Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO FIN DE GRADO

Herramienta para la simulación del sistema de control de tráfico aéreo

Davinia González Morello

Tutor: Pedro Yuste Pérez; co-tutor: Joan Vilà Carbó

Grado en Ingeniería Aeroespacial

Julio 2017

ÍNDICE

Capítulo 1

INTRODUCCIÓN AL PROYECTO

1.1. Motivación.

El presente proyecto tiene como objetivo el desarrollo de un sistema para la simulación del control de tráfico aéreo. A través de este sistema y el programa EuroScope, se pretende crear un laboratorio virtual de control de tráfico aéreo, que incluya puestos de ATCs o controladores aéreos, y puestos de pseudopilotos. El lenguaje de programación elegido para el desarrollo de este proyecto es Java, ya que es portable y puede ser ejecutado indistintamente en sistemas operativos Linux, Mac, Solaris y Windows.

Desde el punto de vista docente, este sistema podrá ser utilizado en prácticas del Grado en Ingeniería Aeroespacial, en la rama de Aeronavegación. Parte del alumnado desarrollará el papel de controladores aéreos en los puestos de ATC, equipados con el programa EuroScope, y estarán encargados de prevenir las colisiones entre aeronaves y acelerar y mantener ordenadamente el flujo de las aeronaves. Para proporcionar servicio de control aéreo en cada fase de vuelo, se abrirán varias dependencias de control en función del grado de experiencia alcanzado por los alumnos. El resto del alumnado actuará como pilotos de aviación, en los puestos de pseudpiloto, desarrollados e implementados en este Trabajo Fin de Grado. Emplear simuladores de vuelo como X-Plane requiere de una mayor experiencia y aprendizaje previo, lo que causa que haya un menor número de pilotos disponibles para realizar una sesión de simulación.

Desde la óptica de la investigación, servirá para crear tráficos y simular sus trayectorias basadas en modelos matemáticos de EUROCONTROL, como herramienta para la construcción y validación de procedimientos de vuelo, y para la detección y el estudio de posibles conflictos entre tráficos en las diferentes fases de un vuelo y en las proximidades de los aeródromos.

1.2. Antecedentes.

En este momento, en el laboratorio Pedro Duque, en cual se realizan prácticas de la especialidad de Aeronavegación, del Grado en Ingeniería Aeroespacial, se encuentra el programa EuroScope integrado con el simulador de vuelo X-Plane. Este simulador cuenta con modelos dinámicos de vuelo muy precisos, pero tiene la desventaja de ser una herramienta compleja, que requiere de mucho tiempo para aprender a usarla correctamente.

Por otro lado, el sistema de pseudopiloto desarrollado en este proyecto no exige mucha experiencia por parte del usuario. Dispone de una interfaz gráfica sencilla, estructurada de manera similar a los instrumentos de vuelo de una aeronave real. Asimismo, cuenta con un mapa en la parte superior de la ventana, que permite mejorar la conciencia situacional del pseudopiloto, mediante la integración de datos de navegación y de la ruta elegida en el plan de vuelo que deberá seguir la aeronave para llegar al aeródromo de destino. Desde la misma instancia del programa pueden lanzarse varios tráficos simultáneamente. Como parte de este proyecto se ha realizado también un manual de usuario de este programa.

Hasta la fecha, se han realizado Trabajos Fin de Grado que han empleado las ecuaciones de BADA de EUROCONTROL, sin embargo, este proyecto se ha desarrollado desde cero, ya que su implementación se ha llevado a cabo en el lenguaje de programación Java, siendo lo más fiel posible a la Programación puramente Orientada a Objetos.

1.3. Resumen.

Para el desarrollo de este proyecto, en primer lugar, se implementa el simulador de vuelo basado en el manual de usuario de la "Base of Aircraft Data" (BADA) Revisión 3.9 de EUROCONTROL. De este documento, se obtienen todas las ecuaciones a implementar, y de los archivos de BADA para cada tipo de aeronave, se obtienen todos los valores de los coeficientes y parámetros incluidos en dichas ecuaciones. Para esto último, es necesario que el programa tenga acceso a los archivos que componen la base de datos BADA.

Una vez implementado el simulador de vuelo, se estudia cómo realizar la conexión con el servidor del programa EuroScope, que como entorno de control ATC, deberá permitir la visualización de los tráficos creados a través del simulador. Para ello, se estudia el protocolo de conexión, cómo se debe realizar el intercambio de mensajes y cuál debe ser la estructura, el contenido y el orden de éstos.

A continuación, se integra el envío de mensajes a EuroScope con el simulador de vuelo desarrollado en Java. La comunicación se realiza mediante la apertura de sockets de conexión TCP entre el programa del simulador y el servidor de EuroScope.

Se desarrolla una interfaz de usuario sencilla que permita tanto la conexión de tráficos y el procesamiento de sus planes de vuelo, como interactuar con el simulador y sus modos de vuelo.

Se añade un complemento al programa del simulador, para permitir que EuroScope pueda recibir tráficos también a través de simulaciones con el simulador de vuelo X-Plane. De esta manera, a la hora de realizar sesiones de simulación con puestos ATC y de pseudopiloto, el usuario que tenga previos conocimientos sobre simuladores de vuelo, podrá participar en la sesión a través de X-Plane.

Finalmente, se desarrolla un caso de ejemplo de utilización del programa como herramienta de investigación, el cual consiste en la validación de un cambio de configuración de las pistas activas, debido a un empeoramiento de las condiciones meteorológicas, en el Aeropuerto de Palma de Mallorca.

Asimismo, se adjunta como anexo el manual de usuario del programa.

Capítulo 2

SIMULADOR DE VUELO BASADO EN BADA

El simulador de vuelo se ha implementado acorde al manual de usuario de la "Base of Aircraft Data" (BADA) Revisión 3.9 de EUROCONTROL. Esta base de datos consiste en un conjunto de ficheros ASCII con los coeficientes de rendimiento y funcionamiento para 338 tipos diferentes de aeronaves. Los coeficientes incluyen los utilizados para calcular el empuje, la resistencia, el gasto de combustible y para especificar las velocidades nominales de crucero, ascenso y descenso.

El manual del usuario proporciona las definiciones de cada uno de los coeficientes y las fórmulas para la obtención de las diferentes velocidades. La última revisión de BADA, la denominada 3.13, fue publicada por el EUROCONTROL Experimental Centre en 2015. Hasta la versión previa solo ha habido cambios en el remodelado parcial o total de algunas aeronaves y la incorporación de nuevos modelos y de nuevas aeronaves al documento de sinónimos, por lo que la base del funcionamiento del simulador no se verá afectada si en algún momento se actualiza la versión de BADA.

En esta sección explicaremos cómo se ha implementado el simulador para cada fase de vuelo, indicando qué ecuaciones y coeficientes se han empleado para cada una de ellas.

2.1. Modelo de la energía de BADA.

El modelo de la energía total de BADA equipara la tasa de trabajo producido por las fuerzas que actúan en el avión a la tasa de incremento de las energías potencial y cinética. Este modelo se presenta en forma de una sola ecuación, que es la siguiente:

$$(Thr - D) \cdot V_{TAS} = mg_0 \frac{dh}{dt} + mV_{TAS} \frac{dV_{TAS}}{dt}$$

Ecuación 2.1. Modelo de la energía BADA.

Los símbolos empleados se definen a continuación, con sus respectivas unidades:

Thr	–	empuje que actúa paralelo al vector velocidad de la aeronave	[N]
D	–	resistencia aerodinámica	[N]
m	–	masa de la aeronave	[kg]
h	–	altitud geodésica	[m]
g_0	–	aceleración gravitacional	[m/s^2]
V_{TAS}	–	velocidad verdadera	[m/s]
$\frac{d}{dt}$	–	derivada del tiempo	[s^{-1}]

En esta ecuación tenemos tres variables, que son el empuje, la velocidad verdadera y la velocidad vertical. Las variables de entrada o control son dos: el *throttle* y el timón de profundidad.

A través del ajuste de la palanca de gases, en mecánica de vuelo se denomina δp, el piloto modifica el empuje, siendo éste máximo si δp=1 y mínimo si δp=0. Lo que en verdad se hace al cambiar de posición la palanca de gases, es modificar el gasto de combustible que le llega al motor. Así, el sistema regulador, con las condiciones de presión, temperatura, velocidad, número de Mach, etc., modificará las rpm del motor y, asimismo, el empuje.

Por otro lado, el piloto modificará el ángulo de ataque del avión cambiando la incidencia del empenaje horizontal completo o solo de una parte de éste, el timón de profundidad. En ambos casos esta deflexión se denomina δE. Este cambio afectará a la sustentación y al momento de cabeceo o *Pitch* del avión.

Volviendo a la ecuación 2.1, con dos variables controladas, nos queda una variable incógnita a despejar. Si realizamos las tres posibles combinaciones de las variables de entradas, tenemos:

- Velocidad verdadera y throttle controlados. Se emplea la ecuación 2.1 para calcular la tasa de ascenso o descenso o ROCD (del inglés, rate of climb or descent).
- Velocidad vertical y throttle controlados. La ecuación 2.1 se usa para despejar la velocidad verdadera de la aeronave.
- Velocidades verdadera y vertical controladas. Se calcula el empuje necesario para mantener las velocidades, mediante la ecuación 2.1.

Para decidir cuál es la combinación de las anteriores que debemos emplear, veamos el siguiente esquema del perfil vertical de una aeronave.

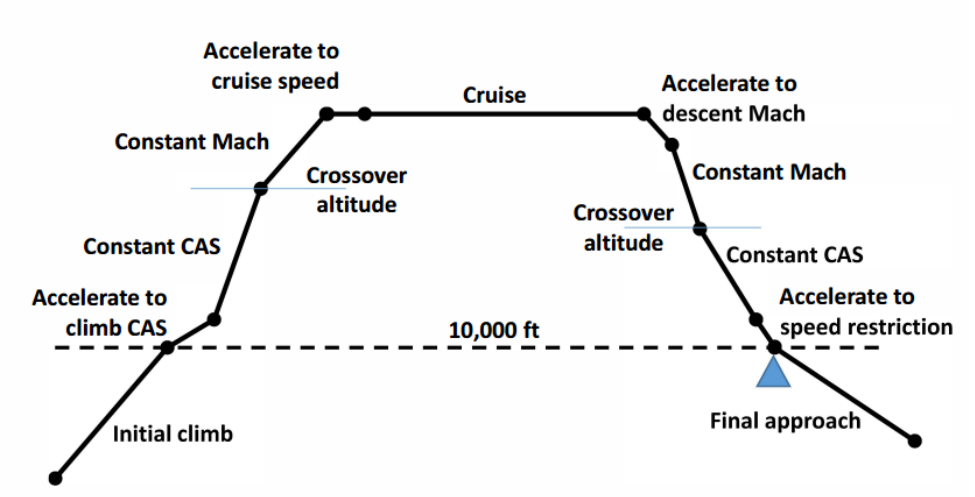


Figura 2.1. Perfil vertical conceptual de una aeronave.

Las fases principales que volará la aeronave son: ascenso, crucero y descenso. En la fase de ascenso, la velocidad verdadera, ya sea la CAS o el Mach, permanecen constantes (salvo en los periodos de aceleración), así como el empuje, el cual es fijo, entonces la combinación de variables que usaremos será la primera propuesta. Durante la fase de crucero, la velocidad verdadera es constante y la velocidad vertical es cero, por lo que podemos despejar el empuje de la ecuación. Por último, en la fase de descenso, sucede igual que en la de ascenso, la velocidad verdadera y el empuje (IDLE) permanecen constante, por lo que podemos conocer la tasa de

descenso, en el tipo de descenso escalonado. En un descenso continuo (DCM) quedarán fijadas la velocidad verdadera y la pendiente, definida esta última por el índice de coste de la operación.

Cuando fijamos la velocidad verdadera, lo hacemos a través del *Airline Procedure Model* de BADA, que determina la velocidad CAS o el Mach, en función de las velocidades definidas en el fichero Airline Procedure File, las velocidades de entrada en pérdida para el despegue y el aterrizaje, unos coeficientes globales para todas las aeronaves contenidos en el archivo BADA.GPF, y la altitud a la que se encuentre la aeronave.

A continuación, veremos detalladamente cómo establecemos las variables de entrada y cómo resolvemos la ecuación de la energía en cada fase.

2.2. Ascenso.

Durante el ascenso, la aeronave gana altura, es decir, aumenta su energía potencial. Desde el punto de qué sucede con la energía cinética, tenemos tres clases de ascensos: la energía cinética se transforma en potencial, esto ocurre en las maniobras acrobáticas; la energía cinética se mantiene, esto quiere decir que la velocidad verdadera es constante; y, por último, la energía cinética aumenta, es decir, se produce una aceleración a lo largo de la trayectoria.

Como se ve en la figura 1, esta fase consta de tres segmentos. Hasta los 10.000 ft, la CAS no puede exceder de 250 kt en la mayoría de los aeródromos del mundo. En el segmento siguiente, se acelera hasta alcanzar la CAS que mantendremos constante. Este segmento y el último están delimitados por la *Crossover altitude*, que es la altitud en la velocidad verdadera para la CAS del segundo segmento se iguala con la velocidad verdadera del Mach a la que la aeronave desea volar. Si siguiésemos volando a CAS constante, la TAS seguiría aumentando, y si a esto le sumamos el hecho de que bajo la tropopausa la velocidad del sonido disminuye, el Mach no dejaría de aumentar y se producirían ondas de choque indeseadas. Por ello, a partir de esta altitud continuamos ascendiendo a Mach constante. La velocidad verdadera disminuirá hasta alcanzar la tropopausa, donde la temperatura se mantiene constante, y, entonces, la velocidad del sonido también.

Asimismo, BADA define un plan o programa para la velocidad CAS de ascenso en función de la altitud en su *Airline Procedure Model.* Cada uno de los tres segmentos tiene asociada una velocidad característica, definida en el Airline Procedure File o fichero APF para cada tipo de aeronave y un rango de masa concreto, pudiendo ser éste último bajo (LO), medio (AV) y alto (HI). En este proyecto, hemos considerado únicamente rangos medios. A su vez, el primer segmento o ascenso inicial, considera incrementos de velocidad desde el despegue hasta los 6.000 ft de altitud. Para aviones de reacción, el plan de velocidades CAS es el siguiente:

de 0 a 1.499 ft	$C_{Vmin} \cdot (V_{stall})_{TO} + Vd_{CL,1}$
de 1.500 a 2.999 ft	$C_{Vmin} \cdot (V_{stall})_{TO} + Vd_{CL,2}$
de 3.000 a 3.999 ft	$C_{Vmin} \cdot (V_{stall})_{TO} + Vd_{CL,3}$
de 4.000 a 4.999 ft	$C_{Vmin} \cdot (V_{stall})_{TO} + Vd_{CL,4}$
de 5.000 a 5.999 ft	$C_{Vmin} \cdot (V_{stall})_{TO} + Vd_{CL,5}$
de 6.000 a 9.999 ft	$\min(V_{cl,1}, 250\ kt)$

de 10.000 ft a *crossover altitude*	$V_{cl,2}$
por encima de *crossover altitude*	Ma_{cl}

Ecuaciones 2.2. Velocidades BADA ascenso

La simbología utilizada en las ecuaciones 2.2 se describe a continuación:

$V_{cl,1}$	–	Velocidad CAS estándar del primer segmento de ascenso	[kt]
$V_{cl,2}$	–	Velocidad CAS estándar del segundo segmento de ascenso	[kt]
Ma_{cl}	–	Número Mach para el tercer y último tramo de ascenso	[-]
C_{Vmin}	–	Coeficiente de velocidad mínima	[-]
$(V_{stall})_{TO}$	–	Velocidad de entrada en pérdida durante el despegue	[kt]
$Vd_{CL,i}$	–	Incremento de velocidad de ascenso	[kt]

Las dos primeras velocidades y el Mach se obtienen del Airline Procedure File, la velocidad de entrada en pérdida, del Operations Performance File, y el coeficiente y el incremento de velocidad, del Global Parameter File.

Para acabar de definir la velocidad que estableceremos para cada altitud, solo queda por determinar la *crossover altitude*, denominada por BADA altitud de transición, que resolveremos con la siguiente fórmula:

$$H_{p,trans} = \left(\frac{1000}{0{,}3048 \cdot 6{,}5}\right) \cdot [T_0 \cdot (1 - \theta_{trans})]$$

Ecuación 2.3. Altitud de transición de BADA.

Siendo θ_{trans} el ratio de temperatura a la altitud de transición. El resultado está expresado en pies.

Una vez que la CAS es fijada, se convierte a TAS, usando las fórmulas del modelo atmosférico del manual de usuario de BADA. El siguiente paso es establecer el empuje, que para esta fase utilizaremos el empuje máximo de ascenso. En el manual de BADA, se explica cómo obtener la potencia de ascenso reducida, la cual no se ha implementado en este proyecto.

El empuje máximo de ascenso para aviones a reacción queda definido mediante la siguiente ecuación:

$$(Thr_{max\ climb}) = C_{Tc,1} \times \left(1 - \frac{H_p}{C_{Tc,2}} + C_{Tc,3} + {H_p}^2\right)$$

Ecuación 2.4. Máximo empuje en ascenso.

Donde $C_{Tc,1}$, $C_{Tc,2}$ y $C_{Tc,3}$ son los parámetros adimensionales del empuje contenidos en el archivo OPF, y H_p, la altitud de presión en pies.

Con el empuje y la velocidad TAS fijados, emplearemos la ecuación 2.1 para calcular la velocidad vertical de la aeronave. La despejaremos de la siguiente manera:

$$VS = \left[\frac{(Thr - D)V_{TAS}}{mg_0}\right] \cdot f\{M\}$$

Ecuación 2.5. Velocidad vertical para el ascenso

Se introduce una variable nueva, el factor de la energía o *energy share factor*, que especifica cuanta energía disponible es asignada a ascender en oposición a acelerar, mientras se sigue un determinado perfil de velocidades durante el ascenso. Según la aeronave ascienda por arriba o por debajo de la tropopausa, y se mantenga constante CAS o Mach, BADA propone cuatro maneras de calcular este factor.

Para obtener la velocidad vertical o tasa de ascenso, de los parámetros de la ecuación 2.5, nos quedaría por definir la masa y la resistencia aerodinámica. La masa de la aeronave que utilizará nuestro programa será la masa de introducida por el usuario en el plan de vuelo. Si este valor es distinto a la masa de referencia indicada en el archivo OPF del tipo de la aeronave, las velocidades operacionales serán modificadas según la fórmula: $V = Vref \cdot \sqrt{\frac{m}{mref}}$.

Para el cálculo de la resistencia aerodinámica, es necesario obtener primero el valor de los coeficientes de sustentación y de resistencia. El coeficiente de sustentación lo determina BADA asumiendo que la pendiente es nula, es decir, en condiciones de equilibrio vertical, igualando el peso a la sustentación. Sin embargo, incluye una corrección para el ángulo de alabeo.

$$CL = \frac{2 \cdot m \cdot g_0}{\rho \cdot {V_{TAS}}^2 \cdot S \cdot cos\phi}$$

Ecuación 2.6. Coeficiente de sustentación.

La superficie de la aeronave se obtiene del fichero OPF, y la densidad, del modelo atmosférico de BADA en función de la altitud de presión.

El coeficiente de resistencia BADA utiliza una versión simplificada de la polar parabólica de coeficientes constantes, es decir, los coeficientes CD_0 y K o CD_2 solo varían según la configuración del avión, y no con el número de Mach o con número de Reynolds. Decimos que es "simplificada" ya que no contiene el término $CL_{minDrag}$, ya que el valor de CD mínimo no se obtiene con un CL nulo. La ecuación queda de la siguiente manera:

$$CD = CD_0 + CD_2 \times CL^2$$

Ecuación 2.7. Coeficiente de resistencia aerodinámica.

Las configuraciones son cinco, despegue (TO), ascenso inicial (IC), crucero (CR), aproximación (AP) y aterrizaje (LD). Sus límites se explican en el apartado 2.7. Configuraciones de la aeronave. BADA proporciona unos valores de CD_0 y CD_2 para cada una de éstas configuraciones en el archivo OPF. Sin embargo, para calcular el coeficiente de resistencia, para las tres primeras configuraciones utiliza los parámetros de crucero.

La resistencia aerodinámica se calcula de la manera estándar:

$$D = \frac{CD \cdot \rho \cdot {V_{TAS}}^2 \cdot S}{2}$$

Ecuación 2.8. Resistencia aerodinámica.

Con todos las variables de entrada calculadas para la ecuación 2.5, podemos despejar la velocidad vertical con la que ascenderá la aeronave.

Ambas velocidades, TAS y VS son dependientes de la altitud, entre otros factores, pero cuando se alcanza una determinada altitud, el cambio en las velocidades no se produce inmediatamente, sino que se produce una aceleración hasta que se alcanzan éstas. La forma de acelerar se encuentra explicada en el apartado 2.6. Aceleraciones.

2.2.1 Ascenso con SID.

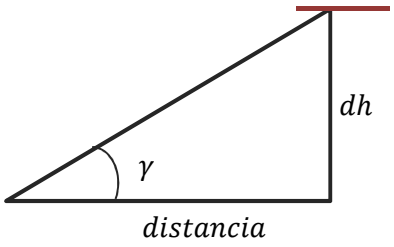
Las salidas instrumentales se obtienen de los ficheros de sector, con extensión .sct, de EuroScope, que contiene además información sobre pistas, aeropuertos, radioayudas, STARs, aerovías, zonas restringidas, peligrosas y prohibidas y todos los waypoints de un FIR. Esta información es actualizada por proveedores de la red de simulación de tráfico aéreo VATSIM.

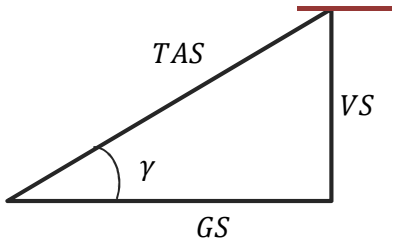
Hemos hecho uso de este documento porque ni ENAIRE ni EUROCONTROL proporcionan la información de sus cartas en formato texto.

Sin embargo, para las SIDs, este fichero SCT solo contiene los puntos de la salida y coordenadas intermedias entre varios puntos, pero no nos proporciona ninguna información en cuanto a restricciones de altitud o velocidad.

Por ello, se ha creado un fichero de texto plano con la información de las restricciones que hay en cada punto de la SID, además de contener la pendiente mínima de ascenso y la altitud inicial a mantener hasta finalizar la salida, si el tráfico no ha recibido otra indicación por parte de un controlador ATC.

En el modo de ascenso, se han fijado la TAS y el throttle, por lo que para cumplir con las restricciones deberemos modificar la velocidad vertical. En las siguientes figuras se representan las magnitudes a tener en cuenta para satisfacer la restricción.





Figuras 2.2 y 2.3. Pendiente y velocidad vertical para cumplir con las restricciones de altitud de la SID.

De la primera figura, conocemos la diferencia de altitud con respecto a la restricción, ya que sabemos a qué altitud vuela la aeronave y cuál es la altitud de la restricción a través del fichero de restricciones de las SID creado. Mediante la arcotangente calculamos la pendiente máxima o mínima, en función si es restricción de altitud máxima o mínima. Con la pendiente requerida y la TAS, de la segunda figura, despejamos la velocidad vertical máxima o mínima, a través del seno de la pendiente. Las ecuaciones quedarían de la siguiente manera:

$$\gamma_{max/min} = \tan^{-1}\frac{dh}{distancia} \qquad VS = \sin\gamma \cdot TAS$$

Ecuaciones 2.9 y 2.10. Cálculo de la pendiente y la velocidad vertical para cumplir con las restricciones.

La velocidad vertical será la máxima, si la restricción es de altitud máxima, es decir, si no se puede sobrepasar esa altitud o nivel de vuelo. A continuación, se calculará mediante la ecuación de ascenso 2.5 de BADA, la velocidad vertical y se comparará con la VS máxima que acabamos de calcular. Si la sobrepasa, esta se limitará a la VS máxima hasta que finalice la restricción. Si, por el contrario, no la supera, se aplicará la VS de la ecuación de BADA. Este proceso es el mismo para una restricción de altitud mínima, con la diferencia de que la VS que se calcula para la restricción es la mínima requerida. Ambas velocidades, la calculada para cumplir con la restricción y la que despejamos de la ecuación de BADA, son recalculadas en cada vuelta del bucle del simulador, de manera que con cada avance de la aeronave éstas se vuelven a evaluar para asegurarnos que se cumple con la restricción.

Al igual que como sucede con la TAS, la velocidad vertical obtenida no se aplica inmediatamente al tráfico, sino que se busca ésta mediante una aceleración normal (explicado en el apartado 2.6. Aceleraciones).

Para la pendiente mínima de ascenso, la tasa mínima de ascenso se calcula de la misma forma, despejándola de la ecuación 2.9. Se compara con la velocidad vertical calculada con la ecuación de BADA, y si está es inferior, se incrementa hasta que se alcance.

Por último, para mantener la altitud inicial hasta acabar la SID, ya que suponemos que en principio no ha habido ninguna autorización ATC, VS se fija a cero.

2.3. Crucero.

Si se mantuviera el empuje máximo, la velocidad de la aeronave sería la máxima. Pero lo normal es que no se vuele de esta manera, ya que se consumiría mucho combustible y se acortaría la vida útil de los motores. Por ello, se busca un valor apropiado de δp, para una velocidad y altitud dada.

En nuestro proyecto, se calculará directamente el empuje, aunque a través del empuje máximo podríamos saber con qué valor de δp está volando la aeronave. De la ecuación 2.1 de BADA, al igual que hicimos en el ascenso con la velocidad vertical, despejaremos en este caso el empuje. Suponemos que el avión vuelo a una altitud constante, por lo que la pendiente y la velocidad vertical son cero. La velocidad o número de Mach de la aeronave en crucero se calcula en función del índice de coste seleccionado en el plan de vuelo. La velocidad verdadera en crucero se recalcula constantemente durante esta fase en función del peso del avión. Esta variación del peso es tan gradual que el vuelo en crucero puede ser tratado con condición quasiestacionaria en todo momento. Por otro lado, el coeficiente de sustentación permanece constante. Estas suposiciones corresponden, según la mecánica de vuelo, al programa de vuelo de crucero 2 (PVC2), el cual está basado en perfiles de velocidad en función de la masa.

La ecuación queda entonces de la siguiente manera:

$$(Thr - D) \cdot V_{TAS} = 0 \qquad \rightarrow \qquad Thr = D$$

Ecuación 2.11. Empuje en crucero.

Ésta es la condición del equilibrio horizontal. A su vez es la simplificación de la ecuación tangencial de la mecánica de vuelo quasiestacionaria. Para obtener el empuje, simplemente calcularemos la resistencia aerodinámica de la misma forma que lo hicimos en el modo de ascenso.

Para la velocidad TAS, no se empleará en esta fase el *Airline Procedure Model*, ya que se ha optado por incluir en el simulador el factor del índice de coste. Éste nos proporcionará una TAS entre la velocidad de máximo alcance o MRC, la cual dependerá del peso instantáneo del avión, y la velocidad máxima operativa, o Mach máximo operativo, si se ha alcanzado la altitud de transición. El cálculo de esta velocidad se explica en detalle en el apartado 2.10. Índice de coste.

Por otro lado, se debe tener en cuenta que no se puede sobrepasar las limitaciones propulsivas del motor, es decir, el empuje máximo disponible, y esto solo lo podemos conseguir limitando la velocidad verdadera.

$$D = \frac{CD \cdot \rho \cdot {V_{TAS}}^2 \cdot S}{2} \qquad \rightarrow \qquad V_{TAS} = \sqrt{\frac{2\,D}{CD \cdot \rho \cdot S}}$$

Ecuación 2.12. Despeje de la velocidad de la fórmula de la resistencia.

Si combinamos las ecuaciones 2.11 y 2.12, la ecuación de la velocidad verdadera es:

$$V_{TAS} = \sqrt{\frac{2\,T}{CD \cdot \rho \cdot S}}$$

Ecuación 2.13. Velocidad verdadera máxima operativa en crucero.

El empuje máximo disponible para la fase de crucero es, según BADA:

$$(Thr_{cruise})_{MAX} = C_{Tcr} \cdot Thr_{\max climb}$$

Ecuación 2.14. Empuje máximo de crucero.

El primer término de entrada es el coeficiente C_{Tcr} , que se obtiene del fichero de parámetros globales BADA.GPF, ya que es el mismo para todos los tipos de aeronaves, siendo igual a 0.95. El segundo término corresponde al empuje máximo de ascenso, que vimos en el apartado 2.2.

2.4. Descenso.

En esta fase de vuelo, al igual que en el ascenso, se conocen el empuje y la velocidad TAS, calculada a partir del número de Mach o CAS. En función del tipo de descenso, si es escalonado o continuo, se obtiene la velocidad vertical.

2.4.1.Inicio del descenso.

Para definir el punto en el que la aeronave comenzará a descender se ha calculado el Top of descent de la ruta. Para ello, en primer lugar se define una pendiente de descenso, la cual se obtiene a partir del índice de coste, de 0 a 100, introducido por el usuario en el plan de vuelo. Los límites de la pendiente son -3º y -5º, es decir, si el usuario selecionase un índice de coste de 50, la pendiente de descenso sería -4º.

En primer lugar, se selecciona el punto o fijo donde terminará la pendiente de descenso. Para ello, se comprueba, comenzando desde el FAP, si se cumple la pendiente y no infringe ninguna restricción de la IAC y la STAR. Si es así, el FAP será el punto donde terminará la pendiente de descenso. En caso contrario, este punto quede determinado por el fijo que intersecta la pendiente y su altitud de restricción. Este sería el planteamiento teórico. En la práctica, la pendiente de descenso no se puede mantener hasta el FAP, ya que se debe asegurar que se llega a éste a la altitud que marca la carta IAC, por lo que si es necesario, se deberá corregir la pendiente antes de llegar a éste, como mínimo desde el IAF.

Una vez que tenemos el punto donde finaliza la pendiente, mediante la diferencia de altitud entre éste y el nivel de vuelo de crucero, y la pendiente, se calcula la distancia a la que se encuentra el punto de inicio de descenso, es decir, el Top of descent. Se despeja la distancia de la ecuación 2.9, de la forma:

$$distancia = \frac{dh}{\tan \gamma_{CI}}$$

Ecuación 2.15. Distancia de pendiente de descenso

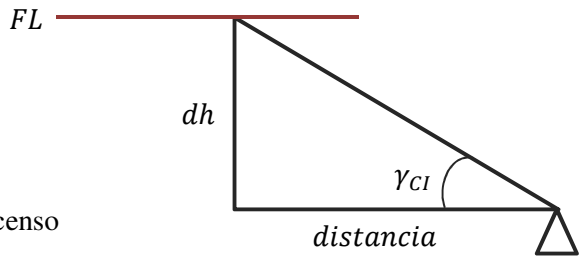


Figura 2.4. Distancia del Top of Descent

Con esta distancia recorrida podemos encontrar la ubicación del Top of descent o TOD. Para ello, se calculará la distancia entre cada tramo de waypoints o fijos hasta que se alcance o sobrepase esta distancia. Si se ha sobrepasado, el TOD se encuentra en el último tramo, por lo que a la distancia total se le resta la distancia recorrida hasta el último tramo, y nos dará una distancia restante al primer waypoint del último tramo. Desde éste y con rumbo segundo waypoint del tramo, se calculan las coordenadas del fijo a la distancia restante calculada. Y este será el Top of Descent para una determinada ruta.

Si el usuario cambiase la STAR en el simulador, lo cual le está permitido hasta antes de llegar al último waypoint de la ruta, el Top of descent se recalcularía.

2.4.2. Tipos de descenso.

Descenso continuo

En este modo, la pendiente queda fijada por el índice de coste, tal y como se explicaba en el apartado 2.4.1. Inicio del descenso. La velocidad la obtendremos una vez más con el *Airline Procedure Model* de BADA. Al igual que ocurría en el ascenso, para asegurar la monotonía en

el descenso, se sigue un programa de velocidad CAS basado en escalones de altitud, el cual, para aviones a reacción, es el siguiente:

de 0 a 999 ft	$C_{Vmin} \cdot (V_{stall})_{LD} + Vd_{DES,1}$
de 1.000 a 1.499 ft	$C_{Vmin} \cdot (V_{stall})_{LD} + Vd_{DES,2}$
de 1.500 a 1.999 ft	$C_{Vmin} \cdot (V_{stall})_{LD} + Vd_{DES,3}$
de 2.000 a 2.999 ft	$C_{Vmin} \cdot (V_{stall})_{LD} + Vd_{DES,4}$
de 3.000 a 5.999 ft	$\min(V_{des,1}, 220\ kt)$
de 6.000 a 9.999 ft	$\min(V_{des,1}, 250\ kt)$
de 10.000 ft a *crossover altitude*	$V_{des,2}$
por encima de *crossover altitude*	Ma_{des}

Ecuaciones 2.15. Velocidades BADA descenso

La simbología utilizada en las ecuaciones 2.15 es similar a la del resto de velocidades en las otras fases:

$V_{des,1}$	–	Velocidad CAS estándar por debajo de los 10.000 ft	[kt]
$V_{des,2}$	–	Velocidad CAS estándar desde *crossover altitude*	[kt]
Ma_{des}	–	Número Mach para el primer tramo del descenso	[-]
C_{Vmin}	–	Coeficiente de velocidad mínima	[-]
$(V_{stall})_{LD}$	–	Velocidad de entrada en pérdida durante el aterrizaje	[kt]
$Vd_{DES,i}$	–	Incremento de velocidad de descenso	[kt]

Los coeficientes C_{Vmin} y $Vd_{DES,i}$ son parámetros comunes a todas los tipos de aeronaves, por lo que se encuentra en el archivo BADA.GPF. Las tres primeras velocidades las encontramos en el fichero APF y la velocidad de entrada en pérdida del archivo OPF.

Con la velocidad verdadera de BADA y la pendiente obtenida del índice de coste, se despeja la velocidad vertical de la ecuación 2.10. Para cumplir con el modelo de la energía, como TAS y VS están fijadas, solo nos queda ajustar el empuje.

Descenso escalonado

La ecuación de la energía de BADA se despejará de la misma forma que en el apartado 2.2. Ascenso. Para obtener la tasa de descenso, debemos obtener primero el empuje, la resistencia aerodinámica, el factor de la energía y la velocidad verdadera.

El empuje para el descenso utiliza diferentes factores de corrección para altas y bajas altitudes, y para la configuración de aproximación y aterrizaje. En función de la altitud y la configuración, se aplicará uno de los cuatro empujes siguientes:

Configuración crucero:

$$Thr_{des,high} = C_{Tdes,high} \cdot Thr_{\max climb} \quad \text{para } H_p > H_{p,des}$$

$$Thr_{des,low} = C_{Tdes,low} \cdot Thr_{\max climb} \quad \text{para } H_p \leq H_{p,des}$$

Configuración aproximación: $Thr_{des,app} = C_{Tdes,app} \cdot Thr_{\max climb}$

Configuración aterrizaje: $Thr_{des,ld} = C_{Tdes,ld} \cdot Thr_{\max climb}$

Ecuaciones 2.16. Empujes para descenso, aproximación y aterrizaje

Los coeficientes $C_{Tdes,high}$, $C_{Tdes,high}$, $C_{Tdes,app}$, $C_{Tdes,ld}$ y la altitud $H_{p,des}$ se obtienen del fichero OPF de BADA, de cada tipo de aeronave.

Hasta el cambio a la configuración de aproximación o aterrizaje, el empuje que se aplica es el denominado en aeronáutica empuje de ralentí o IDLE, que consiste en el empuje mínimo que el motor puede proporcionar para que éste siga funcionando y sea capaz de suministrar energía eléctrica y aire a los pasajeros. Cuanto menor sea este empuje, más rápido será el descenso.

La resistencia aerodinámica se calcula igual que en las otras dos fases, variando los coeficientes de resistencia parásita e inducida, CD_0 y CD_2, en función de la configuración de la aeronave. Para la configuración de aterrizaje, suma al segundo miembro de la ecuación el término $CD_{0,\Delta LDG}$, que representa el aumento de resistencia del tren de aterrizaje, y que también está incluido en el archivo OPF del tipo de aeronave.

El factor de energía se obtiene con las mismas fórmulas que para el descenso. La única diferencia es que la energía disponible será negativa para el descenso.

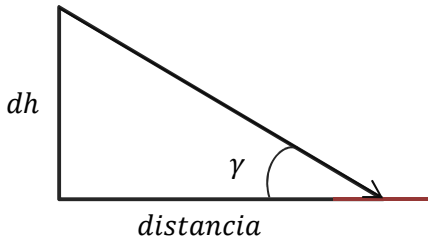
Por último, la velocidad verdadera se obtiene del *Airline Procedure Model*, al igual que en el descenso contínuo.

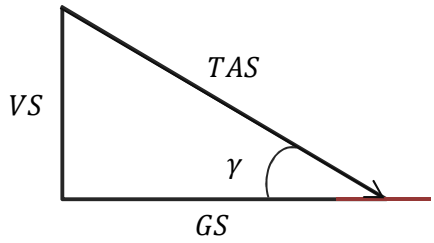
Estas ecuaciones son aplicables al descenso, aproximación y aterrizaje. En el simulador, hemos seguido en estas tres fases los procedimientos pertinentes a cada una de ellas.

2.4.3. Procedimiento de llegada instrumental o STAR.

Las llegadas instrumentales se encuentran ubicadas, al igual que las salidas, en el archivo .sct de EuroScope, pero no se incluyen las restricciones de altitud y velocidad. Por ello, se ha creado en la carpeta ENAIRE del proyecto, un fichero de texto, llamado RestriccionesSTAR, que incluye cada punto de la STAR o coordenada con restricción, seguido de la altitud de ésta, y con un signo positivo o negativo en función de si es una restricción de altitud mínima o máxima, respectivamente.

El modo de proceder para cumplir con las restricciones de altitud es el mismo que el seguido en las SIDs. Primero, se calcula la pendiente requerida para cumplir con la restricción a través de la distancia ortogonal a la restricción y la diferencia de altitud. Entonces, con el valor de la pendiente conocido, a partir de éste y la velocidad TAS obtenida del modelo, se calcula la velocidad vertical, mínima en el caso de ser un restricción de altitud máxima, y mínima, en el caso contrario.





Figuras 2.5 y 2.6. Pendiente y velocidad vertical para cumplir con las restricciones de altitud de la STAR.

Las fórmulas empleadas son las mismas del apartado de ascenso, ecuaciones 3.9 y 3.10. Solo cambia el la relación entre VS máxima y mínima y el tipo de restricción. Si la velocidad vertical obtenida de la ecuación 3.5 de la energía de BADA para el descenso escalonado, o de la ecuación 3.10 de la pendiente, supera la velocidad máxima, entonces la velocidad vertical se limita a alcanzar esta máxima. Si, en el caso de que la restricción sea de altitud mínima, la velocidad vertical no llega al valor de la VS mínima, se fija como objetivo ésta. Se le da prioridad a respetar las restricciones de altitud, antes que al modo de descenso elegido.

2.4.4. Procedimiento de aproximación IAC.

Las aproximaciones no están incluidas en el archivo de EuroScope donde se encuentran las SIDs y STARs. Solo contamos con el fijo de aproximación inicial (IAF), que corresponde al último punto de una determinada STAR. Ni ENAIRE ni EUROCONTROL proporcionan esta información en formato texto, de manera pública, por lo que debemos extraerla de las cartas aeronáuticas del AIP.

Para que el simulador pueda volarlas, se crea un fichero de texto plano, con las coordenadas del fijo de aproximación intermedia (IF) y el punto de aproximación final (FAP). La restricción de altitud la impone el FAP, por lo que desde el IAF se continuará el descenso para alcanzar la altitud del FAP en dicho punto. Para ello, calculamos la velocidad vertical requerida para que el tráfico alcance esa altitud justo en el FAP. El modo de hacerlo es el mismo que para cumplir con las restricciones de la STAR, salvo que no habrá VS máxima ni mínima, sino solo un VS requerida, por lo que no se empleará la ecuación de la energía ni la de la pendiente, ya que es primordial que al FAP se llegue con la altitud adecuada. Se recalculará esta velocidad vertical en cada ciclo del programa.

A partir del FAP, se descenderá con una pendiente constante de 3°, γ_{GS} o pendiente de la senda de planeo. Mediante esta pendiente y la TAS obtenida para altitud de la aeronave a través del *Airline Procedure Model*, calculamos la velocidad vertical para el tramo final de la aproximación.

$$VS = \sin\gamma_{GS} \cdot TAS$$

Ecuación 2.17. Velocidad vertical senda de planeo

Tras sobrevolar el FAP, el avión continuaría descendiendo con la pendiente de tres grados hasta alcanzar la altitud/ altura de decisión, donde el piloto decide si aterriza, en el caso de que tenga referencias visuales de la pista, o si prosigue con el procedimiento de frustrada. En el caso de este simulador, el tráfico continúa descenso manteniendo la pendiente de tres grados hasta alcanzar los 50 ft, donde comienza la maniobra de aterrizaje.

En el caso de que para la pista seleccionada del aeropuerto de llegada no haya incluido un procedimiento de aproximación, el tráfico realizará la siguiente maniobra. En primer lugar, volará hacia el fijo que se encuentra en la intersección del arco de radio 10 NM con centro en el umbral de la pista y la prolongación del eje de pista. Entonces, volará directo a un FAP ficticio, que se encuentra en la intersección del arco de radio 5 NM con centro en el umbral, y la prolongación del eje de pista, a una altitud de 1600 ft. A partir de este punto, la maniobra para el tramo de la aproximación final es el mismo que en el caso de que hubiésemos seguido una IAC.

2.5. Carreras de despegue y aterrizaje.

BADA no proporciona distancias de las carreras de despegue y aterrizaje, pero sí proporciona la FAR Take-Off Length (TOL) o longitud de pista compensada, que es aquella que la distancia de despegue es igual a la distancia de aceleración-frenada, con independencia de la longitud real de la pista en la que el avión despegará. Solo depende de características del avión, su configuración, peso, y de las circustancias atmosféricas, como presión, temperatura atmosféricas y densidad del aire. Esta distancia definen la longitud mínima de una pista en la que podemos realizar un despegue en condiciones de pista compensada.

También nos proporciona la FAR Landing Length (LDL) que es la distancia desde una altura de 50 ft hasta que el avión aterriza y se para completamente, incrementada un 67% por el factor regulatorio de la FAA, para una pista seca.

Para el despegue utilizaremos el criterio de pista compensada, es decir, la distancia de despegue o Take-Off Distance es igual a la TOL. Se debe tener en cuenta que la longitud de la pista donde se realice el despegue debe ser mayor que la distancia TOL, si queremos que la simulación se ajuste a la realidad.

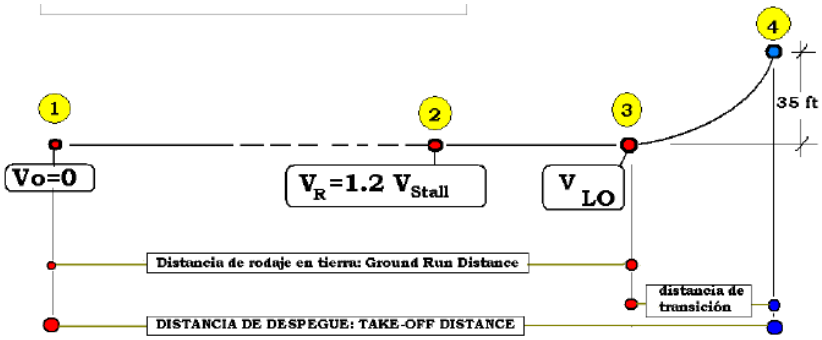


Figura 2.7. Distancia de despegue.

En este esquema vemos representado conceptualmente un despegue. Aplicaremos la hipótesis de que la rotación se produce instantáneamente, de manera que $V_R = V_{LO}$. Para saber en qué punto la aeronave debe comenzar el ascenso, debemos calcular la distancia de la carrera de despegue. Como conocemos la altitud en la que finaliza el despegue, 35 ft, y la velocidad vertical, despejada de la ecuación de BADA para la velocidad V_{LO} igual a la primera velocidad del *Airline Procedure Model*, podemos saber cuál es el tiempo que tarda en recorrer la distancia de transición. Este tiempo lo podemos transformar en distancia a través de la velocidad TAS. Se ha hecho la simplificación de GS=TAS para este tramo. Conocida la distancia de transición, se la restamos a la distancia total de despegue y obtenemos así la distancia de rodaje. Para realizar

la simulación, solo necesitamos acelerar la aeronave desde V_0 hasta V_{LO} en la distancia de rodaje calculada, y esto lo haremos a través de las ecuaciones del "Movimiento Uniformemente Acelerado".

$$d_{rodaje} = d_0 + V_0 \cdot t + \frac{1}{2}at^2 \qquad V_{LO} - V_0 = a \cdot t$$

Ecuación 2.18. Distancia y velocidad MRUA – Condiciones de contorno de despegue

Representamos en color gris los término nulos. A través de éstas condiciones de contorno, se sustituye en la primera ecuación el término de $(a \cdot t)$ despejado de la segunda. Con esto se obtiene el tiempo que tarda la aeronave en recorrer la distancia de rodaje. Con el tiempo despejado, sustituimos su valor en la segunda ecuación y obtenemos la aceleración. Entonces, podemos aplicar repetidamente las ecuaciones para calcular el incremento de velocidad en cada instante de tiempo y la distancia recorrida en cada intervalo, y así simular la carrera de despegue.

$$d_t = V_0 \cdot t + \frac{1}{2}at^2 \qquad V - V_0 = a \cdot t$$

Ecuación 2.19. Distancia y velocidad en cada intervalo de tiempo "t"

A partir de esta distancia d_t recorrida en un intervalo de tiempo, y con rumbo igual a la orientación geográfica de la pista, se calculan las coordenadas del tráfico.

Para el aterrizaje, se utiliza el valor de la LDL multiplicado por 0,67, para obtener la distancia real de aterrizaje, ya que la FAR LDL corresponde a la distancia requerida de aterrizaje tras aplicar el factor regulatorio para la pista seca.

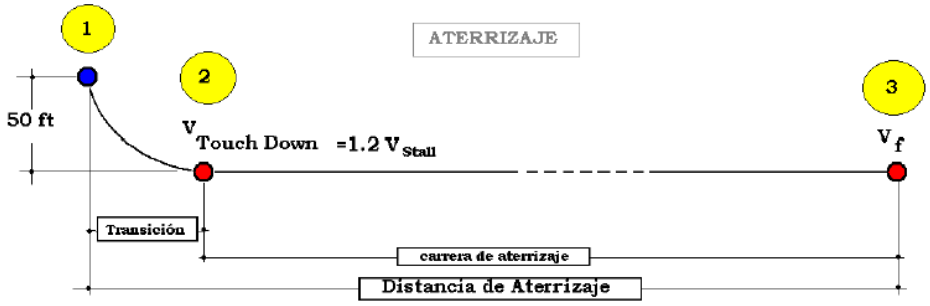


Figura 2.8. Distancia de aterrizaje.

Para calcular la carrera de aterrizaje, seguiremos los mismos pasos que para la de despegue. Conocidas la altitud del inicio de la transición, y la velocidad vertical en este punto, calculamos el tiempo de transición, hasta que la altitud es cero. Este tiempo lo convertimos en distancia, ya que sabemos la velocidad verdadera que lleva la aeronave. Esta distancia se le resta a la distancia real de aterrizaje, y obtenemos así la distancia de la carrera.

$$d_{carrera} = d_0 + V_{td} \cdot t + \frac{1}{2}at^2 \qquad V_f - V_{td} = a \cdot t$$

Ecuación 2.20. Distancia y velocidad MRUA – Condiciones de contorno de aterrizaje

De estas dos ecuaciones de contorno, despejamos el tiempo y la aceleración (desaceleración), y aplicamos las siguientes fórmulas para obtener la velocidad instantánea y la distancia recorrida en cada intervalo de tiempo.

$$d_t = V_0 \cdot t + \frac{1}{2}at^2 \qquad V - V_0 = a \cdot t$$

Ecuación 2.21. Distancia y velocidad en cada intervalo de tiempo "t"

Al igual que en la carrera de despegue, con la distancia recorrida en cada intervalo y el rumbo de la aeronave igual al rumbo geográfico de la pista de aterrizaje, se recalcula la posición del tráfico. Todas las ecuaciones de este apartado se resuelven en unidades del sistema internacional.

2.6. Selección manual de VCAS.

Otra funcionalidad del simulador para todas las fases de vuelo es que la velocidad del tráfico, en lugar de usar la fijada en función del índice de coste y del peso, en la fase de crucero, o la obtenida del *Airline Procedure Model*, en las fases de ascenso y descenso, se puede introducir manualmente en la interfaz del simulador en la columna de Throttle. De esta manera, el empuje se ajustará a la nueva velocidad verdadera obtenida a partir de la velocidad calibrada introducida por el usuario, según las ecuaciones 2.5 y 2.11, respectivamente. Por esta razón, se ha incluido en la interfaz como un modo Throttle.

La activación de esta característica es compatible con todos los modos de vuelo del simulador, ya que solo se modifica la fuente de la velocidad verdadera, por lo que en todos los casos se seguirá cumpliendo el modelo de la energía de BADA.

2.7. Aceleraciones.

Para pasar de una velocidad a otra, lo hacemos a través de aceleraciones. BADA proporciona dos parámetros en el archivo BADA.GPF, que son válidos para todas las aeronaves. Estos son la aceleración longitudinal máxima para vuelos civiles, que limita el incremento de la velocidad verdadera, y tiene un valor de 2 ft/s^2, y la aceleración normal máxima para vuelos civiles, que limita la tasa de ascenso/descenso a 5 ft/s^2.

El modo de aplicar estas aceleraciones es el siguiente:

- Para incrementar la velocidad verdadera, se obtiene la TAS actual y la TAS que se quiere alcanzar. A esta última la llamaremos a partir de ahora TAS objetivo. A la TAS actual se le aplica la aceleración máxima longitudinal de BADA, aplicando la fórmula contenida en el manual de usuario:

$$|V_k - V_{k-1}| \leq a_{l,max}\,\Delta t \quad \rightarrow \quad V_{objetivo} \leq V_{actual} + a_{l,max}\,\Delta t \;;$$

$$V_{objetivo} \geq V_{actual} - a_{l,max}\,\Delta t$$

Ecuación 2.22. Velocidad objetivo vs velocidad actual – aceleración longitudinal

El valor de la velocidad verdadera es en ft/s. Al despejar la inecuación, se obtienen dos fórmulas. La primera de ellas tiene sentido aplicarla a un incremento de velocidad positivo, es decir, la velocidad objetivo que se quiere alcanzar es mayor que la velocidad actual de la aeronave. La segunda, por consiguiente, se aplica a un decremento de velocidad.

- La ecuación de BADA para aplicar la aceleración normal no incluye velocidades verticales, sino que lo hace a través de la pendiente de ascenso/descenso actual y la pendiente que se quiere alcanzar o pendiente objetivo. La velocidad se expresa en ft/s y la pendiente en radianes.

$$|\gamma_k - \gamma_{k-1}| \leq \frac{a_{n,max}\,\Delta t}{V_{TAS}} \quad \rightarrow \quad \gamma_{objetivo,max} \leq \gamma_{actual} + \frac{a_{n,max}\,\Delta t}{V_{TAS}} \;;$$

$$\gamma_{objetivo,min} \geq \gamma_{actual} - \frac{a_{n,max}\,\Delta t}{V_{TAS}}$$

Ecuación 2.23. Pendiente objetivo vs pendiente actual – aceleración normal

Se calcula la pendiente actual de la aeronave, mediante la siguiente ecuación de BADA. También se podría haber despejado del triángulo de velocidades del apartado de ascenso o descenso.

$$\gamma = \sin^{-1}\frac{VS}{V_{TAS}}$$

Ecuación 2.24. Ecuación de la pendiente

Con la pendiente actual obtenida, se puede despejar las dos ecuaciones de la pendiente objetivo. Para obtener la velocidad vertical tras aplicar la aceleración, se vuelve a utilizar la ecuación X de la pendiente, de manera que:

$$VS_{MAX} = \sin\gamma_{objetivo,max} \cdot V_{TAS}$$

$$VS_{MIN} = \sin\gamma_{objetivo,min} \cdot V_{TAS}$$

Ecuación 2.25. Velocidades verticales máxima y mínima

Independientemente de si la velocidad vertical es positiva o negativa, si la VS objetivo se encuentra entre el máximo y el mínimo, esta se aplicará al tráfico, y si la VS objetivo es mayor que la VS máxima o menor que la VS mínima, se establecerán una de estas dos como VS actual, que equivale a aplicar una aceleración normal de 5ft/s^2, ya sea en ascenso o descenso. Este proceso se repite hasta que se alcance la VS objetivo.

2.8. Configuraciones.

BADA define 5 configuraciones aerodinámicas aplicables a una o varias fases de vuelo. Para cada una de ellas, el archivo OPF proporciona la velocidad de entrada en pérdida suministrada por el fabricante de la aeronave y los coeficientes de la polar, CD_0 y CD_2. Cada una de estas configuraciones, recibe a su vez un nombre, que indica la posición de los flaps y slats.

En la siguiente tabla, extraida del *BADA Performance Modelling Report*, se realaciona cada etapa y fase de vuelo con la configuración aerodinámica y el empuje correspondiente. Esta tabla clarifica que no deben confundirse configuración aerodinámica con la fase de vuelo de la aeronave.

Table 3-1: BADA Aircraft Operational Model

Stage	Phase	Aerodynamic configuration	Thrust setting
Climb	TKOF	tkof	max_cmb
	ICMB	icmb	
	CMB	clean	
Cruise	CRZ		max_crz
Holding	HLDG		
Descent	DES		des_hi
			des_lo
	APCH	apch	des_apch
	LDG	ldg	des_ldg
Ground	TAX	clean	idle

Los límites de cada una de estas configuraciones se encuentran definidos en el manual de usuario, y se pueden resumir de la siguiente manera:

Configuración despegue	TO	en ascenso hasta $H_{max,TO}$ AGL
Configuración inicio de ascenso	IC	en ascenso entre $H_{max,TO}$ y $H_{max,IC}$ AGL
Configuración crucero	CR	en ascenso sobre $H_{max,IC}$ AGL,
		en descenso sobre $H_{max,AP}$ AGL,
		en descenso bajo $H_{max,AP}$ AGL, si $V \geq V_{min,cr} + 10$ kt
Configuración aproximación	AP	en descenso entre $H_{max,AP}$ y $H_{max,LD}$ AGL,
		si $V < V_{min,cr} + 10$ kt
		en descenso bajo $H_{max,LD}$ AGL,
		si $V_{min,cr} + 10$ kt $> V \geq V_{min,approach} + 10$ kt
Configuración aterrizaje	LD	en descenso bajo $H_{max,LD}$ AGL,
		Si $V < V_{min,approach} + 10$ kt

Las configuraciones de despegue y ascenso inical están delimitadas únicamente con la altitud. El resto depende también de la velocidad verdadera. Los umbrales de altitud, $H_{max,i}$, se encuentran definidos en el archivo .GPF. Las velocidades mínimas son calculadas con la siguiente fórmula, siendo Vstall la velocidad de entrada en pérdida correspondiente a la fase:

$$V_{min} = C_{Vmin} \times V_{stall} \quad [\text{kt}]$$

Ecuación 2.26. Velocidad mínima

2.9. Rumbos y tasa de giro BADA.

En el modo LNAV, la ruta que seguirá un determinado tráfico está formada por los fijos de la salida instrumental que siga, los waypoints de la ruta definida en el plan de vuelo, los fijos de la llegada instrumental, y el fijo de aproximación intermedia y el punto de aproximación final. Cuando se alcanza uno de ellos, se calcula el rumbo que la aeronave debe llevar para alcanzar el

siguiente. Durante todo el trayecto hasta el siguiente punto o fijo, la aeronave recalculará el rumbo hasta éste.

Los rumbos son ortodrómicos, aunque la aeronave mantendrá este rumbo constante durante un intervalo de tiempo, dando lugar a una ruta orto-loxodrómica.

Para alcanzar rumbo ortodrómico calculado, al rumbo actual se le aplica una tasa de giro, definida por BADA, hasta que alcance el rumbo final, siendo ésta:

$$\dot{\phi} = \frac{g_0}{V_{TAS}} \times \tan \phi$$

Ecuación 2.27. Tasa de giro.

El ángulo de alabeo, ϕ, se encuentra definido en el archivo BADA.GPF. Se incluyen los ángulos máximos y nominales para aeronaves civiles y militares:

$\phi_{nom,civ\ (TO,LD)}$	15°	$\phi_{nom,mil}$	50°
$\phi_{nom,civ\ (others)}$	35°	$\phi_{max,mil}$	70°
$\phi_{max,civ\ (TO,LD)}$	25°		
$\phi_{max,civ\ (HOLD)}$	35°		
$\phi_{max,civ\ (others)}$	45°		

En el simulador se aplicarán únicamente los ángulos de alabeo nominales.

Comienzo del viraje

Suponiendo que todos los waypoints de la ruta serán del tipo *fly-by* o puntos de recorrido de paso, es decir, no es obligatorio el sobrevuelo de éstos, anticiparemos el comienzo del viraje para conseguir que la aeronave se dirija hacia el siguiente waypoint con el rumbo ortodrómico, es decir, recorriendo un arco de círculo máximo entre ambos.

En la siguiente figura, extraída de la Parte III — Sección 2, Capítulo 1, del Doc. 8168 "Operación de Aeronaves" Vol. II de OACI, se esquematiza el modelo para calcular la distancia de viraje.

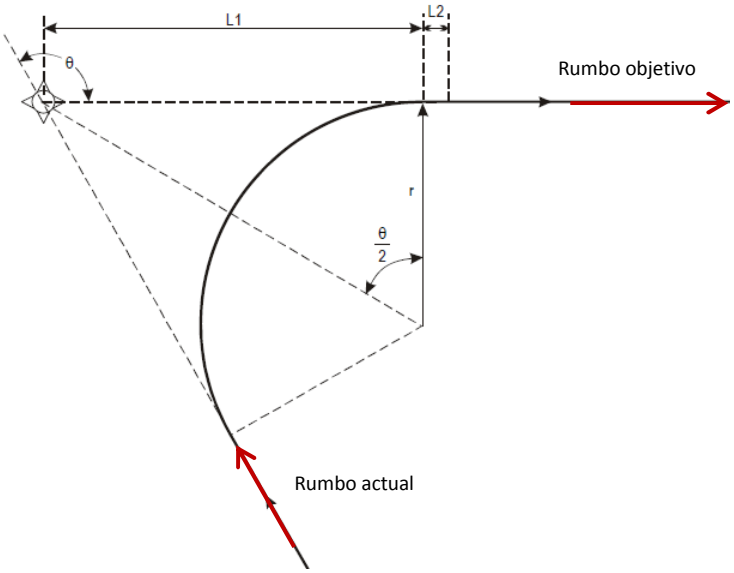


Figura 2.9. Distancia mínima de estabilización — punto de recorrido de paso.

Acorde al Doc. 8168, se ha calculado la distancia entre el waypoint y el comienzo del viraje, a través de la fórmula:

$$d_{viraje} = R \times \tan\frac{\theta}{2}$$

Ecuación 2.28. Distancia de anticipación del viraje.

En la ecuación 2.28., d_{viraje} corresponde a L1 de la figura 2.9, R simboliza el radio de giro y θ el ángulo de viraje. Este último es la diferencia entre el rumbo que lleva la aeronave hacia el waypoint y el rumbo objetivo que se desea seguir en dirección al siguiente. En la figura, L2 representa el retardo de 5 segundos que se considera como tiempo necesario para alcanzar el ángulo de alabeo. En la implementación del simulador no se ha tenido en cuenta. Para obtener el radio de giro, se divide la velocidad verdadera del avión entre la tasa de giro, y nos devuelve el radio en metros.

2.10. Consumo de combustible.

El consumo específico de un motor depende, entre otras cosas, de la altitud y la velocidad o Mach. Sin embargo, BADA solo lo define en función de la velocidad, de la siguiente manera:

$$\eta = C_{f1} \times (1 + \frac{V_{TAS}}{C_{f2}})$$

Ecuación 2.29. Consumo específico para motores a reacción.

Las unidades del consumo específico son kg/(min·kN). La velocidad verdadera se introduce en nudos. Los coeficientes C_{f1} y C_{f2} se encuentran en el archivo .OPF de la aeronave.

El flujo de combustible, en kg/min, se calculará de diferente forma en función de la fase de vuelo:

Crucero: $f_{cr} = \eta \times Thr \times C_{cr}$

Descenso: $f_{min} = C_{f3} \times (1 + \frac{H_p}{C_{f4}})$

Resto de fases: $f_{nom} = \eta \times Thr$

Aproximación y aterrizaje: $f_{ap/ld} = MAX(f_{nom}, f_{min})$

Ecuaciones 2.30. Flujo de combustible para cada fase de vuelo

Los coeficientes C_{f3} y C_{f4} , al igual que en la ecuación 2.29, se obtienen del archivo .OPF de BADA para cada tipo de aeronave. El empuje debe ser introducido en kN en las ecuaciones.

Un bajo consumo de combustible para una distancia recorrida será un factor importante para determinar la altitud y la velocidad de crucero de una aeronave en un vuelo comercial. La aerolínea podría tratar de buscar el máximo alcance por unidad de combustible consumido. Por otro lado, el tiempo de vuelo también tiene repercusiones económicas. Un avión comercial buscará la máxima autonomía en las esperas previas al aterrizaje, asimismo una patrulla marítima deberá volar mucho tiempo con una determinada cantidad de combustible. En este caso, se buscaría la máxima autonomía por unidad de combustible consumido.

10. Índice de coste.

El índice de coste (CI, del inglés *cost index*) es la relación entre el coste por tiempo de una operación y el precio del combustible. Las aerolíneas calculan un valor de CI para su operación y los pilotos lo introducen en el FMC, el cual calculará velocidades económicas para el ascenso, crucero y descenso. Para cada aeronave existe un rango disponible de CI, comenzando siempre en 0 hasta una cifra que varía entre 10^2 y 10^5. Sin embargo, en este simulador se introducirá en forma de porcentaje, de 0 a 100, ya que simplificará las ecuaciones.

$$CI = \frac{Coste\ por\ tiempo\ (\frac{\$}{h})}{Coste\ del\ combustible\ (\frac{cents}{lb})}$$

Ecuación 2.31. Definición del índice de coste

Un valor de CI igual a 0 resultará en velocidades de máximo alcance y el menor gasto de combustible. En el otro extremo, con valores máximos de CI se obtienen las máximas velocidades y el menor tiempo de vuelo, sin importar el gasto de combustible.

En crucero, la elección del nivel de vuelo y la velocidad afectaran al tiempo total de vuelo y al combustible total quemado. Hay dos velocidades relevantes para esta fase: MRC (Maximum-range-cruise) y LRC (Long-range-cruise). La primera corresponde a la velocidad que proporciona el máximo alcance para un determinado gasto de combustible, es decir, se corresponde con un valor de índice de coste cero. La segunda consiste en un aumento de la velocidad (3-5%) que reducirá un 1% las millas viajadas por kilogramo de combustible. Con el objetivo de simplificar la deducción de la velocidad verdadera dependiente del índice de coste,

este segundo valor será sustituido por el valor máximo de índice de coste, que corresponde a la máxima velocidad operativa. Las velocidades resultantes para la fase de crucero son:

Índice de coste 0 → Velocidad de máximo alcance

Índice de coste máximo → VMO/MMO

La velocidad máxima operativa y el Mach máximo operativo se encuentran en el archivo .OPF de BADA para cada tipo de aeronave. La velocidad de máximo alcance se obtiene de las ecuaciones de Angello Miele, que permiten calcular el óptimo del alcance y la autonomía, respetando unos determinados perfiles de velocidad. Para hallar el máximo alcance define un perfil de velocidad en función del peso. También podría ser en función de la altitud pero, en este caso, asumimos que la fase de crucero se volará a un nivel de vuelo constante. La ecuación de la velocidad de máximo alcance es la siguiente:

$$V_{\max\_range} = \sqrt[4]{3} \cdot V^* \qquad \rightarrow \qquad V_{\max\_range} = \sqrt[4]{3}\sqrt{\frac{2W}{\rho(z_0)SCL^*}} \qquad \text{con} \quad CL^* = \sqrt{\frac{CD_0}{K}}$$

La velocidad de máximo alcance está relacionada con la velocidad de mínima resistencia aerodinámica, V^*. Asimismo, la condición de máximo alcance exige volar con una velocidad que varía con $\sqrt{W}$.

Para la fase de descenso, lo más económico es emplear el empuje IDLE o de ralentí, lo que resulta en la máxima pendiente. Para un descenso escalonado, esto es posible, ya que únicamente se fija el empuje a IDLE y la velocidad verdadera según el *Airline Procedure Model*, por lo que se obtendrá una velocidad vertical que cumpla con el modelo de la energía de BADA. Pero para realizar un descenso continuo, se debe fijar una pendiente de descenso desde el Top of descent. La velocidad TAS se vuelve a obtener del modelo de la aerolínea. Debido a estas dos condiciones, para cumplir con la pendiente y a la vez con el modelo de la energía de BADA, se deberá ajustar el empuje.

Relacionando esto con el índice de coste, para un descenso continuo se fija que la pendiente variará entre -3º y -5º, siendo esta última la pendiente de descenso para un índice de coste igual a 0, y -3º la pendiente para el índice de coste máximo.

Capítulo 3

IMPLEMENTACIÓN DEL SIMULADOR

3.1. Lenguaje de programación.

El proyecto está enteramente desarrollado en el lenguaje Java, y emplea una programación puramente orientada a objetos. La plataforma utilizada es la Java Standard Edition (Java SE), en concreto, la versión Java SE 8. Su implementación nos ofrece herramientas de desarrollo de software, a través del Java Development Kit (JDK 8), bajo la licencia GNU General Public License v2 Classpath Exception (OpenJDK).

Para la ejecución del programa únicamente es necesario tener instalado y actualizado Java Runtime Environment o JRE de Java. Éste se compone de la máquina virtual de Java y un conjunto de bibliotecas y componentes que permiten la ejecución de aplicaciones escritas en lenguaje Java. Se puede descargar de la página oficial de Java, https://www.java.com/es/download/, o desde la página de Oracle, en el apartado de descargas de Java, http://www.oracle.com/technetwork/java/javase/downloads/index.html. Se encuentra disponible para los sistemas operativos Linux, Mac, Solaris y Windows.

3.2. Entorno de desarrollo (IDE).

Se ha utilizado el entorno de desarrollo de código abierto y gratuito Netbeans, el cual está desarrollado principalmente para el lenguaje de programación Java. Se ha empleado la versión 8.1. Se distribuye bajo la licencia Common Development and Distribution License (CDDL) v1.0 y la GNU General Public License (GPL) v2, la cual permite tanto su uso comercial como no comercial.

3.3. Estructura del programa.

Como se ha mencionado anteriormente, la programación del software es orientada a objetos. Esto quiere decir que tenemos clases que definen las plantillas para la creación de objetos, lo cual sucede a través de su instanciación. Estas clases contienen atributos, la mayoría privados, a los que accederán otros objetos para requerir o transferir información, mediante métodos *getters* y *setters*. Además, cada objeto tendrá sus propios métodos, y pueden ser sensibles a determinados eventos. También, tendremos clases las cuales instanciaremos únicamente para hacer uso de sus métodos, lo cual nos será bastante útil para agrupar todas las ecuaciones y métodos de conversión que usarán varios objetos creados a partir de diferentes clases. Nos evitamos la redundancia de código, a pesar de que en parte nos alejamos de la programación puramente orientada a objetos.

Podemos dividir las clases de este proyecto en cuatro tipos, según su funcionalidad:

- Clase principal.
  Esta clase es el punto de partida del programa. Cuando se inicia la aplicación, el primer método que se ejecuta es el Main de esta clase.

- Clases interfaces de usuario.
  Se instancian para que el usuario que va a hacer uso de la aplicación pueda interactuar con el programa. Son objetos que heredan de la clase JFrame de la biblioteca de interfaces Swing de Java. Ejemplos de estas clases son los formularios de conexión, o el menú principal.
- Clases para definir objetos.
  Todas las clases deben ser usadas para la definición de objetos, pero hemos agrupado en esta sección los objetos internos del programa, con los cuales el usuario no podrá interactuar, es decir, todos aquellos que no son interfaces.
- Clases auxiliares.
  Por último, estas clases se usan como librerías de funciones y procedimientos, ya que varios objetos diferentes utilizarán los mismos métodos en diferentes partes del programa.

A continuación, se detallará el diseño y la implementación de cada una de las clases del proyecto, para después explicar su funcionamiento de cara al usuario. Se puede ver el código de cada una de ellas en el Javadoc del programa.

3.3.1. Clase principal del programa.

Solo contiene el método estático *main*, al cual no le pasamos ningún argumento para su ejecución. En él, se instancia la clase MensajeConexionES, que es una interfaz que solicita la dirección IP del servidor de EuroScope al que nos conectaremos. Ésta es validada por el programa, y no permite continuar si no es posible conectarse a ningún servidor de EuroScope en la dirección IP introducida. Se han realizado pruebas de conexión, tanto en local, es decir, a un servidor en la máquina donde está siendo ejecutado el programa, como en remoto, conectándose al servidor de EuroScope del Laboratorio Pedro Duque, de la ETS de Ingeniería del Diseño, en la UPV.

Considerando que se ha validado la dirección IP del servidor de EuroScope, se reúne toda la información de navegación que será común a todo el programa. En primer lugar, se instancia la clase DatosENAIRE, y se ejecutan sus métodos obtenerRestriccionesSID(), obtenerRestriccionesSTAR() y obtenerAproximaciones(), para almacenar las restricciones de altitud y pendiente de las SIDs y STARs, y las aproximaciones, con sus fijos de aproximación intermedia y final, contenidas en los archivos de la carpeta ENAIRE del proyecto.

A continuación, se instancia la clase DatosES y se llama a su método obtenerdatosSCT(), el cual almacena los datos necesarios del archivo SCT de EuroScope de cada sector FIR/UIR, que contiene desde coordenadas de waypoints y ayudas de radionavegación, hasta aeropuertos, pistas y SIDs/STARs. En esta versión del programa solo se han añadido los FIR de Madrid y Barcelona.

Una vez que disponemos de la dirección del servidor y de los datos de navegación pertinentes, se lanza la ventana del menú principal, mediante la instanciación de la clase Visor. En el momento que el usuario cierre el menú principal, el programa se cerrará y finalizará su ejecución.

3.3.2. Clases interfaces.

3.3.2.1. Clase MensajeConexionES.

Es la clase a la que pertenece el primer objeto que se instancia en el método Main. Consiste en una ventana que solicita la dirección IP del servidor de EuroScope al que se conectarán todos los tráficos de una determinada sesión. El usuario debe introducirla y pulsar el botón "Enviar".

Al instanciar el objeto, con el constructor se crea la ventana. Éste llama a su método iniciar(), para crear los componentes de la ventana antes de añadirlos.

El botón "Enviar" tiene un manejador de eventos que avisa de cuando el botón ha sido pulsado, y se llama al método validarip(), que simplemente intenta abrir un socket de conexión TCP con EuroScope con la dirección IP facilitada por el usuario y el puerto 6809, que es el puerto cliente de EuroScope. Si lo consigue, la dirección IP se asume que es válida, y se cierra esta ventana, dando paso al menú principal.

3.3.2.2. Menú principal o clase Visor.

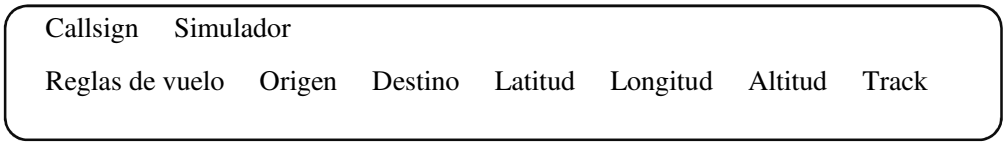
En primer lugar, como todas las interfaces del proyecto, hereda de la clase JFrame de la biblioteca Swing de Java. Con ella se crea la pantalla del menú principal de la aplicación, la cual está compuesta por: una barra de menú para seleccionar opciones, y una ventana.

En la barra de menú se encuentran tres pestañas:

- Tráfico: si se despliega, se puede ver las opciones *Nuevo*, para crear un nuevo tráfico de simulación, y *Desconectar*, para desconectar alguno o todos los tráficos conectados.
- Ver: contiene opciones para la vista de los tráficos.
- Ayuda: contiene un archivo html explicativo del programa.

El constructor de la clase tiene como parámetros la IP del servidor de EuroScope al que se conectarán los tráficos, la cual aparece en la esquina superior izquierda de la ventana, y el objeto de la clase DatosES que previamente se había instanciado en el método Main para almacenar los datos del fichero de EuroScope. Llama al método iniciar(), el cual inicializa y personaliza los componentes del menú, para posteriormente añadirlos a éste. Por último, se crea e inicia el hilo de la ventana. Solo finalizará cuando el usuario cierre el menú principal.

En el método run() del hilo, primero hay un retraso, Thread.sleep(int n), donde n es el tiempo de refresco de los datos, que este caso es de un segundo. Después, comprueba si alguno de los tráficos ha cancelado su creación en alguna de sus fases. Entonces, revisa si hay algún tráfico nuevo, para mostrar un nuevo panel de etiquetas con la siguiente información:



Esto lo hace mediante la llamada al método mostrar_trafico_nuevo, que tiene como parámetros el tráfico del cual se mostrará su información, y la variable *y*, que es la componente *y* de la ventana en la que se sitúa el nuevo panel.

Finalmente, actualiza la información de todos los paneles a través del método actualiza_trafico, que tiene un parámetro entero que indica el número del tráfico a actualizar. Si se ha desconectado o cancelado algún tráfico, se elimina de la lista de tráficos, y se elimina su panel de información, en el caso de que se hubiese creado.

Este método run() se repetirá hasta que el usuario cierre la ventana del menú principal.

3.3.2.3. Clase MensajeUsuarioES.

Se abre al pulsar Tráfico > Nuevo en el menú principal. Solicita al usuario los datos necesarios para enviar el mensaje de conexión de EuroScope, los cuales son: el callsign, el usuario, la contraseña y el nombre a mostrar. Al igual que la clase MensajeConexionES, en el constructor se crea la interfaz, y cuando el usuario pulsa el botón "Enviar" se recogen los datos.

Esta clase se instancia en la clase Tráfico, que se explicará más adelante. Si se cierra esta ventana, se cancela la creación del tráfico.

3.3.2.4. Clase MensajeElegirSim.

Esta interfaz se abre inmediatamente después de haber enviado los datos de conexión a través de la ventana de MensajeUsuarioES. Permite al usuario elegir si el tráfico que se conectará a EuroScope utilizará el simulador de vuelo XPlane o el simulador BADA creado en este proyecto. Si se selecciona esta última opción, en el menú principal aparecerá junto al tráfico un botón con el indicativo de SIMU, que permitirá controlar los modos de vuelo del tráfico. Además, se incluye una última opción "simulador BADA libre" que en lugar de situar el tráfico en la cabecera de la pista del aeródromo de origen, el usuario selecciona el waypoint de la ruta desde donde quiere comenzar la sesión de simulación.

En el constructor se crea la ventana y cuando el usuario haga click en "Enviar", la opción seleccionada se recoge en la clase donde se ha instanciado ésta, es decir, en la clase Tráfico.

Si el usuario cerrase esta ventana, volvería a la pantalla anterior, esto es, la del mensaje de conexión de usuario.

3.3.2.5. Clase PlanDeVuelo.

Se abre después de elegir el simulador, y se instancia en la clase Tráfico. Consiste en una plantilla que imita el plan de vuelo normalizado por OACI, similar a la solicitud de plan de vuelo online del sistema ICARO de ENAIRE.

Las reglas de cumplimentación se encuentran en el AIP, en la sección ENR 1.10-11. Dado que este formulario solo se usará para enviar el mensaje de plan de vuelo ($FP…) a EuroScope, se han omitido algunos campos, como el destinatario y la información suplementaria.

Esta clase, al igual que las anteriores, es una interfaz que se crea únicamente con el constructor, ya que hereda de la clase JFrame. Recibe como parámetros el callsign, ya que la identificación de la aeronave es un campo incluido en el plan de vuelo, y aseguramos que no hay diferencia con el introducido en el mensaje de usuario. Si el usuario se da cuenta de que ha cometido un

error en el callsign, siempre puede volver a la pantalla de los datos de usuario y rectificarlo. También recibe como parámetro el objeto de la clase DatosES creado en el Main del programa, para que en esta misma clase se pueda comprobar si los waypoints que el usuario introduce en la casilla Ruta, las SID y STAR, y los aeropuertos, son válidos.

La validación de los campos se hace a través del evento de pérdida de foco de los campos que necesitan validarse. Para poder llevar a cabo esto, la clase implementa la interfaz FocusListener.

La estructura de la interfaz está compuesta por un panel principal, al que se le van añadiendo paneles con controles de Java Swing que han sido previamente agregados a éstos. Los paneles se crean con un método, que tiene como parámetros la dimensión y la posición del panel, y se inicializan en el método iniciar() junto al resto de controles.

3.3.2.6. Clase MensajeSelecPista.

Esta clase se lanza desde la clase Tráfico, donde se instancia, justo después de enviar el formulario de plan de vuelo. Nos muestra los dos aeródromos, el de origen y el de destino, elegidos en el plan de vuelo, y las pistas disponibles de cada uno de ellos.

Esto lo hace de la siguiente manera. En el constructor se le pasa como parámetros el objeto con los datos de EuroScope, para poder acceder al array de aeropuertos, y el objeto plan de vuelo creado en la clase tráfico previamente, para obtener la información de los aeropuertos introducidos por el usuario. El método iniciar(), busca en el array de aeropuertos el que tiene como identificador OACI el mismo que el que introdujo el usuario en el plan de vuelo, tanto para el origen como para el destino. Entonces, rellena los objetos ComboBox de la interfaz con las pistas incluidas en el objeto Aeropuerto en los dos casos. De esta manera, en la interfaz aparecerá la designación OACI del aeropuerto junto a una caja desplegable con las pistas del archivo de EuroScope disponibles para cada uno de los dos aeropuertos.

Cuando el usuario pulse el botón aceptar, la clase tráfico recibirá la información de las pistas, de la cual utilizará la primera, para enviar la primera posición del tráfico y así situar el avión en la cabecera de la pista. Las coordenadas de las pistas del archivo .sct de EuroScope coincide con el umbral de éstas. Además, la clase tráfico obtiene de la pista del aeropuerto de origen seleccionada su orientación geográfica, la cual utilizará como rumbo inicial para el despegue, evitando así que se salga de la pista.

3.3.2.7. Clase MensajeSelecSID.

A esta interfaz se accede desde el panel del simulador. Sirve para cambiar la SID que se le ha asignado al tráfico por defecto. Le muestra al usuario en una ventana el aeropuerto elegido, la pista seleccionada y todas las SIDs posibles para esta combinación de aeropuerto, pista y primer waypoint de la ruta. Al seleccionar una de ellas, en la clase tráfico se modifica la SID asignada, por lo que cambian los puntos de la SID que volará el tráfico. Este botón solo está activado hasta que el usuario inicia el despegue.

Esta clase se instancia en la clase SIMU la primera vez que se pulsa el botón de la interfaz. Recibe como parámetros el objeto con los datos de EuroScope para acceder a las SIDs, aeropuertos y pistas, y el tráfico al que se le será asignada la SID seleccionada. Implementa la interfaz Runnable al iniciar un hilo en el constructor. Cuando se pulsa el botón “Enviar”, la SID se actualiza y la ventana se oculta, para no tener que volver a instanciarla cada vez que el

usuario acceda. Una vez que el tráfico despegue, este hilo se detendrá, y no se podrá volver a acceder a la ventana, ya que el botón aparecerá como desactivado.

3.3.2.8. Clase MensajeSelecSTAR.

Esta clase es bastante similar a la anterior. Se instancia también en la clase SIMU, con los mismos parámetros como argumentos. Permite elegir una llegada instrumental STAR distinta a la que le ha fijado el programa. De manera análoga, muestra el aeropuerto y la pista de llegada y todas las STARs disponibles con el primer punto de ésta coincidente con el último waypoint de la ruta. Además, hay dos botones de selección, que permiten la elección entre un descenso continuo, por defecto, o escalonado.

En el objeto tráfico se modifican la STAR y el modo de descenso. Esta clase también implementa la interfaz Runnable, de la misma manera que la ventana de selección de SID, que consiste en un hilo que únicamente se detendrá cuando el tráfico haya sobrevolado el último waypoint de la ruta, lo que desactivará el botón de la STAR de la ventana del simulador. Hasta este punto, el usuario podrá abrir esta ventana, cambiar de STAR y cerrar la ventana las veces que quiera.

3.3.2.9. Clase simulador o SIMU.

Esta interfaz permite al usuario modificar el modo de empuje y los modos horizontales y verticales de vuelo del tráfico, a través de un conjunto de componentes, principalmente botones y cajas de texto. A diferencia del resto de interfaces, esta clase hereda de un subtipo de JFrame, OpenMapFrame, ya que además de poder instanciar componentes y controles de la biblioteca Swing de Java, nos permitirá visualizar en un mapa la posición del tráfico y su ruta del plan de vuelo. Se abre cuando el usuario pulsa el botón SIMU del panel de uno de los tráficos creados que aparecen en el menú principal. En el marco superior de la ventana del simulador se indica el callsign de dicho tráfico.

El constructor personaliza la ventana y llama al método iniciar(), que inicializa los componentes, para después añadir éstos junto a sus escuchadores de eventos. Selecciona por defecto los modos plan de vuelo LNAV y VNAV. Por último, inicia el hilo del objeto, de manera que, cuando se abra por primera vez el simulador, éste se creará a partir de este constructor, y aunque se cierre la ventana, el hilo no se detendrá. Así, cuando el usuario vuelva a abrir nuevamente la ventana del simulador de un tráfico concreto, verá exactamente igual la pantalla con los modos de vuelos que habían sido seleccionados.

El método actionPerformed(ActionEvent ae), captura los eventos de acción de los controles de la ventana. En él solamente se pone en color verde el modo seleccionado, y en negro los demás, y se ponen a verdadero o falso las variables públicas booleanas de los modos de vuelo, para informar a las demás clases.

Por último, en el método run() del hilo se comprueban los modos seleccionados, se recoge información de las cajas de texto si fuera necesario, y se sitúan los waypoints de la ruta del plan de vuelo, junto con los de las SID y STAR elegidas, en la parte de la interfaz dedicada a ello. Al final de este método, se actualiza los datos del tráfico en el panel de información.

Una vez explicado el desarrollo de la clase, explicaremos la organización de los controles de la interfaz. Los modos de vuelo y su programación se explican en detalle en el Capítulo 2. Definición del simulador de vuelo basado en BADA y en clase enMovimiento.

En la parte superior de la pantalla hay un panel, de la clase OverlayMapPanel de la biblioteca de OpenMaps, que contiene el mapa de situación del tráfico, a través del cual se puede ver su posición, su rumbo y su ruta previamente definida en el plan de vuelo.

Debajo de éste, hay un panel de información del tráfico, que muestra la posición (latitud, longitud y altitud), el rumbo y las velocidades calibrada, verdadera y vertical de la aeronave.

Finalmente, debajo de estos dos paneles se encuentran los controles del simulador, que están agrupados en tres columnas:

- Modos horizontales.
  Hay dos. El modo heading, permite al usuario modificar directamente el rumbo del tráfico, mediante la introducción de su valor en grados en la caja de texto habilitada para ello, y su confirmación a través de la tecla ENTER. El rumbo del avión se irá modificando de acuerdo a la tasa de giro de éste, hasta que alcance el valor indicado en la caja de texto.

  El modo LNAV sigue la ruta de waypoints del plan de vuelo, y recalcula el rumbo ortodrómico hacia el siguiente waypoint. Los puntos de la ruta se encuentran debajo del botón de selección del modo. Hay cuatro botones para ver cuatro waypoints, y dos botones para desplazarnos por la lista de éstos. Aparecerá en verde el waypoint hacia el que el tráfico se dirige. Si se pulsa otro de éstos, el tráfico volará directamente hacia él, y seguirá el plan de vuelo desde ese punto.

- Modos verticales.
  Son tres. El primero, permite modificar la velocidad vertical del avión y ascender o descender hasta un nivel de vuelo concreto. Se introducen estos dos parámetros y se pulsa la tecla ENTER para confirmar. El siguiente modo, mantiene la altitud del avión. Finalmente, el modo VNAV establece la velocidad vertical mediante las ecuaciones de BADA y las restricciones de la ruta, siguiendo el plan de vuelo.

- Throttle.
  Permite modificar la velocidad calibrada CAS de la aeronave, la cual había sido previamente fijada por el *Airline Procedure Model*, en las fases de ascenso y descenso, y por el índice de coste, como función del peso, en la fase de crucero. Independientemente del modo horizontal y vertical seleccionado y de la fase de vuelo, ajustará el empuje a la velocidad CAS introducida.

En la parte inferior, se encuentran los botones Despegar, para iniciar el despegue del tráfico, y Pausar/Reanudar, para detener el tráfico.

El simulador asigna por defecto una SID y STAR, dependiendo del primer y último waypoint de la ruta incluida en el plan de vuelo. A través de los botones SID y STAR se pueden modificar. La SID solo es posible cambiarla antes del despegue, y la STAR, en cualquier momento antes de la llegada al último waypoint de la ruta.

En la caja de texto Squawk se puede modificar el código transpondedor del avión mediante la introducción de uno nuevo y pulsando la tecla ENTER para enviarlo.

3.3.2.10. Clase MensajeDesconexionTrafico.

Esta clase que se instancia en la clase Visor, desde la interfaz a la cual se accede a esta opción, permite desconectar un tráfico creado, tanto si emplea el simulador de BADA de este proyecto como si se ha conectado a través del simulador X-Plane. Cerrar un tráfico de BADA significa que se destruye la conexión y la instancia de éste. Si se cierra un tráfico de X-Plane, simplemente desconectará a éste del servidor de EuroScope, pero se podrá volver a conectar de nuevo en cualquier momento, sin necesidad de reiniciar la sesión de X-Plane, simplemente creando el tráfico de nuevo.

Esta interfaz permite eliminar varios tráficos a la vez a través de la selección de *checkboxs*. Tras pulsar el botón de "Enviar", los sockets de los tráficos se cerrarán y la instancia del tráfico desaparecerá del Visor.

3.3.2.11. Clase MensajePosicionLibre.

Es la interfaz que permite al usuario seleccionar un waypoint de la ruta y una altitud, para establecer la posición inicial en el modo Simulador Bada Libre, seleccionado en la interfaz de selección de simulador.

Se instancia en la clase Tráfico, después de que el usuario haya abierto el simulador del tráfico desde el Visor. Tras aceptar el envío de los datos, la clase enMovimiento, se saltará el despegue y el resto de fases de vuelo, y situará inmediatamente al tráfico en la posición seleccionada. Además, en el modo LNAV, el tráfico se dirigirá hacia waypoint que siga al seleccionado.

3.3.3. Clases para definir objetos.

4.3.3.1. Clase Fijo.

Esta clase permite crear y almacenar waypoints, puntos de las SIDs y STARs, y VOR/NDBs. Podría usarse para declarar cualquier otro tipo de punto, ya que los parámetros que recibe son su nombre o denominación, su latitud y su longitud, estas dos últimas en forma de cadena de texto, del tipo 'N042.18.17.000', o en formato numérico en grados, lo que da lugar a dos posibles constructores.

Estos fijos se crean en la clase DatosES para guardar los puntos del archivo .sct en forma de objetos, y así, almacenarlos en arrays de esta clase. Éstos son más manipulables para usarlos desde las demás clases. La clase tráfico los emplea para componer la ruta del plan de vuelo, así como para incluir los puntos de la salida y la llegada. La clase enMovimiento se encarga de recorrer este conjunto de puntos creado en la clase tráfico y recalcular los rumbos para alcanzarlos.

3.3.3.2. Clase Pista.

Esta objetivo de esta clase es la creación de pistas como objetos, a partir del archivo sct de EuroScope, y así poder añadírselas a los aeropuertos a las que pertenecen. De esta manera, se

simplifica la tarea de la selección de las pistas de un aeropuerto, a la vez que permite situar al avión antes del despegue y establecer su rumbo inicial. También permite la elección de la SID y la STAR.

El constructor del objeto tiene como parámetros la designación de la pista, su orientación geográfica, las coordenadas del umbral y el código OACI del aeropuerto al que pertenece, todos ellos en forma de cadena de texto.

Al final de la clase, se incluyen los *getters* para acceder a los atributos de ésta.

3.3.3.3. Clase Aeropuerto.

Facilita el manejo de los aeropuertos y sus pistas que seleccionará el usuario en el plan de vuelo, y que el programa asignará a un determinado tráfico para situar al tráfico en la pista del aeródromo de origen y como destino.

Se crea un array de pistas que se llena en el constructor de la clase. Éste recibe como parámetros el código OACI del aeropuerto y el objeto de la clase DatosES para acceder a las pistas, de manera que todas las pistas que tenga en su atributo "aeropuerto" el nombre de un determinado aeropuerto, se añadirán a éste. Esto hace que sea muy sencillo permitirle al usuario que elija una pista.

Por último, tenemos los dos *getters* de la clase, uno para el código OACI y otro para acceder al array de pistas.

3.3.3.4. Clase SIDSTAR.

Con esta clase se instancian objetos tanto SIDs como STARs. Se construyen a partir de los siguientes parámetros: el tipo, es decir, si es SID o STAR; el código OACI del aeropuerto junto al de la pista, esto es así porque en el archivo .sct de EuroScope las agrupan de esta forma, pero se separa en dos atributos distintos del objeto; el nombre de la SID o STAR; un indicativo para saber si es RNAV; y los objetos DatosES y DatosENAIRE creados en el Main de la aplicación. Además, en el constructor se llama al método de la clase añadirrestriccion(), que busca en el archivo de restricciones de ENAIRE si la SID o STAR que se crea tiene restricciones de altitud. Si es así, se añaden al atributo del tipo Restricción.

Esta clase tiene dos métodos más públicos, que permiten añadir los puntos de la SID o STAR, ya sean fijos declarados en el archivo de EuroScope, o coordenadas, un atributo del tipo array. Estos puntos se añadirán a los waypoints de la ruta, siempre separándolos de éstos, ya que si se modifica la SID o STAR, la ruta no se tiene que ver alterada.

Finalmente, los atributos privados tienen sus métodos *getters* para acceder a estas variables desde otras clases.

3.3.3.5. Clase Restricción.

Sirve para crear restricciones de altitud de las cartas de ENAIRE. Como no podemos acceder a esta información en formato texto a través de EuroScope o del AIP de ENAIRE, se ha creado previamente dos ficheros de configuración con las restricciones que vienen al final de las cartas SIDs y STARs. Mediante esta clase, se almacenan estas restricciones en forma de objeto y se asignan a las SID/STAR en cuestión. Cuando el tráfico se disponga a volar alguna de éstas, el

simulador simplemente tendrá que comprobar si tiene restricción, y si la tiene, leerá los atributos de este objeto, y actuará acorde a éstos.

El primer constructor de esta clase tiene como parámetros el nombre de la SID, su aeropuerto y pista, la altitud a mantener hasta el final de éstas si no ha habido ninguna autorización ATC que permita lo contrario, la pendiente mínima de ascenso hasta una altitud determinada, esta altitud concreta, y una cadena de texto con todas las restricciones de la forma Punto – Altitud – Punto – Altitud –.

El segundo constructor, para crear restricciones de las STARs, recibe el nombre de la STAR, su aeropuerto, su pista, y la cadena con las restricciones de la misma forma que en el primer constructor.

3.3.3.6. Clase Punto.

Hereda de la clase Fijo, ya que está formado por un nombre, las coordenadas de latitud y longitud, y además se le añade el atributo altitud. Se crea esta clase para poder instanciar los objetos FAP de las aproximaciones.

Tiene dos constructores, los cuales son heredados de la clase Fijo, es decir, permiten la introducción de las coordenadas en forma de cadena de texto, en grados, minutos y segundos, o en formato double, en grados. A ambos constructores se les añade el parámetro altitud.

3.3.3.7 Clase Aproximación.

Se hace necesario crear este objeto, debido a que está integrado por un fijo, el fijo de aproximación intermedia o IF, y un punto, el punto de aproximación final o FAP. Por esto, no se pueden agrupar en un array de fijos, como hacíamos con los puntos de la SID y la STAR, y se hace necesario crear una clase nueva.

Las aproximaciones son extraídas del fichero Aproximaciones.txt de la carpeta de ENAIRE del proyecto, ya que la única fuente de datos para obtener las aproximaciones instrumentales son las cartas IAC del AIP de ENAIRE. Se han pasado a formato texto algunas de éstas para poder utilizarlas en el simulador.

El constructor de esta clase recibe como parámetros para la creación de las aproximaciones, el nombre de IAC, el código del aeropuerto, la designación de la pista, un objeto Fijo con las coordenadas del IF, y un objeto Punto, con las coordenadas y la altitud del FAP.

Estos objetos aproximaciones se crean en la clase DatosENAIRE, y se agrupan en el array aproximaciones, de la misma manera que hacíamos con las restricciones de las SIDs y STARs.

Según el aeropuerto y la pista de destino seleccionados, se asignará un objeto aproximación al objeto Tráfico. Si para esa combinación de pista-aeropuerto no hubiese ninguna IAC, se le asignaría al tráfico un objeto de la clase Aproximación, del tipo “no IAC”, que está formado un fijo o IF ficticio, que tiene por coordenadas las de la intersección del arco de 10NM con centro en el umbral y la prolongación del eje de pista, y un punto o FAP improvisado, con coordenadas la misma intersección pero con el arco de radio 5 NM, y altitud de 1.600 ft.

3.3.3.8. Clase DatosES.

El fin de esta clase es recoger la información de navegación aérea contenida en los ficheros de EuroScope. En principio, el archivo que más nos interesa es el que tiene extensión .sct, en el que encontramos todos los aeropuertos de un FIR, sus pistas, sus respectivas SIDs y STARs y todos los waypoints, radioayudas y aerovías de éste. Esta información la almacena en arrays de objetos de las clases Aeropuerto, Pista, SIDSTAR y Fijo. Para la reutilización de éstos basta con pasar como parámetro este objeto en las clases que los queramos usar.

En el constructor solamente pasamos el objeto DatosENAIRE, ya que nos hará falta para añadir las restricciones de altitud en el momento de creación de las SIDs y STARs.

El método principal de esta es clase es obtenerDatosSCT(), que realiza la lectura de todo el fichero. Va leyendo línea a línea por bloques según su identificador, y divide estas líneas o cadenas de texto en los atributos para formar objetos de una de las clases anteriores, a través de otros métodos contenidos en la clase para tratar esta información, ya que no todas las líneas tienen la misma estructura ni la misma información. Éstos son: tratarWPT(), tratarVORNDB, tratarPista y tratarSIDSTAR. Los aeropuertos se crean a partir de las pistas, no necesitan ser leídos del fichero. Después crear los objetos, se añaden a su array correspondiente.

De esta manera, solo leeremos una vez el fichero y podremos acceder a la información tantas veces como deseemos desde otras clases.

3.3.3.9. Clase DatosENAIRE.

Tiene la misma estructura que la clase anterior. La utilizamos para leer las restricciones de altitud y pendiente de las cartas SID y STAR del AIP de ENAIRE. Esta información ha sido previamente guardada en dos ficheros de configuración contenidos en la carpeta ENAIRE del proyecto. La estructura del fichero de restricciones de las SIDs es la siguiente:

```
;---------------------------------------------- LEPA ----------------------------------------------
;24R
BAKAX2A     A4000     6.6%     A4000     BAKAX   [A4000+]
```

Las líneas con que comienzan por ";" son todas aquellas que no son restricciones y se usan para indicar el aeropuerto y la pista a la que pertenece la SID o STAR. Las líneas de restricción de la SID contienen la siguiente información, de izquierda a derecha: el nombre de la SID, la altitud a mantener hasta finalizar la salida si no ha habido ninguna autorización ATC que diga lo contrario, la pendiente mínima de ascenso hasta una altitud determinada, ésta altitud, y las restricciones de altitud de los puntos o tramos si las hubiese.

El diseño del fichero de restricciones de las STARs difiere en que no cuenta con la altitud a mantener ni los dos campos relativos a la pendiente mínima de ascenso. Tiene el siguiente formato:

```
;---------------------------------------------- LEPA ----------------------------------------------
;06L/06R
MORSS1P         MORSS [F230-] MHN [A5000+] PA24L09 [A5000+] CDP [A5000+]
```

El formato para la altitud es que utiliza ENAIRE en sus cartas. Si se trata de una altitud en pies, lleva el prefijo A; si se trata de un nivel de vuelo, el prefijo es la letra F. Para la cadena con las restricciones de los puntos, se acompaña al final de un signo, positivo en el caso de que el tráfico deba permanecer "a o por encima" de esa determinada altitud o nivel de vuelo, y negativa, si debe volar "a o por debajo" del mismo.

Los métodos recogerrestriccionesSID() y recogerrestriccionesSTAR() se encargan de leer estos archivos y almacenar en los arrays restriccionesSID y restriccionesSTAR todas ellas, con la estructura que ha sido citada en el apartado 3.3.5.

Asimismo, esta clase lee el fichero de las aproximaciones contenido en la misma carpeta. El formato de este archivo es similar a los anteriores.

```
;---------------------------------------------- LEPA ----------------------------------------------
;06L
IAC/1   IF   N039.28.07.000   E002.32.44.000   FAP   N039.30.11.000   E002.37.04.000   A1700=
```

En primer lugar, se indica el aeropuerto y la pista. Después, se listan las IACs de esa pista determinada, de la forma: nombre, "IF", coordenadas del IF, "FAP", coordenadas del FAP con altitud. La altitud utiliza la misma nomenclatura que para las restricciones, pero el símbolo empleado siempre es un "=", debido a que se tiene que cumplir exactamente esa altitud.

Mediante el método obtenerAproximaciones(), se lee todo el fichero y se crean los objetos Aproximación de todas las IACs, los cuales quedan almacenados en el array aproximaciones.

Esta clase se instancia en el Main del programa, ya que al igual que la clase DatosES, es un objeto que se utilizará como parámetro desde otras clases para acceder a la información almacenada en los arrays.

3.3.3.10. Clase DatosBADA.

A través de esta clase se obtienen todos los datos de los ficheros BADA para el tipo de aeronave seleccionada por el usuario en el plan de vuelo, los cuales se ubican en la carpeta BADA del proyecto. Contiene dos métodos principales, a los que se llama desde la clase Tráfico, una vez que se ha instanciado la clase DatosBADA. Estos son:

- obtenerDatosOPF() : recoge los coeficientes y constantes del archivo con extensión .OPF u *Operations Performance File*, del tipo de aeronave. En primer lugar, busca el archivo que contiene el nombre del tipo de aeronave. Después, lee línea a línea el archivo. Las líneas que empiezan por "CC" son comentarios, y las que empiezan por "CD" son líneas con datos, por eso es sencilla la obtención de éstos. Además, todos los archivos OPF tienen la misma estructura, por lo que es indiferente el tipo de aeronave seleccionada, para el desarrollo de este método. Los datos que se extraen de este archivo son los siguientes:
    - la masa de referencia;
    - la velocidad máxima operativa y el Mach máximo operativo;
    - la superficie alar;
    - la polar y la velocidad de entrada en pérdida para cada una de las configuraciones aerodinámicas de la aeronave;

- el incremento de resistencia aerodinámica producido por el tren de aterrizaje;
- los coeficientes para el cálculo del máximo empuje en ascenso;
- los coeficientes para el cálculo del empuje durante el descenso, y la altitud de transición para el cálculo del empuje en descenso;
- coeficientes para el cálculo del consumo de combustible;
- y las longitudes de despegue y aterrizaje FAR.

❖ obtenerDatosAPF() : recoge las velocidades CAS y número de Mach para cada fase del vuelo, en función del rango de masa de la aeronave, del archivo .APF o *Airline Procedure File*, de un determinado tipo de aeronave. Estas velocidades son utilizadas por el *Airline Procedure Model*, para determinar la velocidad en cada fase y altitud. La estructura del fichero es la misma para todos los archivos .APF. También hace distinción de líneas de comentarios y datos, de la misma forma que en el archivo .OPF.

❖ obtenerDatosGPF(): almacena los parámetros de BADA contenidos en el archivo BADA.GPF, los cuales son válidos para todas las aeronaves de BADA. La estructura del documento es similar a los dos archivos anteriores. Cada línea CD contiene la información de un parámetro, y tiene las siguientes columnas: símbolo o nomenclatura, tipo de vuelo (civil o militar), motor (jet, turbo o pistón), fase de vuelo, y su valor. La línea CC que precede a una línea CD contiene el nombre del parámetro y sus unidades. En esta versión de BADA se encuentran los siguientes:
  - aceleraciones longitudinal y normal máximas;
  - ángulos de alabeo nominales y máximo para cada fase de vuelos, según el tipo de éste (civil o militar);
  - factor de descenso de incremento de resistencia debido al uso de spoilers;
  - factores de empuje;
  - altitudes umbrales de configuraciones aerodinámicas;
  - coeficientes de velocidad mínima;
  - e, incrementos de velocidades del *Airline Procedure Model*.

Los datos que se leen de los archivos BADA, se almacenan en arrays, si es necesario, y se fijan a las variables pertinentes del objeto Tráfico que ha llamado a éstos métodos. Estas variables se usan principalmente desde la clase Cálculos, la clase auxiliar con los métodos para el cálculo de las velocidades y componentes de la dinámica de vuelo del simulador, mediante el uso de las ecuaciones proporcionadas por BADA.

3.3.3.11. Clase Tráfico.

Es una de las clases más importantes del programa. Dentro de las clases para definir objetos, se ha dejado para el final ya que hace uso de la mayoría de ellos. Esta clase representa un tráfico, simulado mediante las ecuaciones de BADA, Revisión 3.9, o que se conecta a través del simulador XPlane. Hereda de la clase Thread, e implementa un hilo que crea en el constructor. Solo se detendrá cuando el usuario desconecte ese tráfico concreto o cierre la aplicación.

En primer lugar, tenemos la declaración de variables de la clase. Éstas se pueden agrupar en dos tipos: las variables para el funcionamiento del programa y las variables o atributos del propio objeto Tráfico, los cuales se pueden dividirse a su vez en: variables del fichero de BADA, como

son coeficientes y velocidades CAS características de la aeronave; variables del simulador BADA, como pueden ser la velocidad vertical, la altitud, la TAS, entre otras; variables de EuroScope, en las cuales agrupamos todos los datos recogidos para enviar los mensajes de conexión, posición y plan de vuelo; y, por último, variables que son objetos de otras clases, como son el objeto SIMU, o interfaz del simulador, ligada al objeto Tráfico, o el objeto de la clase enMovimiento, que realiza los cálculos de posición del propio Tráfico.

En el constructor, simplemente se recogen la dirección IP del servidor de EuroScope al que se conectará el tráfico y el objeto de la clase DatosES para acceder a los waypoints, pistas, aeropuertos y salidas y llegadas instrumentales. Por último, dentro de éste, se inicia el hilo del tráfico.

La clase cuenta con un gran número de *getters* y *setters* debido a que la mayoría de las variables todas las variables declaradas como privadas, para controlar el acceso a éstas. Además de éstos, la clase tiene una serie de métodos para lanzar las interfaces y su recogida de datos.

Finalmente, en el método run del hilo de esta clase se crea una máquina de estados para el tráfico que se instanciará en el menú principal. Primero tenemos un bucle while condicionado por la variable booleana "running" que se inicia como verdadera. El estado inicial es 0. A continuación, hay un switch con una serie de casos, según en qué estado se encuentre el tráfico, los cuales son:

- Case 0: estado inicial. Se abre el socket de conexión a través del método inicio(), con la dirección IP que había sido validada en el mensaje de conexión al inicio del programa, y el puerto 6809, que es el puerto para las comunicaciones cliente/servidor de EuroScope.

  Si el socket se abre, el método inicio devuelve el booleano verdadero, y se pasa al estado 2. Si no es así, se le pide al usuario que revise el estado del servidor de EuroScope.

- Case 1: estado socket abierto. Se llama al método pedirDatosUsuario() que a su vez instancia la clase MensajeDatosUsuario y recoge los datos que introduce el usuario. Si el usuario ha pulsado el botón enviar, se pasa al siguiente estado. En caso contrario, si el usuario hubiera cerrado la ventana, se cancelaría la creación del tráfico y se detendría el hilo, de una manera limpia, estableciendo el valor de la variable "running" a falso.

- Case 2: estado datos de conexión recibidos. En este caso se llama al método pedirSim(), en el cual se instancia la clase MensajeElegirSim y se guarda la opción seleccionada por el usuario. Al igual en el caso anterior, si el usuario pulsa el botón "enviar" de la interfaz, el tráfico pasa al siguiente estado. Si no es así, es decir, si hubiese cerrado la ventana, se volvería a la pantalla anterior. Esta opción de volver para atrás no se ha implementado a través de un botón dedicado a ello para simplificar las interfaces.

  Si se ha pulsado el botón "Enviar", se comprueba si el usuario ha seleccionado el simulador XPlane. En este caso, se llama al método formularioConexionXP(), que lanza la interfaz MensajeConexionXP, para recoger la información de puertos y dirección IP del simulador X-Plane. Si el usuario completa este formulario y pulsa "Enviar", se llama al método enviarXP(), para enviar a XPlane los mensajes que deseamos recibir, y se instancia la clase RecibeXP, que permitirá la recepción contínua de los mensajes de XPlane seleccionados en el método anterior.

- Case 3: estado simulador seleccionado. Se lanza el formulario del plan de vuelo, a través de la instanciación de la clase PlanDeVuelo en el método pedirFP(), que a su vez recoge la información rellenada por el usuario. Todos los campos que lo requieran son validados. Finalmente, cuando el usuario pulse el botón "enviar" de la ventana, se pasará al siguiente estado. Si por el contrario la cierra, se vuelve al estado anterior.

- Case 4: estado datos del plan de vuelo recibidos. Se llama al método pedirPistas() que mostrará la ventana de MensajeSelecPista y recogerá las pistas seleccionadas para el aeródromo de origen y destino. Al igual que en tres casos anteriores, si se pulsa "enviar" se pasa al próximo estado, y si se cierra la interfaz, se vuelve al estado de antes.

- Case 5: estado todos los datos recogidos. A partir de aquí no hay vuelta a atrás. Se envía el mensaje de conexión, que tiene la forma:

  "#AP" + callsign + ":SERVER:" + usuario + ":" + password + ":1:9:11:" + nombre + "\n"

  #APEC-DAF:SERVER:111111:password:1:9:11:DAVINIA

  Para el envío de datos a EuroScope se emplea el método enviarES(String cadena), que envía una cadena de texto que recibe como parámetro. Para ello, obtiene el flujo de salida del socket, y crea un flujo de salida de datos, en el que escribe en Bytes la cadena de texto.

  Tras un retraso de un segundo, se envía el mensaje de plan de vuelo, que consiste en:

  "$FP" + callsign + ":*A:" + Flight_Rules(V|I) + ":" + Aircraft_type + ":" + TAS_{CR} + ":" + DepAirport + ":" + DepTime + ":" + Actual_Time + ":" + CruiseAltitude + ":" + ArrAirport + ":1:40:00:00:"+AltnAirport+":"+remarks+": " + DepAirport + " " + Route + " " + ArrAirport + "\n"

  $FPEC-DAF:*A:I:A320:373.937542:LEPA:1320:1320:23000:LEVC:1:40:00:00:LEAL:/v/: LEPA PETAM PINTO EPAMA ARGOR MULAT LEVC

  En este caso, se crea también la cadena de texto que dará nombre al fichero .txt que guardará la traza del tráfico. Para no sobreescribir ficheros, el nombre que se le asignará tendrá el formato "ddMMyyyy_HHmmss", siendo una combinación de la fecha y hora de este instante. Se crea el fichero, y se inserta la cabecera con el nombre de cada columna, mediante una llamada al método cabeceraTraza(nombre_del_fichero).

  Por último, se extraen los datos del objeto de la clase DatosBADA, según el tipo de aeronave seleccionado en el plan de vuelo. Para ello se instancia la clase y se llama a sus métodos obtenerDatosOPF(), obtenerDatosAPF() y obtenerDatosGPF(). Estos métodos están explicados en la clase DatosBADA, y su contenido se explica en detalle en el Capítulo 2. Definición del simulador de vuelo basado en BADA.

- Case 6: estado conectado y plan de vuelo enviado. Se asignan una SID y una STAR, a través de los métodos asignarSID() y asignarSTAR(), respectivamente. Estos métodos recogen todas las SID/STARs posibles, para un aeropuerto y pista dados, en dos arrays de la clase, llamados posiblesSID y posiblesSTAR, y de éstos, selecciona por defecto la

primera SID/STAR, y se le asigna a los atributos de esta clase SID y STAR. El usuario podrá cambiar la selección después a través de la interfaz del simulador.

También, se le asigna una IAC mediante el método asignarIAC(), en el cuál se recogen todas las aproximaciones disponibles para la combinación aeropuerto-pista seleccionada, en el array posiblesIAC. Si tiene varias IACs, se selecciona por defecto la primera, y se le asigna al objeto atributo IAC de esta clase. Si esta pista no tuviese ninguna IAC disponible, se crea un objeto Aproximación, en este mimo momento, de nombre "noIAC", que es el procedimiento de aproximación para aquellas pistas de las que no se indiquen sus procedimientos de aproximación instrumental en el archivo Aproximaciones.

Por último, se calcula el Top of Descent, a través del método calculo_punto_descenso(). En el Capítulo 2. Definición del simulador de vuelo basado en BADA se explica cómo se calcula este punto. Hecho esto, se pasa al siguiente estado.

- Case 7: estado listo para enviar primera posición. Se manda a EuroScope el mensaje de la primera posición, esto es, en el umbral de la pista de despegue, a la cual se accede a través del aeropuerto de salida de la clase que ha sido elegido en el plan de vuelo. También se fija la orientación del tráfico que coincide con la orientación geográfica de la pista. Entonces, se envía el mensaje:

  "@N:" + callsign + ":" + squawk + ":1:+" +latitud+ ":" + longitud + ":" + altitud + ":" + TAS + ":" + Rumbo + ":0" + "\n"

  @N:EC-DAF:2600:1:39.29422:-0.77576:19100:241.6:2787:0

  La letra "N" del principio es el modo del transpondedor. La letra N indica modo radar primario + modo A/C radar secundario. También es posible indicar modos StandBy, IDENT y distintas combinaciones en función de las respuestas de los radares que recibe.

- Case 8: estado situado en la cabecera de la pista, listo para despegar. En este momento, aparece el tráfico en un panel en el menú principal. En el momento que abra la interfaz del simulador a través del botón SIMU, se instanciará la clase enMovimiento(), que se encargará, como dice el nombre del método, de calcular las posiciones del avión. Además, se pinta la ruta y los waypoints con sus nombres a través de etiquetas, en el mapa del simulador. Una vez que ocurra esto, se pasará al siguiente y último estado. Si se ha seleccionado el simulador X-Plane, este paso se lo saltará.

- Case 9: estado funcionamiento normal. En este caso se envía la posición, cuyas componentes de latitud, longitud, altitud, rumbo y velocidad han sido previamente fijadas con nuevos valores. Se actualiza la posición del relativa del mapa respecto al tráfico, y se orienta la representación del avión en función del rumbo que lleve el tráfico. Se calcula el tiempo que ha transcurrido desde el último envío de posición, que en la primera vuelta será justo después de abrir el simulador, y se guarda la traza del tráfico, mediante el método guardarTraza(nombre_del_fichero, tiempo). Este método simplemente abre el fichero, guarda algunos de los atributos del tráfico, y cierra el fichero. De esta manera, si se desconecta el tráfico en cualquier momento, no se perderá información. Este fichero contiene los siguientes datos:

Latitud	Longitud	Alt [ft]	TRK [º]	CAS [KT]	TAS [KT]	VS [fpm]	Pte [º]	Conf.	Tiempo
N39º32'03.49"	E002º41'10.08"	1903.72	236.24	155.73	160.11	2315.6 5	8.21	IC	0:01:21.57

El tráfico permanecerá en este estado siempre que la conexión no falle o el propio tráfico se desconecte. El tiempo de refresco de la posición es de medio segundo.

3.3.3.12. Clase enMovimiento.

Esta clase es la encargada de recalcular y actualizar la posición del tráfico. Está muy vinculada a la clase SIMU, tanto que podríamos decir que la ventana del simulador es la interfaz de esta clase. Según los modos de vuelo que el usuario seleccione, la clase enMovimiento recalculará el rumbo, las velocidades y la altitud, y a partir de estos datos y el tiempo transcurrido, calculará las nuevas coordenadas de la posición del tráfico.

El constructor recibe como parámetro el objeto tráfico sobre el que recalculará su posición y velocidades. También se instancia la clase Cálculos. Esta clase hereda de la clase Thread e inicia su hilo en el constructor. Éste espera a que el usuario pulse el botón "Despegar".

Despegue

Se inicia la carrera de despegue incrementando su velocidad con una aceleración constante hasta alcanzar la primera velocidad CAS para el despegue que indica BADA para el tipo de aeronave seleccionado. Para el cálculo de la distancia recorrida y la velocidad alcanzada utiliza los métodos distancia(TAS_0, tiempo_) y velocidad(TAS_0, tiempo_) de la clase Cálculos. Ambos tienen como parámetro el tiempo transcurrido en cada pasada del bucle. El cálculo de tiempos para toda la clase se hace a través de la función System.currentTimeMillis(), que obtiene, como dice su nombre, la hora actual en milisegundos.

Conocidas la distancia recorrida por el avión, se recalculan la latitud y la longitud y se fijan estas variables en el objeto tráfico que habíamos pasado como parámetro en el constructor. Entonces, cuando en el switch del hilo de la clase tráfico se envíe de nuevo la posición, está tendrá los nuevos valores de latitud y longitud.

Cuando el tráfico alcanza la longitud de la carrera de despegue, se le asigna la velocidad TAS alcanzada y comienza el bucle "infinito" (este bucle solo termina cuando se desconecta el tráfico).

Bucle

En éste, primero se comprueba si el usuario ha pulsado el botón "Pausa". En tal caso, este bucle se detiene en el método wait(), y espera a que se le notifique que puede seguir. Esto sucederá cuando el usuario pulse de nuevo el botón de "Pausa".

Recogida de información actual del tráfico.

Primero, obtenemos la altitud que lleva el tráfico y calculamos su TAS. Para ello usamos el método determitarTAS(altitud) de la clase Cálculos. Todos los métodos que se citarán en esta clase estarán ubicados en la clase Cálculos, si no se indica lo contrario**.** Este obtiene la TAS según BADA para cada fase de vuelo y cada altitud. No se le asignará esta TAS directamente, sino que está pasará a ser la TAS_{Target} o TAS objetivo, que se alcanzará mediante la aceleración constante máxima para los vuelos civiles según BADA, esta es, 2 ft/s^2. Esta TAS ligeramente

incrementada por la aceleración es calculada mediante el método calculoTASactual(TAS, TAS_{Target}, tiempo_). Obtenemos entonces la CAS para pasársela al tráfico, con el único fin de guardarla en el fichero de la traza.

En el caso de que el usuario haya seleccionado una velocidad calibrada en la columna de Throttle, la TAS objetivo se calculará a partir de la transformación a TAS de la CAS seleccionada, y el proceso para alcanzarla será el mismo que en el caso anterior.

Se calcula la pendiente que lleva el tráfico a través de la arcotangente de las velocidades vertical y verdadera. También se comprueba en qué configuración se encuentra el avión, siendo 1. Take-Off (TO), 2. Climb (IC), 3. Cruise (CR), 4. Approach (AP) y 5. Landing (LD). En función de esta configuración se calculan los coeficientes aerodinámicos y la tasa de giro. Estas dos variables se fijan en el objeto tráfico también.

Por último, se recalcula la masa del avión, en función del consumo de combustible para la fase de vuelo y el tiempo transcurrido.

Comprobación de los modos de vuelo.

Una vez obtenidos todos estos valores, se comprueban los modos de vuelo seleccionados. Primero comprueba los modos horizontales. Si el usuario está en modo heading, calcula el heading con el método calculoheading(heading_Simu, TAS). El primer parámetro es el heading indicado en la ventana del simulador. Este método tiene en cuenta la tasa de giro según BADA. Después, calcula la distancia loxodrómica recorrida con el método calculoDtotal(TAS, tiempo_, pendiente). Conocidos el rumbo y la distancia, a partir de las coordenadas actuales se calculan las nuevas coordenadas del tráfico, mediante los métodos calculoLat2(distLoxo, rumbo, lat_0) y calculoLon2(distLoxo, rumbo, lon_0, Lat_nueva, lat_0). Se calcula también el heading de EuroScope, ya que hay que aplicar una fórmula concreta. Por último se fijan las nuevas coordenadas, rumbo y heading al tráfico.

Si el usuario hubiese seleccionado el modo LNAV, el cálculo de la distancia y las coordenadas es exactamente el mismo. Solo difiere la obtención del rumbo, que es el rumbo ortodrómico (rumbo de círculo máximo o de la distancia más corta entre dos puntos) hacia el próximo punto del plan de vuelo, así el avión realiza una ruta orto-loxo, es decir, una ruta ortodrómica en trozos de loxodrómica (a rumbo constante). Para ello hace uso del método calculoRumboPlan(wptdirecto, TAS), donde wptdirecto es el índice del waypoint de la ruta hacia el que se dirige. Otra diferencia respecto al modo heading es que la realización del viraje comienza a una distancia concreta del siguiente waypoint (de tipo flyby) para asegurar que el tráfico se dirige hacia el siguiente waypoint con un rumbo determinado.

Cuando se alcanza esta distancia, el índice wptdirecto se incrementa. Este índice se puede modificar también manualmente en la ventana del simulador, a través de la pulsación de los botones con los nombres de los waypoints de la ruta debajo del botón del modo LNAV.

Seguimos con los modos verticales. Si el modo VSpeed está seleccionado, con la velocidad vertical introducida en el simulador y el tiempo transcurrido desde la última vuelta del bucle, se recalcula la altitud, siempre que no se haya alcanzado la altitud/ nivel de vuelo introducido en el simulador. De esta manera, es la clase la que decide si la velocidad vertical introducida es positiva, si se tiene que ascender para alcanzar dicha altitud, o negativa, en caso contrario.

Si se ha pulsado el modo Hold, se mantendrá la altitud, la velocidad vertical es cero.

Y, finalmente, si se sigue el modo VNAV, el cálculo de esta velocidad vertical se realiza de diferente manera según la fase de ascenso, crucero o descenso. En ascenso, primero se comprueba si estamos volando la salida instrumental o SID. Si es así y ésta tiene restricción en el fichero de configuración de restricciones de la SID en la carpeta ENAIRE del proyecto, se realizan los siguientes pasos:

1º. Se comprueba si el siguiente punto de la SID tiene restricción de altitud, ya que por orden de prioridad, esto es lo más importante, dado que podría haber un obstáculo próximo. Si tiene restricción, se calcula la velocidad vertical requerida para cumplir con ésta, a través del método vspte(TAS, altitud, altitud_restriccion, Lat, Lon, wptdirecto). Entonces, nos fijamos en si la altitud de la restricción del fichero es un límite infrerior o superior, en función del signo positivo o negativo que se indique al final de ésta.

   Si la altitud es un límite superior, es decir, no se puede superar, vendrá acompañada del signo negativo, y la velocidad vertical requerida es la máxima permitida. Si la velocidad vertical actual del tráfico es mayor que esta VS permitida, entonces el tráfico reducirá su VS hasta alcanzarla. Si la altitud es un límite inferior, esto quiere decir que el tráfico deberá volar por encima de ésta, y por lo tanto, la velocidad vertical requerida será la mínima que debe llevar el tráfico para cumplir con la restricción. Por ello, si la velocidad vertical actual que lleva el tráfico es menor que la mínima se acelera verticalmente. Estos casos están explicados detalladamente en el Capítulo 2. Definición del simulador de vuelo basado en BADA.

   Con el método comprobarAceleracionNormal(pendiente, TAS, tiempo_, vsRequerida) se utiliza la aceleración normal máxima para ascender, según BADA, 5 ft/s^2 para vuelos civiles. Si la velocidad vertical requerida es mayor que la que se puede alcanzar con esta aceleración, se irá incrementando gradualmente en cada pasada de bucle la velocidad a través de la aceleración constante, hasta que se alcance.

2º. Las restricciones incluyen, como se vio en la clase Restricción donde se explica el contenido del fichero, una pendiente mínima de ascenso hasta que se alcanza una determinada altitud. Entonces, comprobamos si se ha alcanzado esa altitud, si no es así, se calcula la velocidad vertical mínima para cumplir con la restricción. Si con la velocidad vertical calculada mediante las ecuaciones BADA se cumple, no hay problema; si no es así, se establece esta velocidad vertical que cumple, como velocidad vertical objetivo, y se acelera progresivamente hasta que se alcance.

3º. Las restricciones también contienen una altitud a mantener hasta que finaliza la SID, si no ha habido ninguna autorización ATC. Vamos a suponer que este es el caso, entonces si no se ha finalizado la SID, una vez que se alcance esta altitud, se mantendrá hasta que el tráfico llegue al primer waypoint de la ruta.

Por último, si se ha acabado la SID con restricción o si la SID no tenía restricción, se calcula la velocidad vertical BADA de ascenso a través del método VSspeedcl(altitud, TAS), la cual se alcanza a través de la máxima aceleración normal constante.

Con la velocidad vertical calculada, se fija ésta en el tráfico y se recalcula la altitud con el tiempo transcurrido desde la última vuelta del bucle.

Si estamos en fase de crucero, la velocidad vertical es cero y se busca obtener el empuje necesario para cumplir con las velocidades TAS y VS. Se hace a través de la llamada al método

ThrCR(altitud, TAS). Esta TAS estará en función del Índice de coste durante esta fase. La fase de crucero terminará cuando se alcance el Top of descent, cuyas coordenadas han sido previamente calculadas en la clase Tráfico, al inicio del vuelo y, si ha habido algún cambio de STAR, después de éste.

Si el tráfico se encuentra en fase de descenso, primero se comprueba si la STAR que volará tiene restricciones en alguno de sus puntos o tramos. Si esto sucede, se irá comprobando punto a punto si tiene restricción de altitud.

Si un punto tiene restricción, se calcula la velocidad vertical mínima/máxima requerida, y al igual en el apartado primero de las restricciones de las SID, en función de si es un límite superior o inferior, esta VS requerida calculada en función de la pendiente será la velocidad máxima permitida, si es un mínimo de altitud; o la mínima requerida si es un límite superior de altitud que no se debe volar por debajo de él. Si la velocidad vertical del tráfico supera la velocidad máxima permitida o es inferior a la velocidad mínima, entonces se busca alcanzar la velocidad requerida a través de la aceleración constante mediante el método comprobarAceleraciónNormal().

Si la velocidad vertical actual ya cumple con la restricción o la STAR no tenía ninguna restricción, se sigue recalculando la velocidad vertical según las Ecuaciones de BADA, en el caso de un descenso escalonado, o para cumplir con la pendiente determinada por el índice de coste, en un descenso continuo.

Tras la STAR, cuando se alcanza el fijo de aproximación inicial, comienza la fase de aproximación. En ésta se ajusta la velocidad vertical para cumplir con la altitud del punto de aproximación final (FAP). A partir de este punto, se desciende con una pendiente constante de 3º, hasta que se aterriza. En el momento que la altitud es cero, comienza la desaceleración de la velocidad verdadera de la aeronave, hasta que ésta se detiene completamente. Desde que el tráfico se encuentra a 50ft de altitud hasta que se detiene es la fase de Aterrizaje, y el rumbo de la aeronave se mantiene constante e igual al rumbo geográfico de pista.

Finalmente, se actualiza la altitud, calculada en función de la fase, en el objeto tráfico. Aquí acaba la vuelta del bucle infinito del hilo de esta clase.

3.3.4. Clases auxiliares.

3.3.4.1. Clase Cálculos.

Esta clase se instancia en las demás fundamentalmente por tres razones: primero, para realizar conversiones de unidades, o convertir cadenas de texto a valores numéricos; segundo, para realizar cálculos de distancias y rumbos, que se utilizará desde la clase enMovimiento para calcular el cambio de posición de un tráfico, y desde la clase Tráfico, para obtener las coordenadas de las componentes del procedimiento de aproximación sin IAC; y, tercero, para realizar los calcular las variables de las ecuaciones BADA y cinemáticas del simulador.

Por esto, tiene dos constructores. Uno vacío, que no recibe parámetros, empleado para efectuar conversiones, y otro que recibe un objeto de la clase Tráfico, del que se obtienen y se fijan variables.

Conforme a lo dispuesto en el primer párrafo, citaremos los métodos de cada una de las tres categorías en las que hemos agrupado éstos.

- Métodos de conversión.
  - kmh2knot y knot2kmh: convierte velocidades en kilómetros por hora a nudos, y viceversa.
  - ms2knot y knot2ms: convierte velocidades en metros por segundo a nudos, y viceversa.
  - ft2m y m2ft: convierte altitudes o distancias en pies a metros, y viceversa.
  - fpm2ms y ms2fpm: convierte velocidades verticales en pies por minuto a metros por segundo, y viceversa.
  - deg2rad y rad2deg: convierte ángulos o rumbos en grados a radianes, y viceversa.
  - rad2degminsec y deg2degminsec: convierte coordenadas en radianes o grados, respectivamente, a cadena de texto con el formato [º ‘ “].
  - degminsec2rad y degminsec2deg: convierten coordenadas en formato [º ‘ “] a radianes o grados, respectivamente.

- Métodos de navegación.
  - calculoRumbo: recibe como parámetros las latitudes y longitudes de dos puntos, y calcula el rumbo inicial ortodrómico, del primero al segundo. Es utilizado por el método calculoRumboPlan para calcular el rumbo entre dos waypoints o fijos en el modo LNAV.
  - calculoDtotal: calcula la distancia recorrida por el tráfico, con una velocidad verdadera y una pendiente dadas, en un intervalo de tiempo. Los parámetros que recibe son la TAS, la pendiente y el tiempo. Se llama desde la clase enMovimiento, para calcular la nueva posición del tráfico en cada ciclo del programa.
  - distanciaOrto: devuelve la distancia ortodrómica entre dos puntos. Sus parámetros son las dos coordenadas de los puntos. Esta clase se instancia en la clase enMovimiento para calcular las distancias a las restricciones o en las aproximaciones.
  - calculoLat2: se obtiene la latitud de un segundo punto, tras aplicar un rumbo constante y una distancia desde unas coordenadas iniciales. Para ello, recibe el rumbo, la distancia recorrida y la latitud del primer punto. Se utiliza en la clase enMovimiento, para calcular una nueva posición.
  - calculoLon2: calcula la longitud del segundo punto. Los parámetros introducidos son los del método calculoLat2, más la nueva latitud obtenida de método y la longitud del primer punto.
  - calculoheading: devuelve el rumbo instantáneo, tras imponer un rumbo final, el cual se quiere alcanzar, y una tasa de giro. Recibe la TAS y el rumbo fijado por el usuario en la ventana del simulador o el rumbo del modo LNAV. Se llama desde la clase en movimiento, para calcular el rumbo instantáneo del modo heading, y desde el método siguiente, para calcular el rumbo alcanzado en un intervalo de tiempo hasta que se alcance el rumbo ortodrómico al waypoint.
  - calculoRumboPlan: calcula el rumbo ortodrómico al siguiente waypoint y devuelve el rumbo instantáneo tras aplicar la tasa de giro de BADA según la configuración aerodinámica. Recibe la posición del fijo del array que contiene todos los puntos de la ruta, incluidos los de la SID, STAR y aproximación, y la TAS del tráfico.

- Radiogiro: se obtiene el radio de giro, en metros, calculado a partir de la tasa de giro BADA y la velocidad verdadera que lleva el tráfico. Solo recibe la velocidad TAS como parámetro.
- RumboObjetivo: obtiene el rumbo ortodrómico entre el waypoint al que se dirige la aeronave y el waypoint siguiente a éste, de manera que debe ser el rumbo que debe tomar la aeronave al alcanzar el primer waypoint. Recibe la posición en el array de waypoints de la ruta de cada uno de éstos. El rumbo lo devuelve en grados.
- anguloviraje: calcula la diferencia entre el rumbo que lleva la aeronave hacia el siguiente waypoint y el rumbo objetivo hacia el siguiente a éste. El ángulo de viraje será, por tanto, el ángulo total absoluto que necesitará virar el avión, en grados.
- distanciacomienzoviraje: devuelve la distancia, en kilómetros, a la que el tráfico deberá comenzar a virar para que alcance el rumbo objetivo en dirección al siguiente waypoint. Recibe como parámetros el índice de los dos waypoints siguientes y la velocidad verdadera del tráfico.

- Métodos de ecuaciones BADA: todas las fórmulas y despeje de ecuaciones incluidos en este apartado están extraídos del manual de usuario de BADA, salvo las ecuaciones cinemáticas para simular las carreras de despegue y aterrizaje.
  - CAStoTAS y TAStoCAS: obtiene la velocidad verdadera de la aeronave, en función de la altitud y la CAS, y viceversa, a través de las fórmulas para la conversión de velocidades del modelo atmosférico de BADA.
  - MachtoTAS y TAStoMach: calcula la TAS a partir del Mach, y viceversa, en función de la altitud.
  - fmcCAS y fmcMa: métodos para obtener el factor de energía o *energy shared factor* de BADA, para CAS o Mach constante, respectivamente. También recibe como parámetro la altitud.
  - Tmaxclimb, TmaxCR, Tdesc: calculan el empuje máximo en ascenso, máximo en crucero y en descenso, en función de la altitud, a través de los coeficientes BADA.
  - ThrCR: despeja el empuje de la ecuación de la energía durante la fase de crucero, y comprueba que no supere el empuje máximo disponible para esta fase, TmaxCR.
  - VSspeedcl y VSspeeddesc: despeja la velocidad vertical de la ecuación de la energía para las fases de ascenso y descenso, respectivamente. Recibe como parámetros la altitud y la velocidad verdadera.
  - CL: calcula el coeficiente de sustentación, en función de la velocidad verdadera y la altitud. Además, tiene en cuenta el ángulo de alabeo.
  - CD y CDdesc: calculan el coeficiente de resistencia aerodinámica, a través de los valores de los coeficientes de resistencia parásita e inducida de BADA para cada configuración. El segundo comprende las configuraciones de descenso. Ambos dependen también de la altitud y de la TAS, ya que la ecuación de la polar contiene el término CL.
  - Drag y Dragdesc: devuelve el valor de la resistencia aerodinámica, en función de los valores de CD, la altitud y la TAS.

- Hptrans: obtiene el valor de la *crossover altitude* o altitud de transición, a partir de la velocidad CAS y el Mach.
- Configuracion: devuelve le valor numérico correspondiente a una configuración aerodinámica, siendo 1-TO, 2-CL, 3-CR, 4-AP, 5-LD. Dependen de la altitud, la velocidad CAS y la velocidad vertical.
- tasadegiroBADA: calcula la tasa de giro del tráfico a partir de la velocidad TAS y el ángulo de alabeo en función de la fase de vuelo.
- determinarTAS: obtiene el valor de la velocidad verdadera a partir de la CAS obtenida del *Airline Procedure Model*, en función de la altitud, la fase de vuelo y el tipo de aeronave. Para la fase de crucero, tiene cuenta el índice de coste incluido por el usuario en el plan de vuelo.
- calculoTASactual: devuelve la TAS tras aplicar la aceleración máxima longitudinal de BADA, durante un intervalo de tiempo (vuelta de bucle), a la TAS que lleva la aeronave.
- pendientemaxima: obtiene pendientes máxima y mínima que puede alcanzar el tráfico tras un intervalo de tiempo, según la aceleración normal máxima de BADA.
- comprobarAceleraciónNormal: calcula la velocidad vertical, tras aplicar aceleración máxima normal de BADA, a la velocidad vertical que lleva la aeronave. Se introduce como parámetro la velocidad vertical despejada de la ecuación del modelo de la energía, que se fija como velocidad vertical objetivo.
- tieneRestriccionAlt: comprueba si el siguiente fijo del array de la ruta tiene o no restricción de altitud. Este método se encuentra sobrecargado, ya que recibe el índice del waypoint siguiente o el nombre, y la restricción de la SID/STAR.
- altitudRestriccion: extrae la altitud, en pies, de la restricción del siguiente fijo, en formato texto.
- cadenaAlt2Double: transforma la altitud de la restricción en formato texto a valor numérico.
- pteRestriccion: calcula la pendiente desde la posición del tráfico hasta el punto donde comienza la restricción. Recibe la posición del tráfico, la altitud de la restricción y el waypoint con restricción. Devuelve la pendiente en radianes.
- vspteRestricción: devuelve la velocidad vertical mínima o máxima para cumplir con la pendiente a la restricción, en m/s.
- vs3degreeSlope: calcula velocidad vertical requerida para seguir la senda de planeo de 3º. Únicamente recibe la velocidad verdadera del tráfico como parámetro.

- Métodos para las carreras de despegue y aterrizaje: se incluyen las ecuaciones cinemáticas para calcular las distancias, velocidades y aceleraciones en las carreras de rodaje del despegue y el aterrizaje.
  - carreraTO: calcula la distancia de la carrera de despegue, restándole la distancia de transición a la longitud de pista compensada proporcionada por BADA.
  - carreraLD: calcula la distancia de la carrera de aterrizaje, de manera similar a la anterior. La longitud de aterrizaje FAR que obtenemos de BADA se divide entre el factor regulatorio de la FAA y se le resta distancia desde los 50ft hasta que el touch-down.

- acelTO: calcula la aceleración constante para alcanzar la velocidad BADA en la distancia de la carrera de despegue.
- acelLD: calcula la desaceleración necesaria para frenar el tráfico en la distancia de la carrera de aterrizaje calculada.
- distancia y distanciaLD: calcula las distancias instantáneas que se recorren durante las dos carreras.
- velocidad y velocidadLD: calcula las velocidades instantáneas que se alcanzan en las carreras tras aplicar la aceleración/desaceleración durante un determinado intervalo de tiempo.

- Métodos para el índice de coste: permiten calcular la velocidad verdadera para la fase de crucero y la velocidad vertical de descenso para cumplir con el índice de coste introducido por el usuario en el plan de vuelo.
  - V_max_alcance: devuelve la velocidad verdadera de máximo alcance para crucero. Es una ecuación extraída de la mecánica de vuelo, que al mantener la altitud constante en la fase de crucero, define un perfil de velocidad en función del peso del avión. Se explica en detalle en la sección del Capítulo 2. Definición del simulador de vuelo basado en BADA.
  - CAS_CI: calcula la velocidad CAS para crucero en función del índice de coste, la velocidad máxima operativa y la velocidad de máximo alcance.
  - Ma_CI: calcula el número de Mach para crucero, en el caso de que la aeronave haya superado la altitud de transición o crossover altitude, en funcióndel índice de coste, el Mach máximo operativo, la velocidad de máximo alcance.
  - VSDCM: obtiene la velocidad vertical de descenso para seguir la pendiente definida por el índice de coste. Depende de la velocidad verdadera de BADA.

Capítulo 4

COMPLEMENTO PARA EL SIMULADOR DE VUELO X-PLANE

En la sección 4 "Implementación del programa" se indicaba la posibilidad en la interfaz de usuario de elegir entre el simulador BATS desarrollado en este proyecto y el simulador de vuelo X-Plane. Esta segunda opción habilita la conexión y el envío/recepción de datos entre el programa y un simulador X-Plane que se encuentre corriendo en la misma máquina o en remoto.

Este complemento adicional permite instanciar los tráficos del simulador X-Plane en el mismo servidor que los tráficos del simulador BATS, de manera que la simulación del vuelo se podrá realizar a través de la interfaz de la cabina del avión en X-Plane, o bien desde la ventana del pseudopiloto del simulador BATS.

4.1. Conexión e interacción con X-Plane.

Además del preciso modelo de vuelo, X-Plane cuenta con otra característica que lo hace muy potente: envía datos a través de la red y permite la recepción de ciertos parámetros. Para ello utiliza el protocolo UDP (User Datagram Protocol) basado en el intercambio de datagramas. Sus principales características son: no necesita de una conexión/ sincronización previa entre el origen y el destino de los mensajes; envía datagramas o paquetes de datos y no bytes individuales (protocolo TCP); no chequea si ha ocurrido un error en la recepción de un paquete, ni los ordena, por ello se dice que no es fiable; y, tiene la ventaja de que los paquetes que se envían son más simples, ya que reduce la cantidad de información necesaria, y eficaz, en el sentido de que el envío es más rápido.

Las conexiones UDP trabajan con puertos que tienen una longitud de 16 bits, lo que da un rango del 0 al 65.535. Los puertos del 0 al 1024 están reservados. Todos los puertos se pueden configurar en la ventana de Conexiones de red, de la pestaña Ajustes.

En la imagen 4.1, se diferencian dos bloques, uno para la salida de datos y otro con los puertos UDP. En el primero, le indicaremos la dirección IP del equipo al que X-Plane enviará los datos y el puerto de salida. Desde nuestro programa recibiremos estos datos via UDP, y este será el puerto que utilizaremos. La dirección IP será "127.0.0.1" si estamos ejecutando el programa del simulador en local. Si queremos enviar los datos del simulador X-Plane desde otro ordenador, se introducirá su dirección IP. En el segundo apartado, podemos cambiar los puertos UDP. De estos tres, solo nos sirve el primero, el puerto UDP de recepción, ya que el puerto de envío de datos desde X-Plane, se ha configurado en el apartado anterior. X-Plane recibe en el puerto 49000, por defecto, pero se puede cambiar por otro.

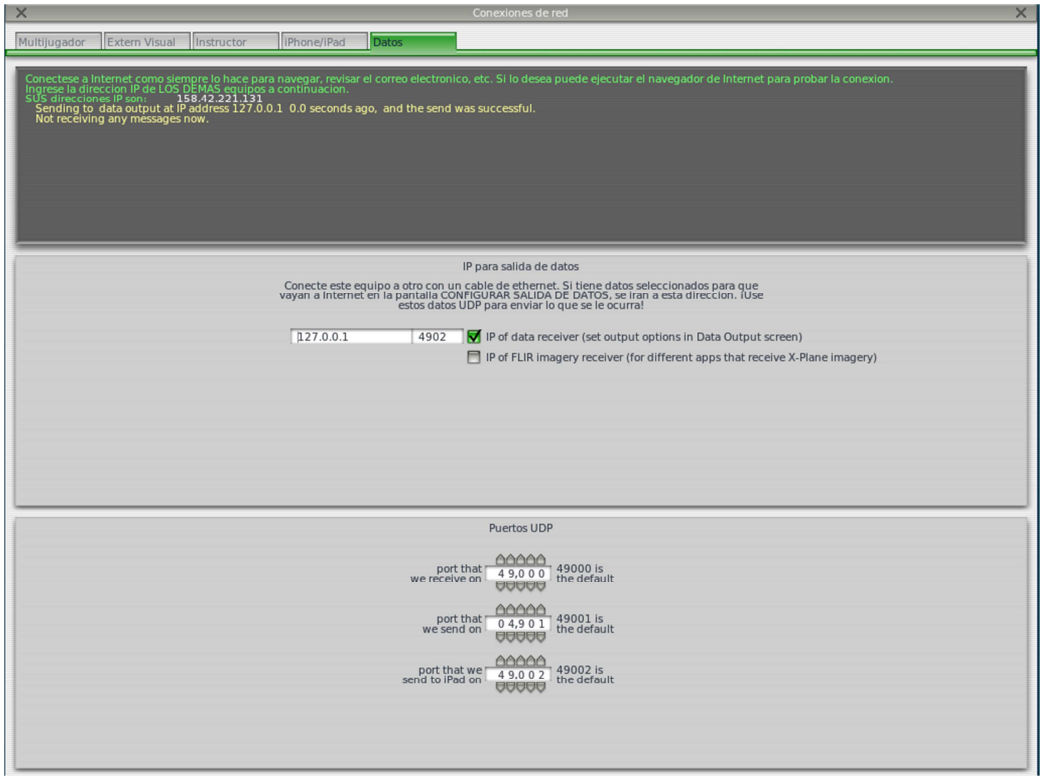


Figura 4.1. Ajuste Conexiones de red, X-Plane

Una vez definidos los puertos de conexión, se desarrolla la parte del programa para el envío y recepción de mensajes.

4.2.Programación.

Para la implementación de esta nueva funcionalidad del programa se han añadido dos nuevas clases, una del tipo de interfaz de usuario y otra para definir el objeto de recepción de datos que se instanciará en la clase Tráfico si corresponde. Para el envío de datos solo se ha añadido un método, ya que solo se enviarán cuatro mensajes, por lo que no será necesario crear un hilo adicional para ello.

4.2.1. Clase MensajeConexionXP.

Solicita al usuario la dirección IP de la máquina a la que se enviarán los datos de X-Plane, el puerto de recepción y el de envío desde X-Plane. Esta clase hereda de JFrame, por lo que se crea una interfaz de usuario solo con instanciarla. No recibe ningún parámetro. Proporciona *getters* para la obtención de los puertos y la IP desde la clase Tráfico en la que se instancia.

4.2.2. Clase RecibeXPlane.

Implementa la interfaz Runnable, la cual permite crear un hilo paralelo a la ejecución del programa que solo se encarga de recibir datos enviados por el puerto de salida de X-Plane.

En primer lugar, crea un socket UDP o DatagramSocket con este puerto. A continuación, en un bucle se creará un paquete para la recepción de datos o DatagramPacket, y a través del socket se

guardarán en él los datos que se reciban. Se extraen los datos en un vector de bytes para su manipulación.

Todos los mensajes de X-Plane están formados por una cabecera de 5 bytes y un espacio de datos de 36 bytes de longitud. El índice o número del mensaje que se recibe viene incluido al principio del espacio de datos, tras la cabecera. Para encontrar este índice de mensaje se emplea la fórmula [5+36k], siendo k el orden en que se solicitó el mensaje en el método enviaXP(), que se verá en el siguiente apartado.

Cada mensaje contenido en el vector de bytes se separa en vectores de 4 bytes para su transformación a decimales coma flotantes y se almacenan en un vector de 8 posiciones de tipo coma flotante, a los cuales se accederá desde la clase Tráfico.

4.2.3. Método enviaXP().

Este método ubicado en la clase Tráfico, envía los números o índices de los mensajes que queremos recibir de X-Plane.

Todos los mensajes disponibles se encuentran en X-Plane en Ajustes, datos de entrada y salida. En esta ventana podemos ver que cada mensaje cuenta con 4 checkboxs. El primero permite la observación del mensaje en la cabina mediante el uso de etiquetas, el segundo y tercero permiten visualizar estos datos después del vuelo de manera gráfica o en forma de texto, y el cuarto habilita el envío de estos mensajes vía UDP.

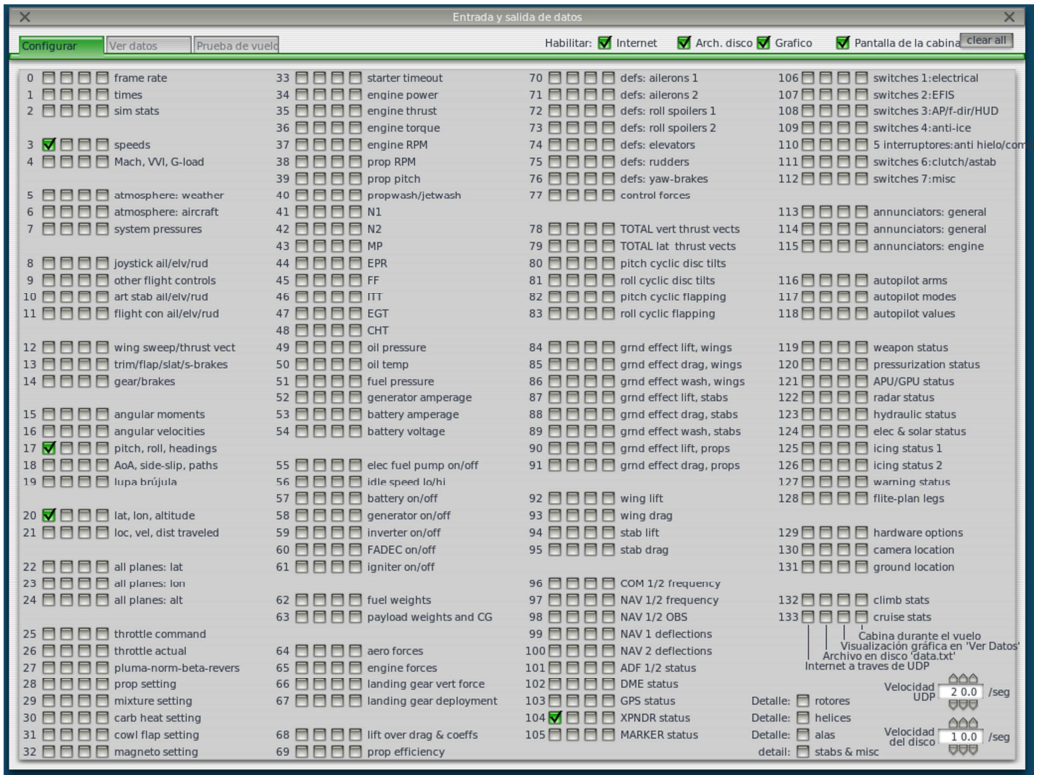


Figura 4.2. Entrada y salida de datos, X-Plane

Para que el usuario no tenga que seleccionar los mensajes de manera manual, en este método se envían estos índices de mensaje a X-Plane a través de un socket UDP. Para ello es necesario

crear un paquete, que estará formado por el mensaje en forma de array de bytes, la longitud del mensaje, la dirección IP donde se está ejecutando X-Plane y el puerto de recepción de X-Plane. Estos dos últimos los obteníamos de la interfaz MensajeConexiónXP.

El mensaje para activar el envío de un determinado mensaje de X-Plane debe comenzar por "DSEL", y para desactivarlo, por "USEL". Para activar y desactivar la visualización en la cabina de estos mensajes se emplean "DCOC" y "UCOC".

Los mensajes que enviaremos en este método son los DSEL de los siguientes mensajes:

- Mensaje Nº. 3: velocidades de la aeronave.

- Mensaje Nº. 17: ángulos de euler, de los cuales nos interesa el rumbo del avión.

- Mensaje Nº. 20: posición de la aeronave, esto es, latitud, longitud y altitud.

- Mensaje Nº. 114: código transpondedor y modo.

Una vez realizado el envío de estos mensajes, el simulador comenzará a recibir estos datos vía UDP mediante el hilo de la clase RecibeXP.

Capítulo 5

EJEMPLO DE UTILIZACIÓN PARA FINES DE INVESTIGACIÓN: CAMBIO DE CONFIGURACIÓN DE PISTAS

5.1. Escenario de simulación.

Se plantea un escenario ficticio para el día 25 de junio de 2017 en el Aeropuerto de Palma de Mallorca (LEPA), ya que es el tercer aeropuerto más importante en España por volumen de pasajeros y el más rentable del país, con cerca de 25 millones de pasajeros anuales totales, superado únicamente por el Aeropuerto Adolfo Suárez Madrid-Barajas y el Aeropuerto de Barcelona El Prat.

Dispone de dos pistas paralelas que pueden utilizarse simultáneamente. Existen dos configuraciones de uso preferente: la configuración Oeste (Llegadas: RWY 24L; Salidas: RWY 24R) y la configuración Este (Llegadas: RWY 06L; Salidas: RWY 06R). La configuración Oeste será preferente siempre que la componente de viento en cola no exceda de 10 kt en pista seca, o pista mojada con acción de frenado buena. Para acelerar el tránsito de llegadas la RWY 24R podrá utilizarse a iniciativa del ATC. Los pilotos que soliciten el uso de una pista distinta a la correspondiente configuración en uso deberán asumir posibles demoras significativas.

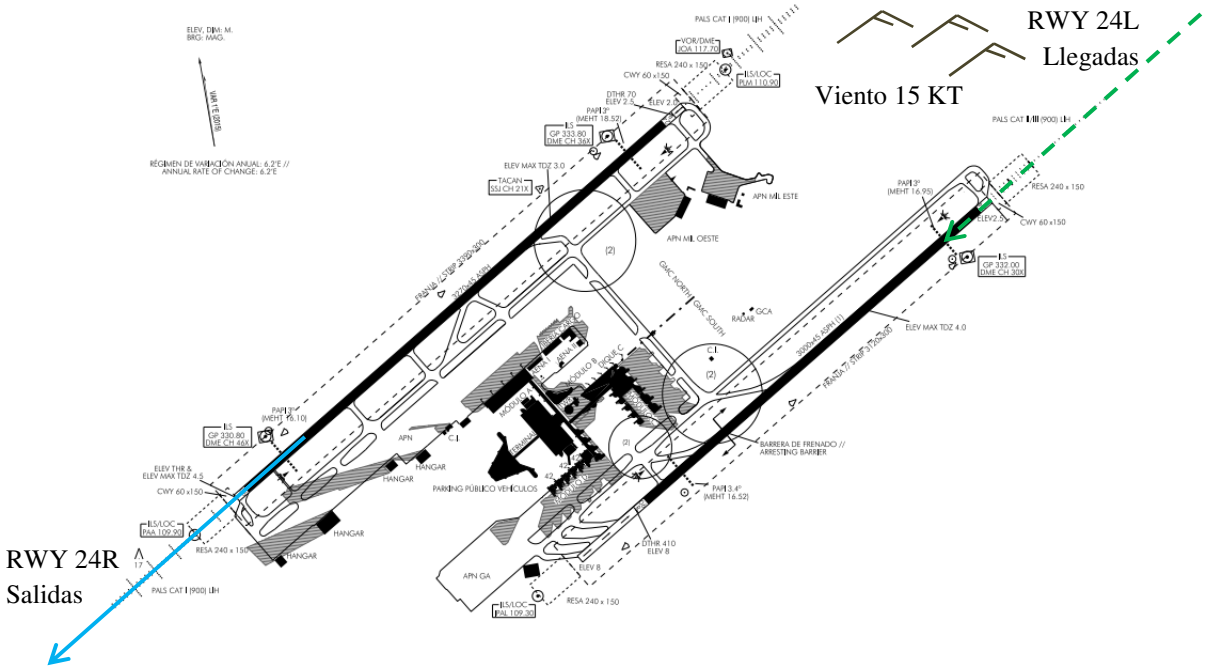


Figura 5.1 Plano de Aeródromo LEPA – Configuración Oeste.

El intervalo horario para este escenario es entre las 1400 y las 1430 UTC. La lista de vuelos de este escenario contiene 8 vuelos: 4 salidas y 4 llegadas. El aeropuerto se encuentra operando en configuración Oeste al comienzo del intervalo.

El último METAR trasmitido previo al intervalo horario para el aeropuerto es el siguiente:

LEPA 2501330Z 07008KT 9999 FEW040 30/12 Q1017

La Oficina Meteorológica del Aeródromo emite un informe especial SPECI, debido a un emperoamiento en las condiciones meteorológicas a causa de un incremento de la velocidad del viento.

SPECI LEPA 251358Z 07015KT 040V100 9999 FEW025 30/12 Q1016

La dependencia de control de Torre de Palma decide cambiar la configuración del flujo de tráficos, modificando las pistas activas, a configuración Este. Avisa a los controladores contiguos y modifica el ATIS.

La dependencia de Torre suspende los despegues y la el control de Rodadura comienza el taxi de las aeronaves hacia la nueva pista activa, ya que es más fácil bloquear salidas en tierra durante 10-15 minutos que mantener en el aire en circuitos de espera a todas las llegadas. Torre autoriza el aterrizaje de los tráficos que se encuentren en las fases de aproximación y aterrizaje, y la dependencia de Aproximación proporciona vectores al resto de llegadas hacia la nueva aproximación, una vez que el último tráfico que despegó esté lo suficientemente lejos. Cuando el nuevo flujo de llegada ha sido ordenado, se reanudan los despegues.

Este proceso necesita de la coordinación de las dependencias de Rodadura, Torre, Aproximación y Control de área, este último en el caso de que se requiera que algunos tráficos realicen procedimientos de espera (dependerá de cómo de congestionado esté el espacio aéreo).

5.2. Validación del cambio de configuración.

Para validar el tiempo necesario para la realización del cambio de configuración, se emplea el simulador desarrollado en este proyecto.

Para ello, se ejecuta el programa y se simula el escenario planteado en el apartado anterior. Una vez que tenemos situados los tráficos, comienza la simulación.

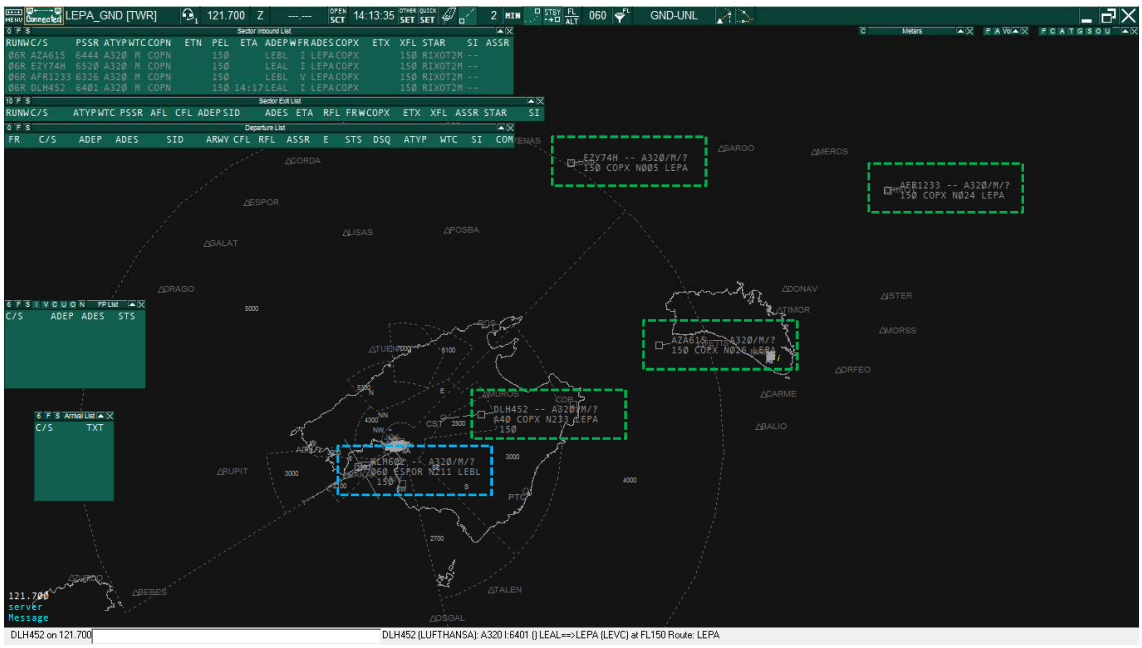


Figura 5.2 Radar EuroScope – 4 llegadas 24L y 1 salida 24R, 3 salidas en espera

Figura 5.3 Visor de tráficos.

En este momento hay cuatro llegadas a Palma de Mallorca para la pista 24L, y una salida, KLM602, que ha sido efectuada por la pista 24R. Hay tres salidas más previstas dentro del intervalo horario fijado, a las cuales se les proporciona las instrucciones para rodar hasta el punto de espera H8, de la pista 06R.

Uno de los tráficos de llegada, DLH452, se encuentra en la fase de aproximación, por lo que se le permitirá aterrizar en la pista 24L, y una vez abandone la pista se realizará el cambio de configuración.

Para ello, al resto de tráficos en llegada se les proporcionarán vectores para incorporarse a la llegada TOLSO3M hasta el fijo de aproximación inicial ADX y así poder realizar la aproximación instrumental de la pista 06L.

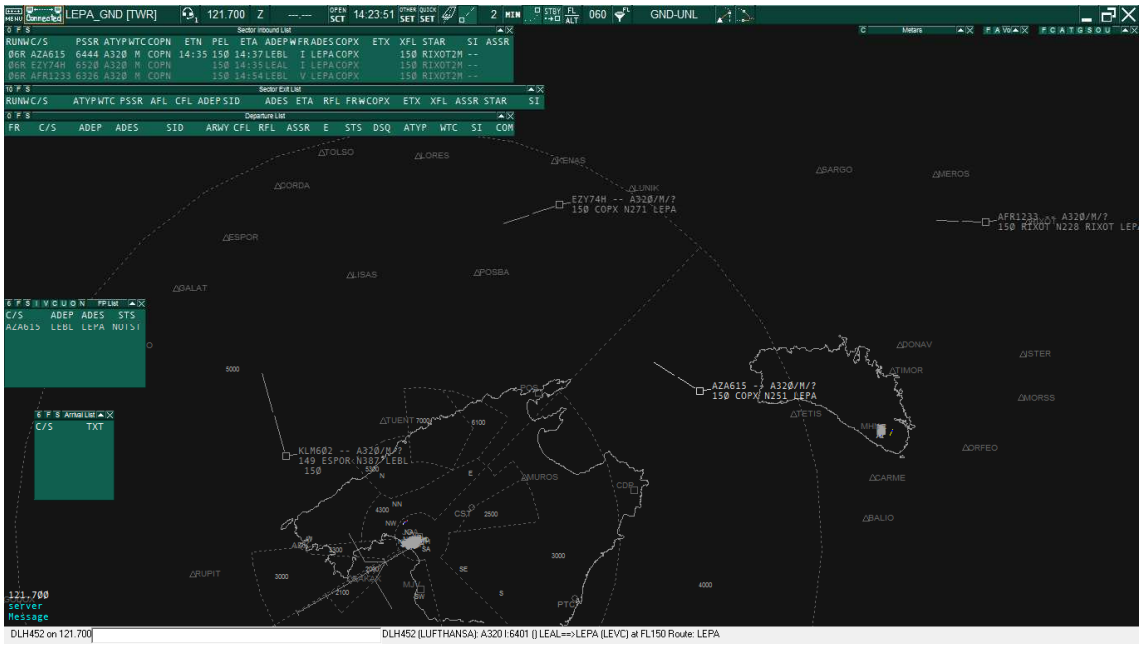


Figura 5.4 Radar EuroScope – 3 llegadas 06L y 1 salida 24R, 3 salidas en espera 06R

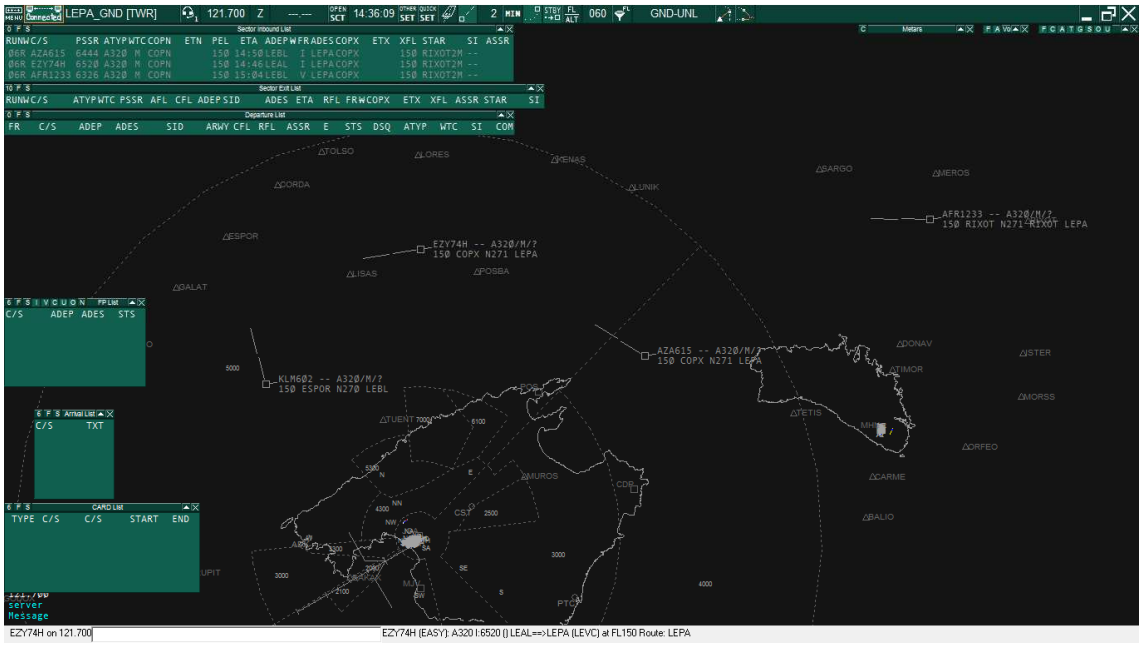


Figura 5.5 Radar EuroScope – 3 llegadas 06L y 1 salida 24R, 3 salidas en espera 06R

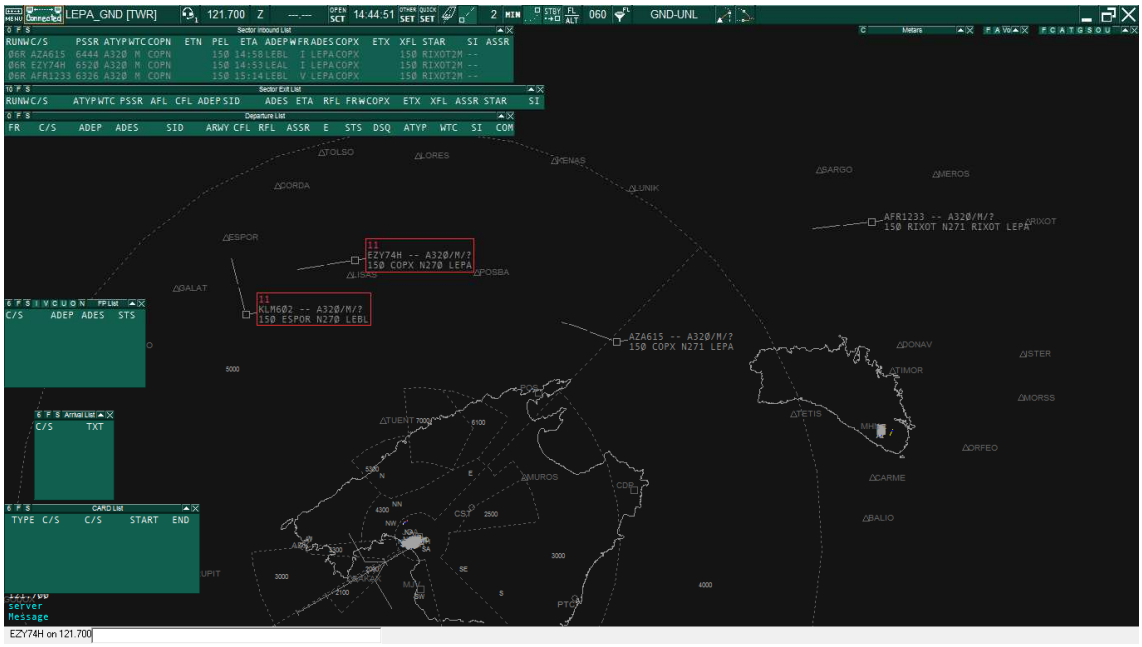


Figura 5.6 Radar EuroScope – 3 llegadas 06L y 1 salida 24R, 3 salidas en espera 06R - Conflicto

A través del escenario de simulación y la herramienta EuroScope, se pueden detectar posibles conflictos en el cambio de configuración de pistas. Además, las alertas de conflictos se puede personalizar en EuroScope.

Una vez han sido encauzadas las tres llegadas, se reanudan los despegues, ahora desde la pista 06R.

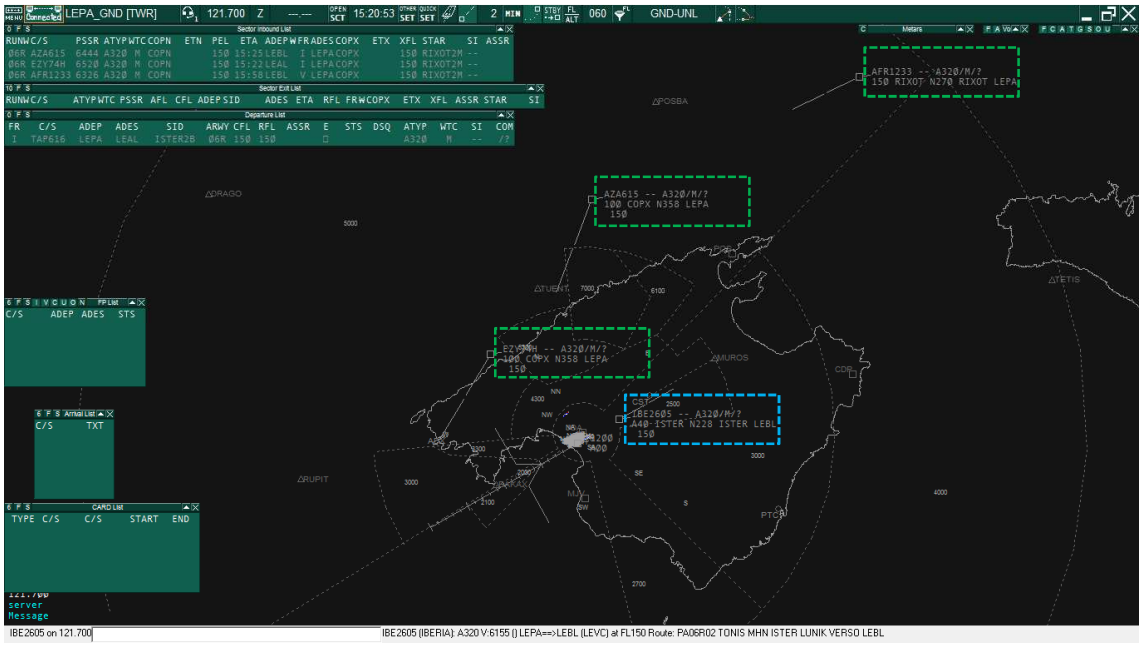


Figura 5.7 Radar EuroScope – 3 llegadas 06L y 1 salida 06R, 2 salidas en espera 06R

Se repite la simulación, realizando diferentes combinaciones posibles en el orden de las decisiones, con diferente configuración de tiempos, sobre los tráficos de este escenario para llevar a cabo cambio de configuración. De esta manera se busca la combinación que resulte en un tiempo más corto para autorizar el despegue de las cuatro salidas y autorizar el aterrizaje de las cuatro llegadas de este caso de ejemplo, sin llegar a tener conflictos entre las mismas.

Capítulo 6

CONCLUSIONES Y TRABAJO FUTURO

Este proyecto ha permitido crear un nuevo laboratorio virtual de simulación en el Laboratorio Pedro Duque de la ETS de Ingeniería del Diseño, que podrá ser aprovechado por los alumnos del Grado en Ingeniería Aeroespacial.

Asimismo, deja abierta una línea de investigación que continuará el siguiente curso. Entre las ampliaciones de este proyecto, se encuentra la de sustituir las fuentes de datos de waypoints, SID/STARs, radioayudas, extraídas por ahora del archivo de sector .SCT de EuroScope, y las restricciones de las SID/STARs, además de las aproximaciones instrumentales, obtenidas de archivos de configuración desarrollados como parte del proyecto a partir de las cartas aeronáuticas del AIP de ENAIRE, por datos de la European AIS Database (EAD) de EUROCONTROL, que consiste en un repositorio único y centralizado de información aeronáutica. Estos datos son: NOTAMS, toda la información aeronáutica publicada en los AIP (información de aeródromo, incluyendo procedimientos y obstáculos; información de rutas, espacios aéreos, radioayudas y waypoints; e información general como Organización y Autoridades), además de enmiendas, suplementos, AIC y cartas aeronáuticas.

Además, también se podría incluir trayectorias de tráfico histórico o filtrado por fecha, de la base de datos DDR2 de EUROCONTROL. Esto se podría realizar a través de otra ampliación interesante, la cual consiste en hacer posibles simulaciones en modo batch, es decir, cargar escenarios de simulación directamente a través de ficheros de configuración. De esta manera se agiliza la creación de tráficos, a la vez que permite simular situaciones concretas entre aeronaves, lo que permitiría analizar y estudiar situaciones de posibles conflictos.

Otra posible mejora sería la incorporación en el simulador de nuevos modelos de aeronaves y de sistemas aéreos pilotados por control remoto (RPAS, del inglés *Remotely Piloted Aircraft System*).

Por último, se podría ampliar el simulador para un posible estudio futuro sobre la gestión automática de autorizaciones ATC a través de comunicaciones de enlace de datos o Data link, con el simular la automatización de las órdenes de los controladores aéreos.

A nivel personal, el desarrollo de este Trabajo Fin de Grado ha sido muy enriquecedor, ya que he puesto en práctica los conocimientos de mecánica de vuelo y navegación aérea adquiridos durante el Grado. Además, ha sido la primera vez que me enfrentaba al desarrollo de una aplicación Java de mayor complejidad, ya que aparte de las conexiones TCP/IP y UDP, se ha tenido que sincronizar la interfaz de usuario con diferentes hilos de ejecución dentro de una misma instancia del programa.

Capítulo 7

BIBLIOGRAFÍA Y RECURSOS

Para el desarrollo y la implementación de este proyecto se ha consultado la siguiente bibliografía:

Del Cerro, Lluis. EUROSCOPE. MANUAL DE USO MODO PROFESIONAL. 2013.

ENAIRE. Servicio de Información aeronáutica (AIS). AIP España [en línea] <www.enaire.es>

EUROCONTROL Experimental Centre. EEC Technical/Scientific Report No. 11/03/08-08. USER MANUAL FOR THE BASE OF AIRCRAFT DATA (BADA) REVISION 3.9. 2011.

EUROCONTROL Experimental Centre. EEC Technical/Scientific Report No. 2009-009. BASE OF AIRCRAFT DATA (BADA) AIRCRAFT PERFORMANCE MODELLING REPORT. 2009.

EuroScope. Euroscope 3.1d Manual. 2010.

EuroScope. EuroScope Wiki page. 2014. < http://www.euroscope.hu/mediawiki/index.php >

ICAO. Doc. 8168 OPS/611 Operación de Aeronaves. Vol. II. Construcción de procedimientos de vuelo visual y por instrumentos. 2006.

IES Abastos. Apuntes del Grado Superior de Desarrollo de Aplicaciones Multiplataforma. 2015-2017.

Magraner Rullan, José Pedro. Apuntes de la asignatura11889 MECÁNICA DE VUELO (I) UPV (DMMT). 2015.

OpenMap. OpenMap API. 2016.

Oracle. Java SE APIs.

Vilà Carbó, Joan; Yuste Pérez, Pedro. Apuntes de la asignatura Transporte, Navegación y Circulación Aérea. 2014.

X-Plane Developer. Sending Data to X-Plane.rtfd. 2015.

Yuste Pérez, Pedro. Apuntes de la asignatura Gestión del Espacio Aéreo II. 2017.

Para la implementación del programa en Java, se han empleado los siguientes recursos:

EUROCONTROL Experimental Centre. BADA Revision 3.9. 2011.

EuroScope. Archivos de sector .SCT y .ESE. 2017.

ENAIRE. Servicio de Información aeronáutica (AIS). AIP España. Cartas. 2017.

Anexo A

MANUAL DE USUARIO DEL PROGRAMA

En este Anexo, se indican las instrucciones de uso de la aplicación. Para más información sobre las características del programa, leer capítulos de la memoria de este Trabajo de Fin de Grado - Herramienta para la simulación del sistema de control de tráfico aéreo.

A.1. Introducción.

Este simulador de vuelo consiste en una aplicación de escritorio basada en Java, que permite la realización funciones de simulación en escenarios de control tanto para la formación y aprendizaje del alumnado, así como para la investigación sobre la validación de procedimientos de vuelo y el análisis de posibles conflictos entre aeronaves en las diferentes fases de vuelo.

Este manual de usuario incluye las indicaciones necesarias para la ejecución de la aplicación y permite al usuario el aprendizaje de las funcionalidades básicas del simulador.

A.2. Requerimientos del sistema.

Para la ejecución de la aplicación Java en un escritorio de PC, solo es necesario tener instalado el paquete del entorno de ejecución Java (JRE), y actualizado a la versión más reciente, para evitar problemas de compatibilidad con la aplicación. Este paquete está disponible para sistemas operativos Linux, Mac, Solaris y Windows.

Para versiones a partir de Java 8, la memoria RAM mínima requerida son 128 MB, y el espacio en disco para JRE, 124 MB, y para Java Update, 2MB.

Tras la realización de análisis de memoria de esta aplicación, se recomienda disponer de al menos 500MB libres de memoria RAM para su correcta ejecución.

A.3. Ejecución del programa.

Para ejecutar el programa, basta con hacer doble click en el icono de la aplicación.

En primer lugar aparecerá la ventana de conexión al servidor de EuroScope (Figura A.1). Se debe introducir la dirección IP del servidor, local o remoto, al que quiera conectarse. Si se dispone de un servidor local de EuroScope en la misma máquina, para conectarnos a éste se empleará la dirección IP de *loopback*, esta es “127.0.0.1”.

Si la dirección introducida en la interfaz es válida, aparece un cuadro de diálogo que informa de ello y el programa continúa; si no lo es, avisa al usuario y espera a que éste introduzca una nueva dirección IP válida.

Figura A.1. Ventana de conexión a EuroScope

A continuación, aparece la ventana visor de tráficos (Figura A.2). En la parte superior, bajo la barra de menú, se encuentra la dirección IP del servidor de EuroScope al que se conectarán los tráficos que se creen en esta sesión.

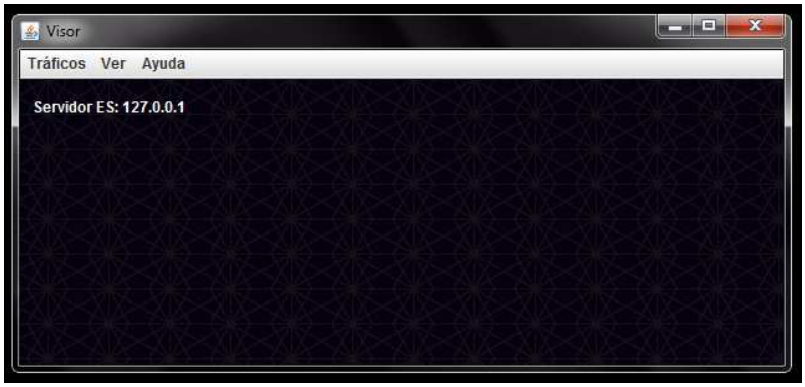


Figura A.2. Ventana visor de tráficos

La barra de menú contiene tres opciones o pestañas, Tráficos, Ver y Ayuda. Si se pulsa la pestaña de tráficos, se despliegan dos nuevas opciones: Nuevo y Desconectar. A través de la opción Nuevo, se crearán nuevos tráficos, tanto para usar el simulador basado en BADA como el simulador de vuelo X-Plane. La opción Desconectar permitirá desconectar del servidor de EuroScope, y detener la simulación en el caso de emplear el Simulador BADA, los tráficos que se seleccionen.

A.4. Creación de tráficos.

Se pulsa la opción Nuevo de la pestaña Tráficos. A continuación, aparece la ventana de conexión de un tráfico a EuroScope, en la cual se deben introducir el *callsign* de la aeronave, la identificación del usuario, la contraseña y el nombre de éste (Figura A.3).

No está permitido repetir un mismo *callsign* en la misma sesión de EuroScope. Si se da la situación, más adelante el programa desconectará el tráfico e informará al usuario de ello. La respuesta no es inmediata debido a que al servidor de EuroScope le lleva tiempo rechazar el tráfico con *callsign* duplicado.

Figura A.3. Conexión de usuario a ES

Si se cierra esta ventana, se cancelará la creación del tráfico. Si, por el contrario, se pulsa el botón "Enviar", se cerrará esta ventana y se abrirá la de selección de simulador (Figura A.4).

Las dos opciones de esta ventana son el simulador de BADA (BATS) o el simulador de vuelo X-Plane. Si se cierra esta ventana, se volverá a la pantalla anterior de conexión a EuroScope. Si se pulsa el botón de "Enviar", en función de la elección seleccionada, el programa solicitará una determinada información al usuario.

Figura A.4. Selección de simulador

En el caso de haber seleccionado el simulador X-Plane, el programa le solicitará al usuario la siguiente información:

Figura A.5. Información de conexión a X-Plane

- Dirección IP X-Plane: consiste en la dirección IP de la máquina en la que se está ejecutando el simulador X-Plane que se quiere conectar a EuroScope.
- Puerto recibe: es el puerto configurado en X-Plane para recibir datos vía UDP. Por defecto, es el 49000.
- Puerto envía: es el puerto configurado en X-Plane para enviar datos vía UDP, como la posición, la velocidad, los ángulos de Euler y el código transpondedor de la aeronave.

Antes de enviar este formulario, es necesario configurar estos puertos en el simulador X-Plane.

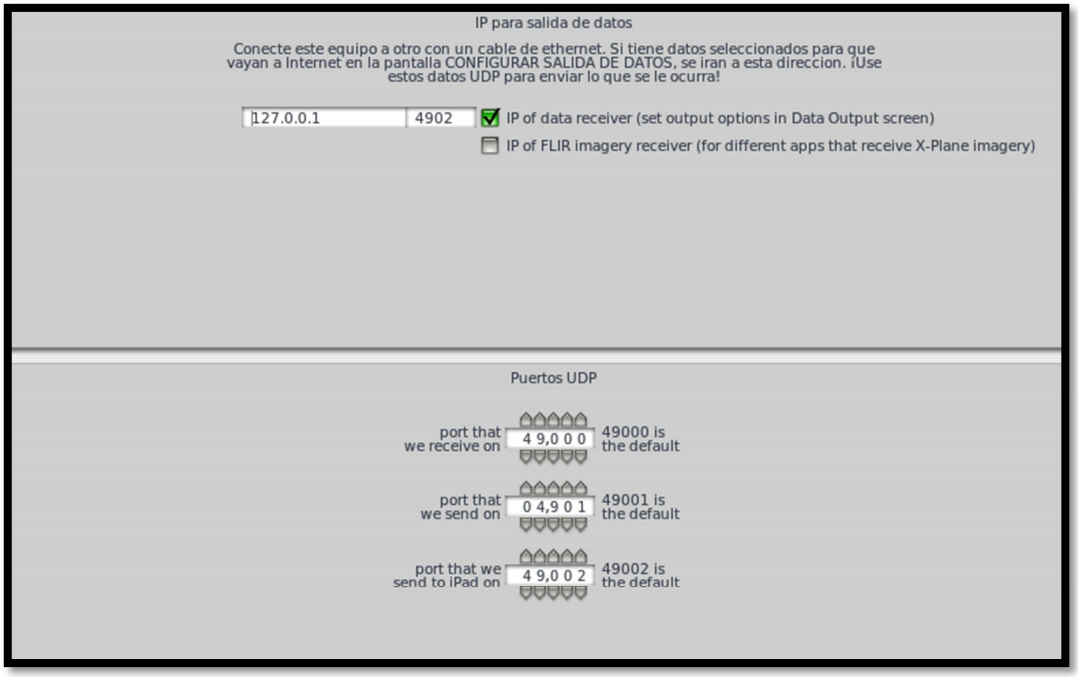


Figura A.6. Conexiones de red, Datos, X-Plane

Una vez definidos los puertos en X-Plane, y completado el formulario de Conexión a X-Plane del programa, se pulsa el botón "Enviar". El programa le enviará al simulador X-Plane, la información futura que deseará recibir de éste. Si se cierra la ventana antes de pulsar "Enviar", se volverá a la pantalla de selección de simulador. A continuación, tanto si se hubiese seleccionado el simulador X-Plane y se hubiese enviado el formulario de conexión, como si se hubiese seleccionado el simulador de BADA, aparecerá la ventana con el formulario de Plan de Vuelo (Figura A.7).

Deben completarse obligatoriamente los campos: reglas de vuelo, tipo de aeronave, masa de despegue, aeropuertos de salida, llegada y alternativo, hora de fuera calzos, duración estimada de vuelo, ruta, velocidad de crucero, nivel de vuelo, índice de coste. Asimismo, estos campos serán validados al cambiar de casilla y al enviar el formulario.

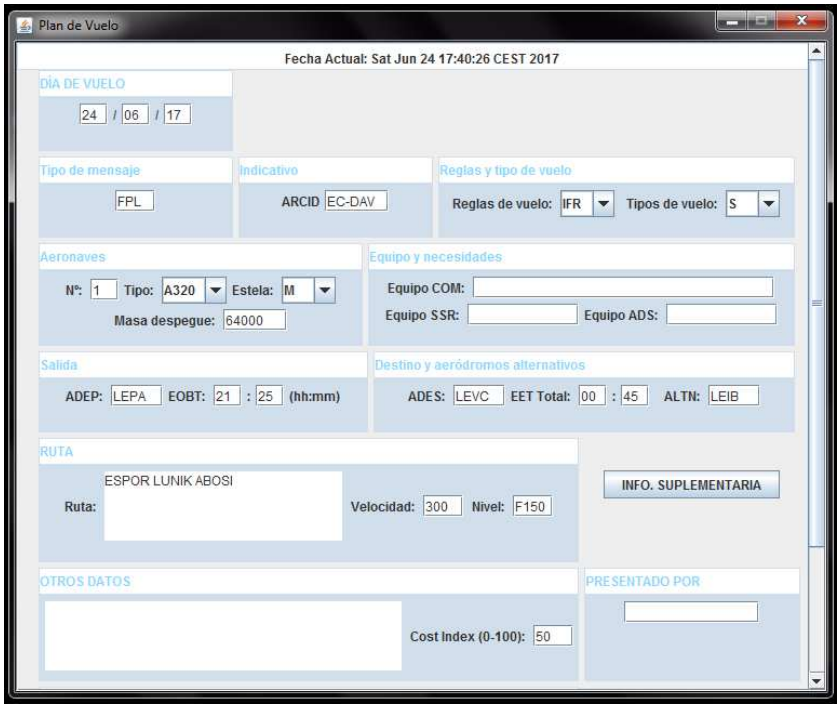


Figura A.7. Formulario de Plan de Vuelo

Si se cierra esta ventana, se vuelve a la pantalla de selección de simulador. Al pulsar el botón "Enviar", se envían datos de conexión y el plan de vuelo a EuroScope, y a continuación, se abre la ventana de selección de pistas (Figura A.8).

En ella aparecen las pistas disponibles de los aeropuertos de salida y destino elegidos en el plan de vuelo. Tras pulsar el botón "Enviar" aparecerá el tráfico en el visor de tráficos y se situará en la cabecera de la pista seleccionada. Si se cerrara esta ventana antes de pulsar el botón "Enviar", se pide

Figura A.8. Selección de pistas

confirmación a través de un cuadro de diálogo, ya que si se acepta el cierre de la ventana, se cancelará la creación del tráfico.

A.5. Funcionalidades del simulador.

Tras la conexión del tráfico, si se ha seleccionado el simulador de BADA, se podrá abrir la ventana del simulador a través del botón SIMU a la derecha de la etiqueta del tráfico en el visor de tráficos (Figura A.9).

Figura A.9. Visor - tráfico

Esta ventana contiene en la barra de título el *callsign* del tráfico al que pertenece el simulador. En la parte superior se encuentra un mapa con la aeronave situada en la cabecera de la pista del aeropuerto de salida, y la ruta que ha sido definida en el plan de vuelo. Debajo de éste, hay un panel con la posición, el rumbo y las velocidades de la aeronave.

A continuación, se encuentran los controles del simulador. Están agrupados en tres columnas. La primera se refiere a los modos de vuelo horizontales, esto es, se puede elegir entre LNAV, que seguirá la ruta del plan de vuelo, y TRK, que permite seleccionar el rumbo de la aeronave manualmente.

La selección del modo LNAV puede hacerse, pulsando el botón LNAV o bien pulsando el botón del waypoint al que se desea volar directo desde la posición actual de la aeronave. Mediante los botones + y - , se puede recorrer la lista de waypoints de la ruta. Aparecerá resaltado en color verde el waypoint hacia el que se dirige la aeronave.

Para introducir el rumbo manualmente, basta con incluir en la casilla el rumbo en grados y confirmar el modo mediante la tecla ENTER o pulsando el botón TRK.

La segunda columna permite controlar el perfil vertical del tráfico. El modo VNAV seguirá el perfil de velocidades verdadera y vertical de BADA, salvo en la fase de crucero, en la que la velocidad verdadera del avión está condicionada por el índice de coste introducido en el plan de vuelo. El modo HOLD permite mantener una determinada altitud. Por último, el modo V/S, permite introducir una velocidad vertical distinta, casilla VS, hasta alcanzar el nivel de vuelo introducido en la casilla FL. Este modo se activa tras pulsar el botón V/S, una vez rellenadas las casillas. Además, en la casilla FL aparecerá en el modo VNAV el nivel de vuelo de crucero seleccionado en el plan de vuelo. Si en la salida instrumental hay indicada una altitud o un nivel de vuelo “a mantener” salvo autorización ATC, esta altitud inicial aparecerá en la casilla FL, hasta que se alcance y se modifique por el nivel de vuelo de crucero.

La tercera columna contiene los modos throttle, es decir, ajusta el empuje del motor en función de la velocidad CAS introducida por el usuario. Para activarlo se pulsa el botón VCAS tras introducir la velocidad calibrada deseada. Para desactivarlo basta con pulsar de nuevo el mismo botón.

Debajo, se encuentra la casilla del código transpondedor, para modificarlo solo hay que introducir el nuevo código y pulsar la tecla ENTER.

Figura A.10. Simulador

Finalmente, en la parte inferior de la ventana, se encuentran en la parte de la izquierda los botones que permiten el cambio de la SID y la STAR. La primera se puede cambiar hasta antes del despegue. La segunda, hasta que se alcance el último waypoint de la ruta.

Figuras A.11 y A.12. Selección de SID y STAR, respectivamente.

Tras cambiar la SID o la STAR se actualizará la lista de waypoints del simulador.

Por último, el botón de despegar inicia el despegue de la aeronave, y el botón de pausar/reanudar permite detener el simulador en cualquier momento, salvo en el despegue.