deriva = Tren.linea_simulador.marcha0_deriva;
        Tren.remotor = Tren.linea_simulador.marcha0_remotor;

    elseif Tren.correccion == 0
        Tren.reg_vel = Tren.linea_simulador.marcha1_reg_velocidad;
        Tren.deriva = Tren.linea_simulador.marcha1_deriva;
        Tren.remotor = Tren.linea_simulador.marcha1_remotor;

    end


    if Tren.correccion > 0

        if Tren.correccion < dif_M2
            Tren.reg_vel = Tren.linea_simulador.marcha1_reg_velocidad;
            Tren.deriva = Tren.linea_simulador.marcha1_deriva;
            Tren.remotor = Tren.linea_simulador.marcha1_remotor;

        elseif Tren.correccion > dif_M2 && Tren.correccion < dif_M3
            Tren.reg_vel = Tren.linea_simulador.marcha2_reg_velocidad;
            Tren.deriva = Tren.linea_simulador.marcha2_deriva;
            Tren.remotor = Tren.linea_simulador.marcha2_remotor;

        elseif Tren.correccion == dif_M2
            Tren.reg_vel = Tren.linea_simulador.marcha2_reg_velocidad;
            Tren.deriva = Tren.linea_simulador.marcha2_deriva;
            Tren.remotor = Tren.linea_simulador.marcha2_remotor;

        elseif Tren.correccion == dif_M3
            Tren.reg_vel = Tren.linea_simulador.marcha3_reg_velocidad;
            Tren.deriva = Tren.linea_simulador.marcha3_deriva;
            Tren.remotor = Tren.linea_simulador.marcha3_remotor;

        elseif Tren.correccion > dif_M3
            Tren.reg_vel = Tren.linea_simulador.marcha3_reg_velocidad;
            Tren.deriva = Tren.linea_simulador.marcha3_deriva;
```

```
            Tren.remotor = Tren.linea_simulador.marcha3_remotor;
        end

    end

end
```

TIEMPO DE PARADA EXTRA

```
function [tiempo_parada_extra, Tren] = TiempoParadaExtra(Tren,  Linea, Linea2)
    %Se mira en la vía en la que está el tren
    if Tren.linea_simulador.estacion(1) == 1111.17
        vector_marchas = Linea.tiempos_marchas_via1;
    else
        vector_marchas = Linea2.tiempos_marchas_via2;
    end

    %Se busca los segundos que se ganan o se pierden para la posición en la
    %que está el tren
    auxiliar = find(vector_marchas(:,1) > Tren.posicion, 1);

    %Si está en la última vía
    if Tren.posicion == Tren.linea_simulador.estacion(end)
        auxiliar = find(vector_marchas(:,1) == Tren.posicion, 1);
        auxiliar = auxiliar +1;
    end

    dif_M2 = vector_marchas(auxiliar,3); %Segundos perdidos en marcha 2
    dif_M3 = vector_marchas(auxiliar,4); %Segundos perdidos en marcha 3

    if Tren.correccion > dif_M3
        Tren.flag_parada_extra = 1;
        tiempo_parada_extra = Tren.correccion - dif_M3;
        %Se satura el tiempo de parada extra
        if tiempo_parada_extra > 20
            tiempo_parada_extra = 20;
        end
    else
        tiempo_parada_extra = 0;
    end
end
```

DISTRIBUCIÓN LOGNORMAL

```
function [t_add_logn] = ParadaLognormal(Tren)

%Se generan los números aleatroios de distribución lognormal de media 15 s
%y desviación estandar de 3 s
```

```
if Tren.parada == 1 || Tren.parada == 18
    numlog = lognrnd(22,3);
else
    numlog = lognrnd(15,3);
end

num = log(numlog);

%Se trunca al valor inmediatamente superior
tiempo_parada_logn = fix(num+2);

%Se satura en caso de que quede un tiempo menor al de parada
if tiempo_parada_logn < 23 && (Tren.parada == 1 || Tren.parada == 18)
    tiempo_parada_logn = Tren.tiempo_parada;
elseif (Tren.parada > 1 && Tren.parada < 18) && tiempo_parada_logn < 16
    tiempo_parada_logn = Tren.tiempo_parada;
end

t_add_logn = tiempo_parada_logn - Tren.tiempo_parada;
```