UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Campus de Vera, Camino de Vera, Valencia, España

TRABAJO DE FIN DE MÁSTER

APLICACIÓN CONTROLADORES PREDICTIVOS PARA LA OPTIMIZACIÓN ENERGÉTICA DE EDIFICIOS

Por

Marcos Luna Fernández

21 de septiembre de 2016

MÁSTER EN AUTOMÁTICA E INFORMÁTICA INDUSTRIAL. Curso 2015-2016

Tutor : Javier Sanchís Sáez

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Resumen

En el presente proyecto, se ha diseñado un control predictivo de una máquina de climatización HVAC con el fin de optimizar el consumo del mismo durante los meses de verano. Para tal fin, se ha empleado el software de diseño energético EnergyPlus junto a DesignBuilder para construir un modelo base que nos servirá como real. A partir de la creación de un entorno de Co-simulación entre EnergyPlus y Matlab con el toolbox MLE+, se ha identificado dicho modelo para simular la temperatura operativa junto al consumo, utilizando un diagrama de bloques Simulink o el comando lsim, y posteriormente emplear un Algoritmo Evolutivo con el que se ha minimizado el consumo en base a la selección de los setpoints horarios futuros y el estudio del comportamiento del sistema según las predicciones obtenidas.

1.2. Motivación

Con el diseño del control predictivo, se pretende abordar un tema hasta ahora nunca visto en mis estudios. Se trata de un tipo de control alejado del clásico y que se ha desarrollado sobre todo en otros campos más que en el de control de temperatura. Las nuevas tecnologías y las precisiones en las predicciones meteorológicas hacen completamente factibles implantar el control predictivo con el fin de optimizar el consumo de lo que puede ser un radiador, aire acondicionado o un sistema de climatización compacto HVAC.

El control clásico únicamente nos permite seleccionar distintas consignas en función de la realimentación del usuario. No obstante, lo que para una persona puede ser calor, para otra es frio y viceversa. Normalmente en las tecnologías modernas se instala un termostato dual que soluciona este problema seleccionando las consignas y estableciendo un control por banda (fijar una temperatura mínima y una temperatura máxima), pero esto lleva a un incremento del coste en el consumo en comparación con dejar la máquina constante en un valor medio o inferior al dado por banda.

Por tanto, el control predictivo resulta ideal para este tipo de sistemas, ya que nos permite adelantarnos a la tendencia térmica del edificio y actuar en consecuencia a ello, reduciendo a priori el consumo de la o las maquinas instaladas en la zona. No obstante, en el cálculo de la temperatura operativa de una zona intervienen numerosos factores como puede ser la temperatura del bulbo seco, el punto de condensación, la presión, la radiación solar, la radiación difusa, la nubosidad, etc.… Esto crea un sistema complejo y difícil de controlar, por lo que se hace necesario y preciso realizar una correcta identificación del sistema con el objeto de que al menos se sigan las tendencias de temperatura y consumo si bien se puede tener un margen de error más amplio que en otro tipo de controles.

Además de lo emocionante que resulta tocar temas no vistos hasta ahora junto un campo completamente nuevo (el campo de la energía), la realización de un proyecto poco común y la mejora del sistema por parte de uno mismo son suficientemente motivos para embarcarme en un proyecto de tal calibre.

Para finalizar, el proyecto es de carácter experimental, por lo que los resultados se deberán evaluar sacando una serie de conclusiones por los que se han originado. Como todo experimento, se realizarán diferentes pruebas con distintas condiciones, y se implantarán cada vez controles más complicados o correcciones en las pruebas. El razonamiento de dichos puntos desencadenará en una resolución final que será descrita al finalizar el proyecto.

1.3. Objetivos

La realización del proyecto de control predictivo sobre una máquina HVAC tiene como objetivo principal demostrar las ventajas del uso de predicciones meteorológicas para simular el comportamiento del modelo en las próximas horas y asignar las consignas de temperatura en base a las mismas, todo ello en detrimento de un control clásico ‘encendido/apagado’ o de selección de consigna de temperatura que no tienen en cuenta dichos parámetros.

La idea principal enfoca la selección de consignas para obtener el comportamiento del modelo en el horizonte de predicción y en base a ello optimizar el consumo de salida para reducir el gasto energético y por causa-efecto el impacto económico y ambiental.

El sistema a controlar posee 3 entradas: Temperatura ambiente, radiación solar y consigna de temperatura (dividido en calor y frio), aunque únicamente se controlará la consigna de frío, ya que solo determinaremos el comportamiento para la época de verano. La simulación se realizará con el uso de ficheros climáticos que definen el comportamiento del clima de una zona en concreto (se ha seleccionado el clima de Valencia como referencia). La idea es emplear las predicciones meteorológicas en tiempo real y aplicarlas al control del sistema, aunque eso es algo que no se ocupa en el presente proyecto.

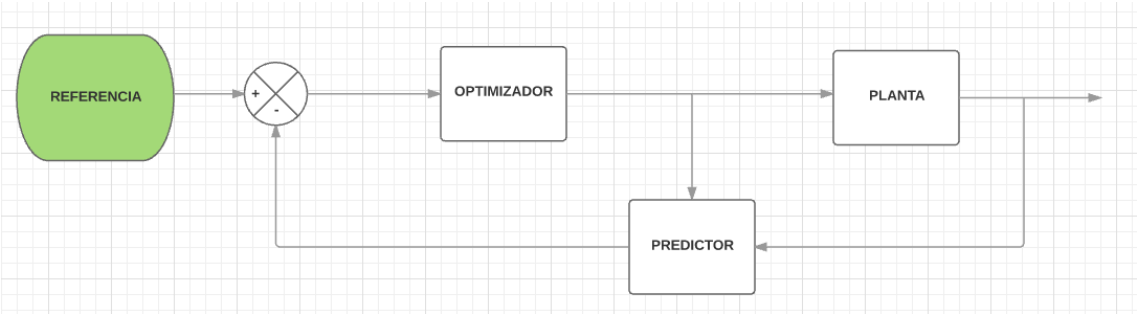

Figura 1. Diagrama básico de funcionamiento

Para el desarrollo del control, se ha de seguir una serie de puntos previos para obtener el entorno de trabajo:

- Creación de un modelo físico que se usará como modelo real, empleando el software DesignBuilder y exportándolo a formato de EnergyPlus.
- Elaboración de un entorno de cosimulación entre EnergyPlus y Matlab por medio de la 'ExternalInterface' y el uso del toolbox MLE+.
- Control del modelo real eligiendo las consignas para los diferentes pasos de tiempo.

Una vez creado dicho entorno, se procederá a la identificación del sistema:

- Modificación de los archivos climáticos de simulación para crear señales de tipo escalón y ver cómo afectan al sistema las 3 entradas de manera individual, tanto para la salida de temperatura operativa del modelo, como para el consumo de la máquina de clima (siempre y cuando este encendida).
- Extracción de los datos para trabajar en Matlab.

- Uso del toolbox de identificación para obtener la función de transferencia de cada una de las entradas con sus respectivas salidas.
- Creación de un modelo Simulink o uso de lsim que pretende sustituir al real para poder emplear sobre él el control predictivo.

La parte final del proyecto consiste en la creación del control predictivo:

- Programación del algoritmo genético básico obtenido del AI2 para emplearlo sobre el modelo creado.
- Empleo de una función de coste.
- Realización de pruebas bajo diferentes condiciones para observar los resultados obtenidos en el consumo manteniendo unos límites de temperatura.
- Obtención del control final.

2. ANTECEDENTES

2.1. Control Predictivo

El control predictivo por modelo (Model Predictive Control – MCP) es un tipo de control desarrollado en la década de los 70-80s para tareas y procesos industriales.

Originalmente se introdujo en 1976 mediante una patente presentada por Juan Manuel Martín Sánchez (Catedrático de Universidad) denominada 'Control Predictivo' y 'Adaptativo Predictivo'. Dice así:

"El sistema de control adaptativo predictivo de la presente invención actualiza en tiempo real los parámetros de un Modelo Adaptativo-Predictivo a partir del cual se predice el vector dinámico de salida del proceso que está siendo controlado y el vector de control que controla la operación de este proceso es calculado con el objetivo de que el vector dinámico de salida predicho se haga igual al vector dinámico de salida deseado." [1]

Más adelante se empleó para la industrial de procesos en refinerías o plantas de tratamientos químicos, así como en sistemas de potencia. En la actualidad se encuentra en diversas áreas dentro del campo de ingeniería (robótica, automática, metalúrgica, etc…)

Consiste en un tipo de control basado en predicciones de las variables del sistema a controlar por las cuales se genera un comportamiento deseado en base a un modelo definido previamente. Se emplea principalmente para resolver comportamientos dinámicos complejos, sistemas MIMO y procesos inestables, aunque puede ser aplicado con otros fines.

El control predictivo por modelo se basa en la optimización iterativa y de horizonte finito. En un tiempo t de muestreo se calcula el estado actual de la planta en base a una estrategia de control que minimice el costo de una función (gracias a un algoritmo de minimización como por ejemplo un algoritmo evolutivo). Por tanto se explotan diferentes trayectorias y se elige la más 'óptima'. El horizonte de predicción depende de los estados anteriormente calculados por lo que al control predictivo se le denomina también **'control de horizonte errante'** ya que siempre va hacia adelante.

Se emplea un modelo matemático que representa de la manera más real posible al sistema a controlar, al cual se le denomina modelo de predicción. Gracias a ello, se

puede predecir el comportamiento de las variables a controlar (salidas del sistema) durante un límite u horizonte fijado por el operador, calculando para ello las variables manipuladas futuras con el fin de que converjan en los valores fijados de referencia.

Los valores asignados a la variable manipulada corresponden a una optimización según el criterio designado por el operador. Dicho criterio a optimizar se le denomina función de coste, en el cual sus variables de cálculo son determinadas por el modelo predictivo. A la salida más óptima se le denomina 'fitness'.

Para que la discrepancia entre el modelo real y el modelo predictivo sea lo más robusta posible, se precisa además de una correcta identificación del sistema, una realimentación de manera que se resuelva la predicción en un determinado intervalo de tiempo, se realimenta al sistema y se calcula de nuevo. De esta manera el intervalo se va deslizando a lo largo del tiempo (control errante).

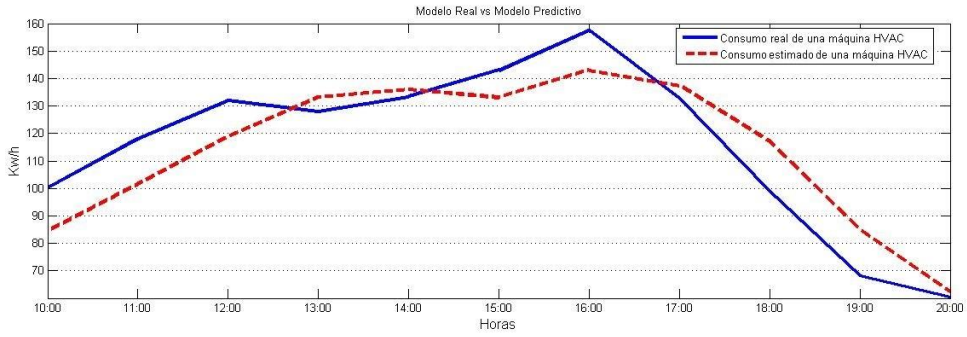

Figura 2. Ejemplo de predicción de consumo de una máquina HVAC

Como se observa existen ligeras diferencias entre uno y otro modelo, debido principalmente a la precisión de la identificación y las herramientas de simulación empleadas. También se ve cómo se siguen las tendencias. Este error puede afectar a la hora de seleccionar los límites de trabajo en la optimización del proceso, por lo que se ha de tener en cuenta.

2.2. Algoritmo Evolutivo

Los algoritmos evolutivos son herramientas de optimización y búsqueda de soluciones en base a una serie de criterios siguiendo la evolución biológica. A veces

existen problemas que no se pueden resolver vía métodos tradicionales y en los cuales el empleo de un algoritmo evolutivo (abreviado AE) resulta ser una solución a éstos.

La historia de los AE se remonta a la década de los 60s, donde Jonh Holland presentó la posibilidad de introducir mecanismos de la biología evolutiva en el campo de Inteligencia Artificial. No obstante todo quedó de manera teórica, siendo implementado más adelante ya que las técnicas de entonces no eran suficientes para dicha labor. Esto dio lugar a un método de optimización como técnica no convencional y aplicado al ámbito real. De los AE surgieron otras técnicas como son: **Algoritmos Genéticos** (Goldberg), Programación Genética (Koza), estrategias de evolución (Rechenber/Schwefel), etc…. Los más extendidos son los Algoritmos Genéticos (abreviado AG).

Koza definía los Algoritmos Evolutivos de la siguiente manera:

"Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud. "[2]

Los algoritmos genéticos son un tipo de algoritmo evolutivo basado en los principios darwinianos de evolución genética y selección natural. Es decir, según una metodología a seguir, se seleccionaría el individuo más apto en detrimento del resto, debido a que poseen una mayor tasa de supervivencia (tanto de longevidad como de reproducción). Estos individuos poseen un código genético que es heredado por sus hijos, los cuales de nuevo pasaran por un proceso de selección del más apto (de esta manera según el número de individuos y generaciones se obtendría una población más óptima.

Gracias al aumento de las prestaciones y carga computacional de los ordenadores durante la época de los 80s, se pudo implementar los algoritmos genéticos para resolver problemas hasta entonces inabordables. De esta forma, comenzaría el desarrollo de una nueva área de estudio en el campo de Inteligencia Artificial. Sus principales aplicaciones van desde el reconocimiento de patrones, cálculo de estrategias de marketing y mercado, diseño de circuitos (seleccionar el enrutamiento

más óptimo), ingeniería aeroespacial, química, programación a bajo nivel, videojuegos y un sinfín más de campos en los que se precisa procesos de optimización.

El funcionamiento es similar al principio biológico de selección natural ya descrito. Los AG trabajan con poblaciones de individuos, que no son más que posibles soluciones al problema planteado. Partiendo de una población inicial generada aleatoriamente, se somete a cada uno de ellos a una función de aptitud que evalúa lo buen candidato que resulta dicha solución. El algoritmo genético buscará una solución que se desconoce, por lo que se emplearán diferentes técnicas para detenerlo cuando se encuentre una solución cercana a lo deseado, se cumpla un número máximo de iteraciones, no haya cambios en la población más 'apta' entre diferentes generaciones, etc…

Para estudiar a las diferentes poblaciones antes se les aplica una serie de procesos:

- Selección de los mejores: En este caso, una vez evaluada la aptitud de cada individuo, se selecciona un número finito como candidatos para ser cruzados entre ellos.
- Cruce: El cruce se produce entre dos individuos que darán lugar a dos descendientes los cuales poseerán características de ambos progenitores (sus cromosomas).
- Mutación: Al igual que en los procesos biológicos, se añade un parámetro de mutación para alcanzar y aumentar el espacio de búsqueda de la población actual (lugares a los que no se podía llegar).
- Elección: Se elige a los mejores candidatos para que pasen a la generación siguiente.

El proceso del AG es iterativo. Esto quiere decir que para un grupo de población, se le someterá a los procesos descritos arriba generando un par descendiente por cada par de individuos. Siempre se escoge al más apto (dentro de lo posible). El algoritmo repetirá el proceso para la nueva generación, terminando cuando el usuario elija el criterio de finalización deseado.

```
Iteration: 4
   xmin: [19.5 0 22.4] -- f(xmin): 39.2533
-----------------------------------------------
Iteration: 5
   xmin: [19.5 0 22.5] -- f(xmin): 38.8379
-----------------------------------------------
######   RESULT   #########
   Objective function for xmin: 38.8379
   xmin: [19.5 0 22.5]
-----------------------------------------------
-----------------------------------------------
```

Figura 3. Ejemplo de iteraciones y resultado final

Debido al carácter aleatorio del propio algoritmo, es completamente normal y razonable no obtener el mismo resultado cuando se ejecute más de una vez, teniendo en cuenta que la población inicial se eligen de manera aleatoria y las mutaciones afectan en mayor o menor grado a diferentes miembros. También es cierto que a mayor número de individuos y de generaciones de los mismos el resultado mejorará a no ser que la función converja en un valor.

3. METODOLOGIAS Y HERRAMIENTAS

3.1. EnergyPlus

EnergyPlus es un potente software de simulación energética de edificios, naves, fábricas…. Desarrollado por la DOE (Departamento de Energía, Estados Unidos). Se emplea principalmente para realizar estudios energéticos de demanda y consumo, altamente solicitados hoy en día, ya sea por rendimiento y eficiencia energética como para cumplimentar la normativa medioambiental.

El programa realiza un modelo por ordenador que recoge todas las variables que afectan directamente al cálculo térmico de un edificio (cargas térmicas, HVAC, calderas, cerramientos, número de pisos etc…). El modelo se somete a una simulación energética, metiendo a todas estas variables en la misma ecuación (sistema complejo). Se realiza un análisis a partir de los resultados obtenidos, y estos a su vez son empleados para evaluar el coste energético de mantenimiento, permitiendo de esta manera realizar estrategias para minimizar su impacto según las exigencias requeridas.

EnergyPlus integra distintos programas en uno solo, que trabajan de manera simultánea para calcular las condiciones energéticas del edificio. Para ello se ayuda de exponer al modelo a las diferentes condiciones meteorológicas que nos podemos encontrar en la vida real a partir de unos ficheros climáticos que simulan el comportamiento climático de una zona en concreto (lo más aproximado posible).

ArranqueConsumo.csv

	W+/S-}	Longitude {E+/W-}	TimeZone {+/- GMT}	Elevation {m}	Column 6
0001		-0.47	1.0	62.0	
0002	esign Condition		Design Stat	ColdestMonth	DB996
0003			Units	{}	{•C}
0004	sign Data 2009 ASHRAE Handbook		Heating	1	1
0005	e	Period Type	Period Start Day	Period End Day	<repeat to # periods>
0006	ek Nearest Max Temperature For Period	Extreme	8/26	9/ 1	Summer - Week Nearest Average Temperature For
0007	erature Depth {m}	Soil Conductivity {W/m-K}	Soil Density {kg/m3}	Soil Specific Heat {J/kg-K}	Jan {C}
0008					10.35
0009	aving Start Date	Daylight Saving End Date	Number of Holidays	Holiday Name	Holiday Date
0010		0	0		
0011					
0012					
0013					
0014					
0015	ntervals per Hour	DP Name/Description	DP Start Day of Week	DP Start Day	DP End Day
0016		Data	Sunday	1/ 1	12/31
0017		Datasource	DryBulb {C}	DewPoint {C}	RelHum {%}
0018		Datasource	Dry Bulb Temperature {C}	Dew Point Temperature {C}	Relative Humidity {%}
0019		C9C9C9B8*0?9?9?9?9?9?9A7A7B8B8A7*0*0E8*0*0	5.0	3.6	91
0020		C9C9C9B8*0?9?9?9?9?9?9A7A7B8B8A7A7*0E8*0*0	5.2	4.2	93
0021		C9C9C9B8*0?9?9?9?9?9?9A7A7B8B8A7A7*0E8*0*0	5.0	4.3	95
0022		C9C9C9E8*0?9?9?9?9?9?9A7A7A7A7A7*0E8*0*0	4.6	3.9	95
0023		C9C9C9B8*0?9?9?9?9?9?9A7A7B8B8A7A7*0E8*0*0	4.1	3.3	95
0024		C9C9C9B8*0?9?9?9?9?9?9A7A7B8B8A7*0*0E8*0*0	3.5	2.7	95
0025		A7A7E8E8*0?9?9?9?9?9?9A7A7A7A7A7*0E8*0*0	3.6	2.6	93
0026		A7A7E8B8*0?9?9?9?9?9?9A7A7B8B8A7A7*0E8*0*0	4.0	3.0	93
0027		A7A7E8B8*0H9H9H9I9I9I9I9A7A7B8B8A7A7*0E8*0*0	4.0	2.0	87
0028		A7A7E8B8*0H9H9H9I9I9I9I9A7A7B8B8A7A7*0E8*0*0	5.0	3.0	87
0029		A7A7E8B8*0H9H9H9I9I9I9I9A7A7B8B8A7A7*0E8*0*0	7.0	3.0	76
0030		A7A7E8B8*0H9H9H9I9I9I9I9A7A7B8B8A7A7*0E8*0*0	11.0	3.0	58

Figura 4. Estructura interna de un fichero de clima .epw (formato CSV)

A estos ficheros se les denomina archivos de clima con extensión **'EPW'.** En realidad consisten en archivos .CSV (tablas de datos separadas por ;) adaptados al formato de EnergyPlus, el cual computará los datos reflejados en ellos para simular el clima y como afecta este al modelo en concreto (*Véase Figura 4).*

Los cálculos energéticos se realizan en régimen transitorio, donde se emplean funciones de transferencia de calor para calcular la conducción de las superficies que forman el edificio. Además, tiene en cuenta el flujo térmico entre diferentes zonas o con el exterior. El paso de simulación (TimeStep) puede ser ajustado, así como el periodo de simulación. No obstante, este último solo puede ser seleccionado de manera diaria, siendo el mínimo exigido de simulación 24 horas y no pudiendo partir la simulación en una hora en concreto (simula días completos).

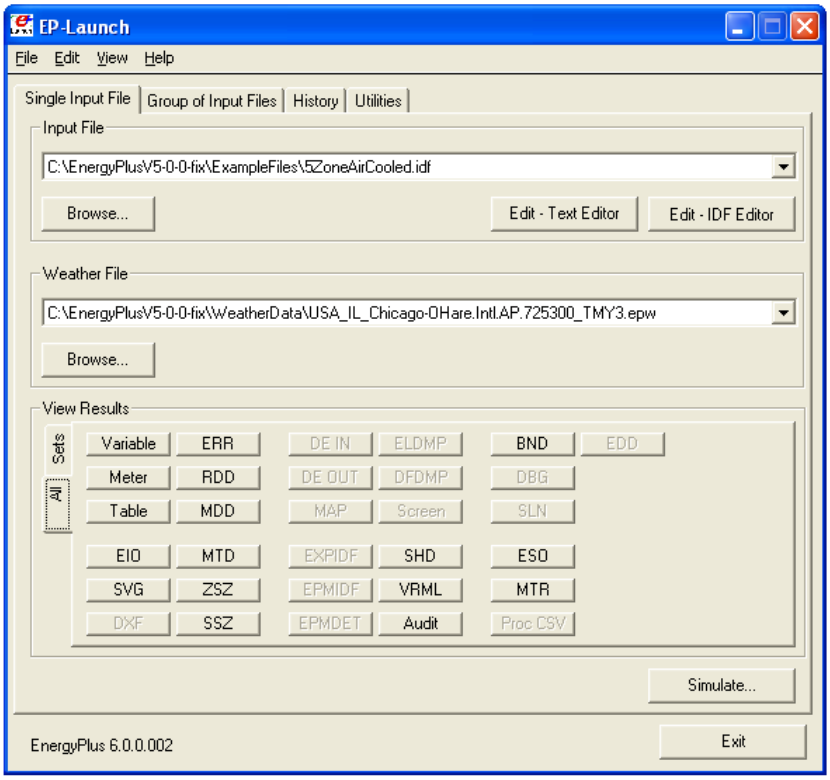

Figura 5. Interfaz básica de EnergyPlus

Los ficheros creados por EnergyPlus tiene un formato denominado *'IDF'* que no es más que un formato de texto donde se recoge el diseño del modelo físico (cerramientos, materiales, etc…), la máquina de climatización, ventilación, combustibles, variables de salida y un sinfín más de parámetros los cuales no corresponden con el tema a tratar en el trabajo.

Normalmente, se suele emplear software adicional para la creación del modelo y simular un control sencillo con diferentes máquinas HVAC. Entre los programas de mayor uso junto a esta plataforma nos encontramos con OpenStudio (emplea el motor de SketchUP) y el DesignBuilder.

3.2. Matlab

La plataforma de trabajo Matlab está diseñada para solucionar problemas matemáticos que se presentan en el día a día de un ingeniero. Emplea un potente lenguaje matricial capaz de resolver problemas de todo tipo, denominado lenguaje **M**.

Matlab integra dentro de su red numerosos toolboxes de apoyo para desarrollar tareas específicas (procesado de señales e imágenes, simulaciones, desarrollo de redes neuronales, etc…). La mayoría de ellos cuenta con su propia interfaz gráfica que trabaja en el lenguaje de Matlab. Resulta de gran utilidad disponer de dichas herramientas no solo a nivel de GUI, sino también a nivel de comandos, de manear que se puedan realizar interacciones entre distintos tipos de toolboxes desde la propia consola de comandos de Matlab.

Dentro de sus prestaciones se encuentra el manipulado de vectores y matrices, empleo de funciones, procesado y representación de datos, implementación de algoritmos, comunicación con otros entornos y lenguajes de programación. Se emplea principalmente en tareas de investigación y desarrollo.

Además, Matlab nos permite crear scripts con los cuales se pueden realizar diferentes tareas como llamar a otras toolboxes, funciones, cambiar parámetros, cargar archivos de datos, y todo ello de manera automática.

El lenguaje M emplea programación orientada a objetos muy similar al lenguaje C/C++ solo que basa su poder en el cálculo matricial. Se puede construir funciones con variables, bucles de control, comandos especiales para funciones matemáticas, etc….

La interfaz de Matlab dispone de diferentes ventanas siendo la principal la consola de comandos o **‘Workspace’.** De manera opcional contamos con ventanas de gran ayuda como la de selección de directorio (para ver donde estamos, fundamentalmente para ejecutar funciones), ver los objetos y clases creados y un historial de comandos.

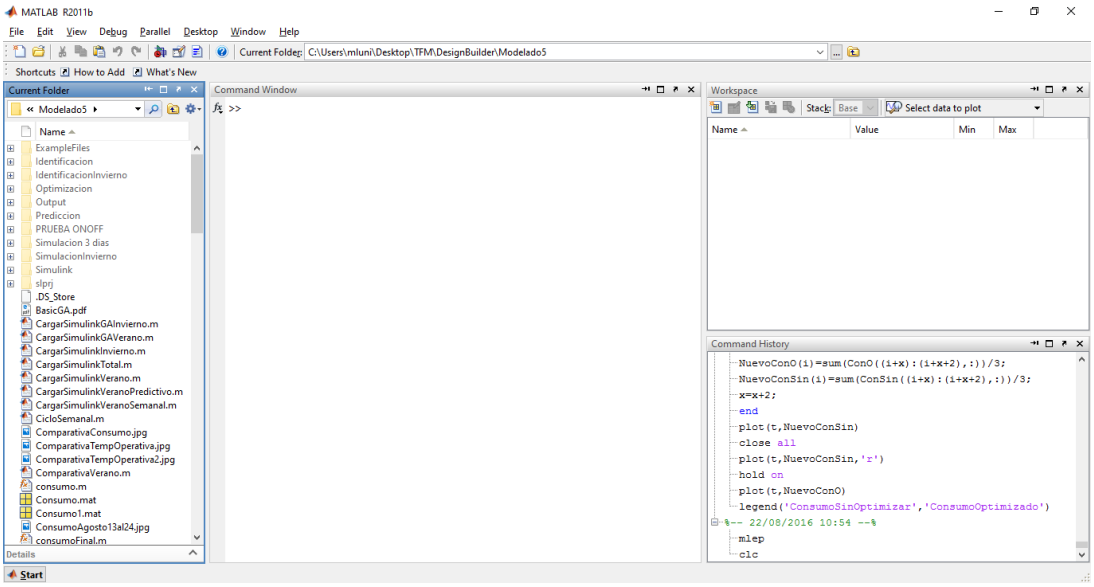

Figura 6. Interfaz Matlab v2011b

3.3. Simulink

Simulink es un entorno de trabajo desarrollado en Matlab que emplea bloques para realizar simulación de procesos. Por defecto suele venir con el programa principal, ya que constituye una herramienta muy utilizada y de gran desarrollo en los entornos de simulación. No solo se relaciona con Matlab sino que también posee bloques que se adaptan con otros programas o lenguajes (por ejemplo un conversor a código C/C+).

Gracias a su propia GUI, se pueden arrastrar los diferentes bloques personalizables de un sinfín de librerías incorporadas en el mismo, unidos entre sí mediante líneas. A su vez, se pueden crear subsistemas con el fin de organizar y simplificar los diferentes bloques (división entre principal y secundarios o subfunciones), pudiendo crear un bloque con x entradas e y salidas.

Simulink es capaz de tomar datos tanto de los bloques propios de la librería como de variables encontradas en el workspace de Matlab, pudiendo además exportar los resultados al mismo para trabajar o llevar a cabo análisis sobre ellos. También existe una serie de bloques específicos que trabajan directamente con la línea de comandos.

Dentro de la propia interfaz se puede comprobar los resultados obtenidos, seleccionar el paso de simulación, emplear bloques matemáticos, funciones de transferencia. En ocasiones resulta más sencillo elaborar un diagrama en Simulink y llamarlo directamente en Matlab que programar todo el código para ello, por lo que resulta una herramienta muy útil en el día a día de un ingeniero. Existen diferentes modos de simulación, siendo el empleado por defecto el **'Rapid Accelerator'.**

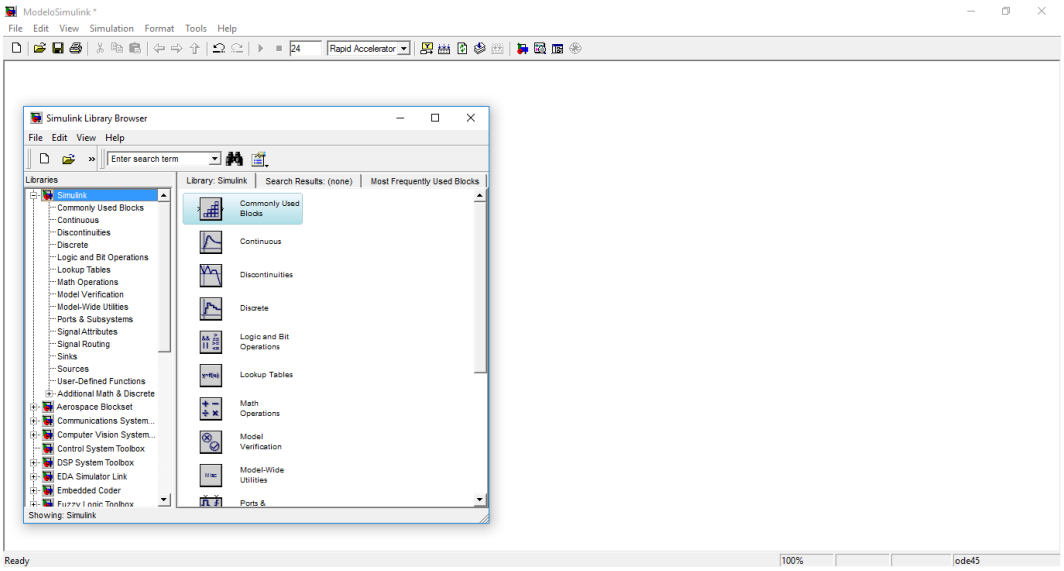

Figura 7. Interfaz Simulink

3.4. System Identification Toolbox

Este toolbox es una herramienta que construye modelos matemáticos de sistemas dinámicos para ser empleados posteriormente en Matlab o Simulink. Admite múltiples formatos de entrada para la identificación de un sistema, siendo necesario contar al menos con una entrada y una salida a esa misma. El formato de entrada creado (también puede ser cargado externamente) es el **'Iddata'** mientras que el formato de salida de las funciones de transferencia creadas se exportan como ficheros **'idproc'**. Invocando a los diferentes parámetros de la FDT se puede obtener los valores de la ganancia, delay, etc.…

No solo se pueden identificar sistemas lineales, sino que también permite crear modelos no lineales a partir de la estimación de los parámetros de la función de transferencia que relacione la entrada con la salida. Emplea diferentes métodos de búsqueda y estimación de los parámetros en función del sistema a identificar. Entre los métodos de estimación lineales se encuentra la reducción al mínimo de predicción de errores (PEM), la de máxima verosimilitud y la identificación del sistema subespacio.

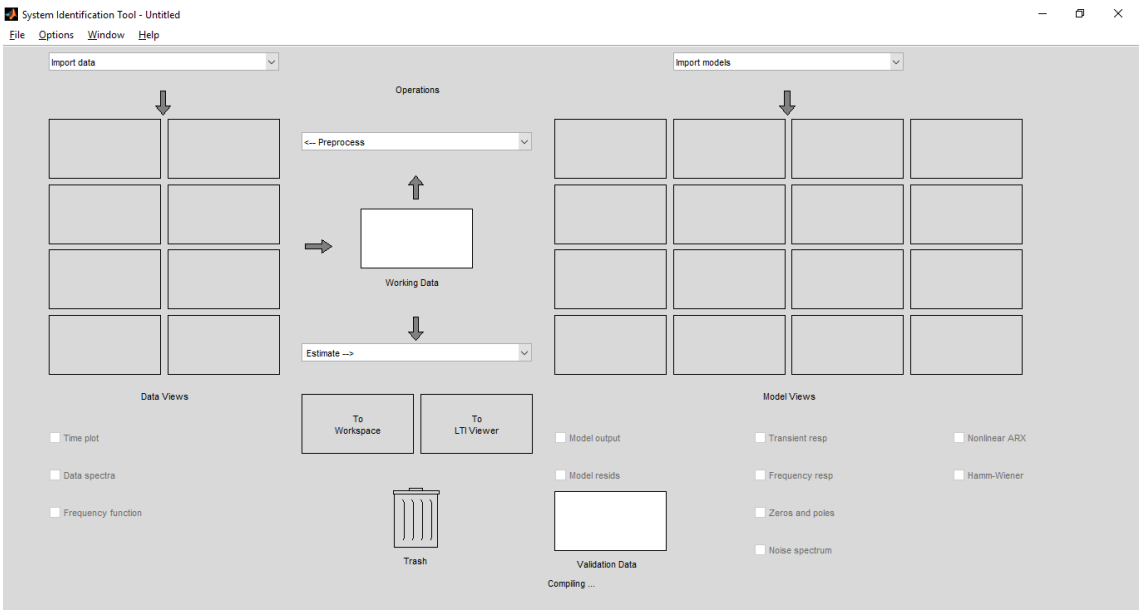

Figura 8. Interfaz System Identification Tool

Para sistemas no lineales, emplea otras técnicas como la estimación de modelos Hammerstein-Wiener y modelos ARX

3.5. Matlab EnergyPlus (MLE+)

Un grupo de investigadores ha desarrollado un toolbox que permite el uso del software EnergyPlus en Matlab, pudiendo interactuar entre uno y otro de manera sencilla. Se extraen variables de salida de EnergyPlus (como medidas) y se introducen variables de entrada de Matlab para modificar parámetros a lo largo de la simulación, previa configuración del archivo **'IDF'.**

Este paquete de trabajo incluye una interfaz gráfica en la cual se permite desarrollar los pasos necesarios para crear el entorno de simulación. Hace falta cargar las variables de entrada a EnergyPlus y las salidas del mismo y posteriormente emplear un fichero de control desarrollado en lenguaje M para realizar los cambios deseados en el sistema (básicamente aplicar un determinado tipo de control).

Seleccionando el archivo IDF y el fichero de clima correspondiente, se pueden simular los resultados los cuales pueden ser exportados al Workspace de Matlab, para trabajar con ellos.

El toolbox también cuenta con otras herramientas como aplicar un control sin necesidad de programarlo o identificar un sistema por medio de escalones, además de una función de conexionado BACnet empleado para conectar la máquina de climatización con los diferentes componentes del edificio y estos entre sí.

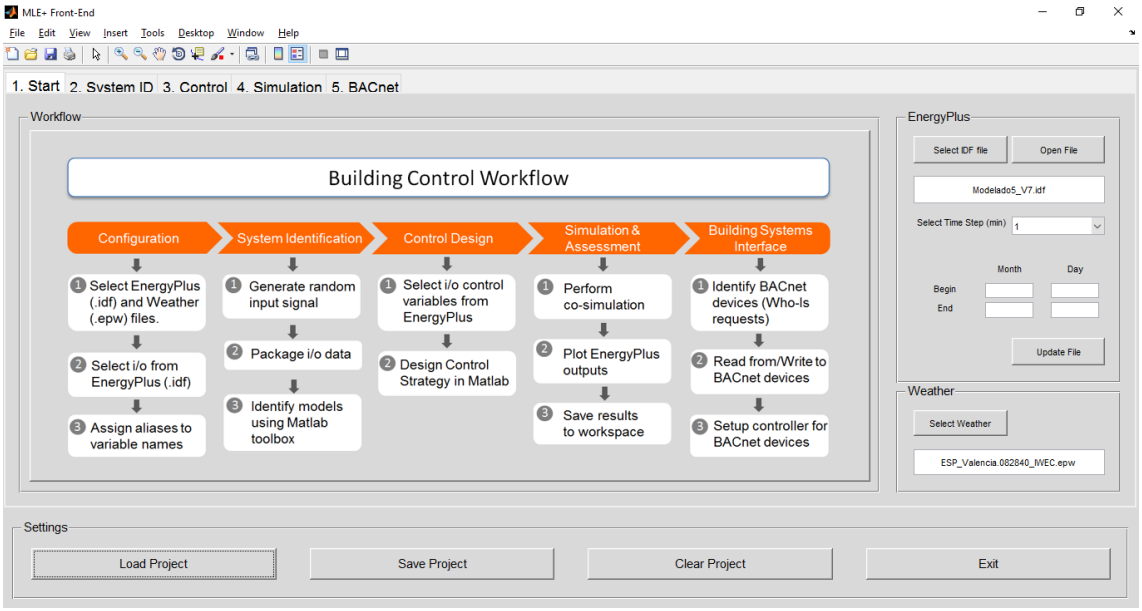

Figura 9. Interfaz MLE+

3.6. DesignBuilder

Para facilitar el diseño de un modelo basado en EnergyPlus, existen software externos que nos permite construir un edificio y seleccionar sus características más significativas en base a una serie de condiciones establecidas.

DesignBuilder es una potente herramienta de diseño que integra diferentes modos de simulación, entre los que se encuentra el motor de EnergyPlus. Gracias a el programa, podemos tomar en cuenta parámetros como la eficiencia térmica, el uso de energía, cerramientos del edificio, actividad laboral, etc… todo ello de la mano de una sencilla interfaz que nos permitirá seleccionar diferentes opciones.

Una vez configurado el modelo, DesignBuilder nos da la facilidad de simularlo y realizar un control clásico sobre él. Descarga de una base de datos los ficheros de clima y exporta los resultados a unos gráficos representativos, donde se pueden ver las diferentes variables de temperatura, consumos, rendimientos, etc…

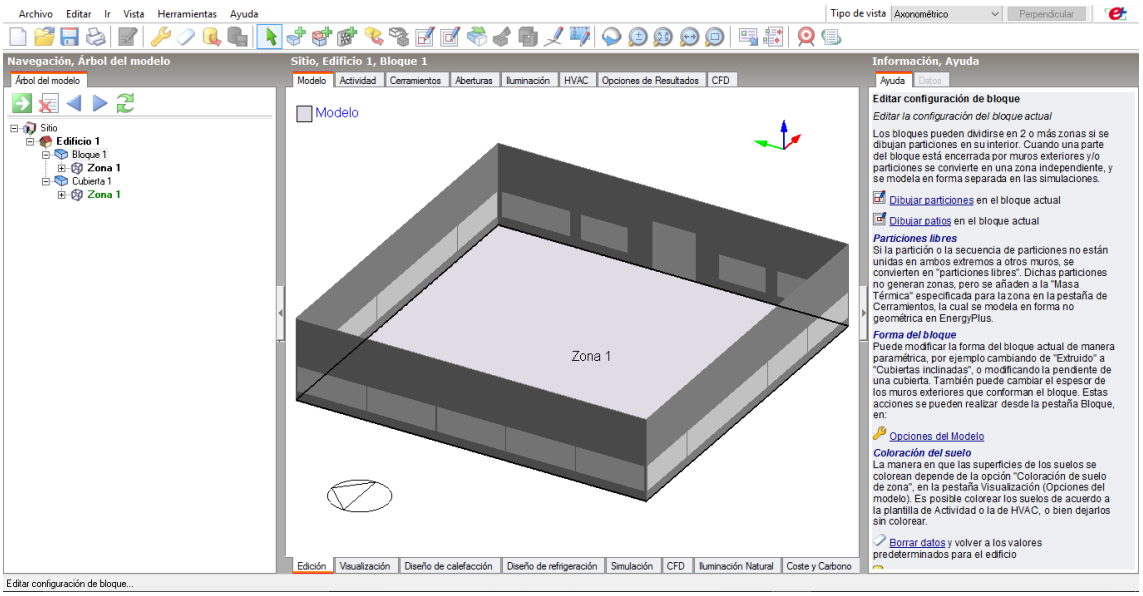

Figura 10. Interfaz de DesignBuilder

DesignBuilder además nos permite exportar los resultados del modelo creado a un archivo IDF, lo que resulta de gran utilidad ya que de esta manera podemos crear un modelo en EnergyPlus partiendo desde 0 y con todas las configuraciones necesarias.

4. MODELADO DEL SISTEMA

4.1. Creación del Modelo

Como punto de partida, es necesario crear un modelo inicial que servirá como modelo real a controlar. Para tal fin, se ha empleado la herramienta EnergyPlus, un potente software de simulación energético.

EnergyPlus utiliza ficheros IDF, que no son más que archivos de texto con datos del modelo y del sistema energético. Ya que el programa no cuenta con su propia interfaz gráfica o de diseño, se empleará un software externo para crear el modelo, diseñar la máquina de climatización, y exportar estos datos para trabajar más adelante con ellos. Para tal fin se emplea el software DesignBuilder, en el cual viene incorporado EnergyPlus en sus diferentes versiones.

4.1.1. Diseño

Una vez instalado y abierto el DesignBuilder, creamos un nuevo proyecto, donde es necesario introducir los datos de la zona climática a simular (aunque luego se puedan modificar por el usuario) así como otros datos correspondientes a las normativas de según qué país nos encontremos. Para dotar de mayor realidad al modelo seguiremos la norma ISO establecida en España para la construcción de oficinas.

Se creará un modelo simple que consiste en una oficina de planta baja con tejado, ventanas y una única puerta. Empleando la interfaz gráfica de desarrollo se selecciona un modelo base a seguir, se crea el contorno de la estructura y se incorpora a él las ventanas y la puerta, de tal manera que queda como lo representado en la figura 11.

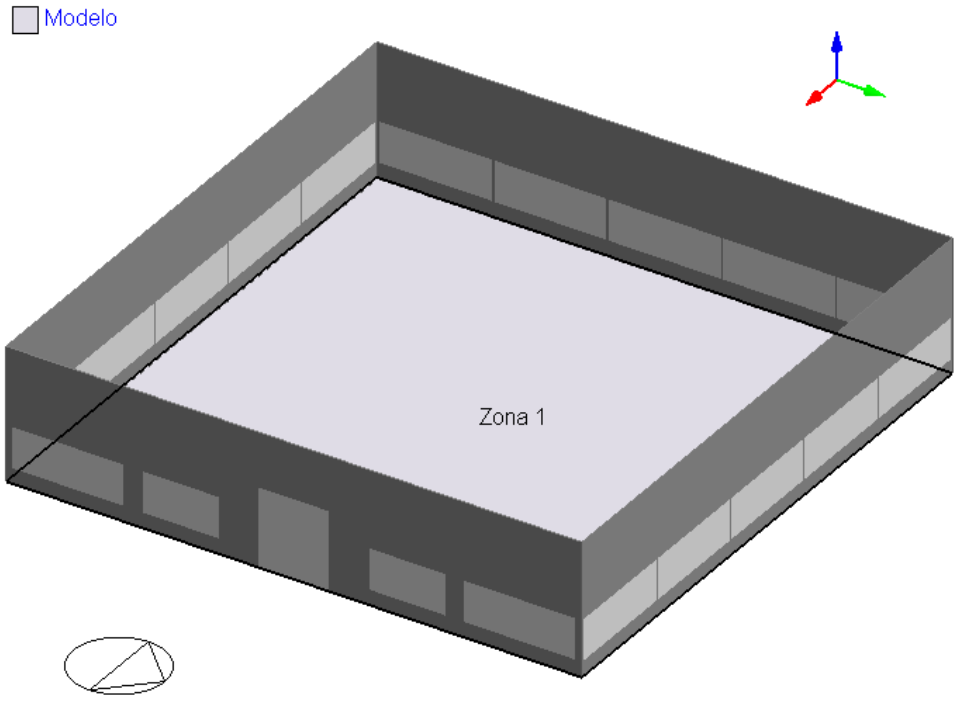

Figura 11. Modelo real a controlar

Para evitar que la zona se caliente en exceso sobre todo durante el verano, añadimos una cubierta de manera que el modelo quede dividido en dos zonas: planta baja y tejado. Únicamente se ocupará la primera de las zonas ya que es donde interesa aplicar el control, dejando el tejado como aislante término artificial. Además, en este último las temperaturas pueden alcanzar valores entre 40-50 º C en los meses más cálidos, por lo que el consumo de la máquina de clima se dispararía e imposibilitaría el control del mismo debido a su complejidad.

Para seleccionar los cerramientos y materiales de la zona, se emplean las plantillas incorporadas en el programa. Se ha seleccionado una plantilla estándar que recoge la normativa española para el diseño y construcción de oficinas. En la pestaña de actividad, con el fin de simular e identificar el modelo de manera correcta, se ha designado los parámetros para que no afecten directamente a la temperatura operativa de la zona (variable de salida de nuestro sistema) tal y como se muestra en la figura 12. Se establecen unas consignas de referencia para simular y comprobar que todo funcione correctamente.

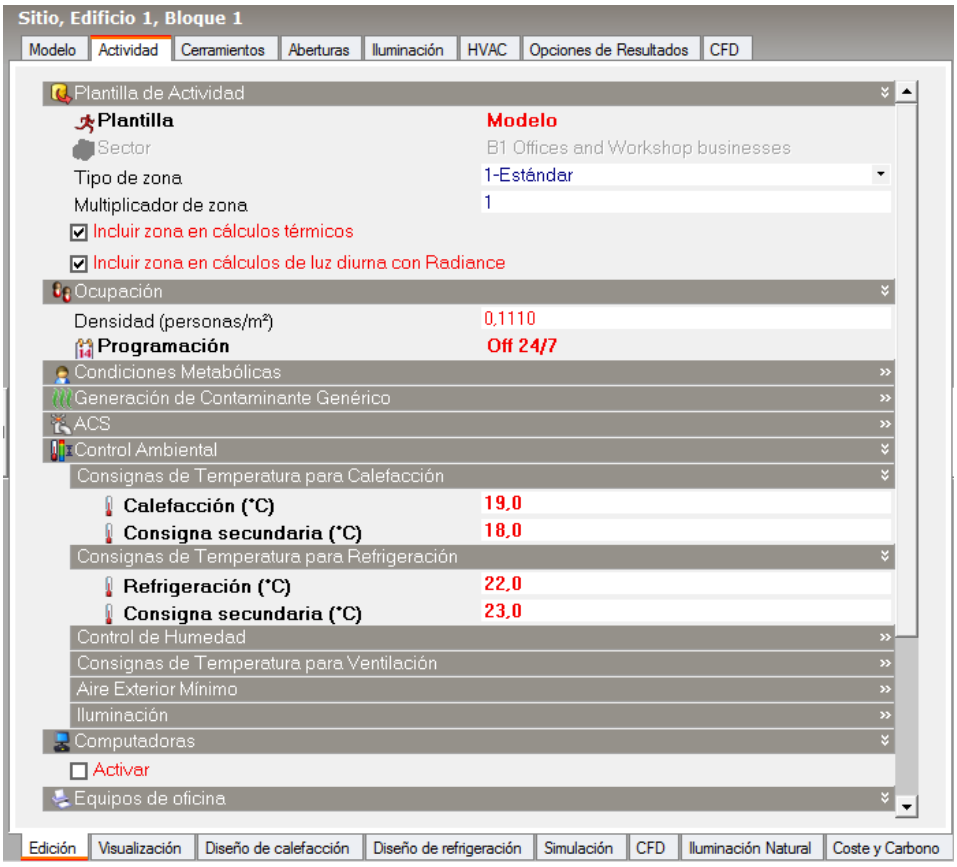

Figura 12. Pestaña de Actividad

En la pestaña de HVAC, seleccionamos la plantilla de un modelo que emplee consumo eléctrico tanto para calentar como para enfriar. El modelo de HVAC seleccionado es un PTAC Electric Heating, un aparato pequeño que se utiliza comúnmente en hospitales, oficinas, residencias, etc.… Es importante en el diseño de actividad de cada una de las funciones de la máquina, seleccionar el parámetro **'Always ON'** con el fin de simular la máquina fuera de las horas normales de oficina (por defecto viene limitado su uso a un horario de 8:00 AM a 7:00 PM). Comprobamos que tanto para calentar como para enfriar este seleccionado como combustible **'Electricidad'.**

Dentro de Editar>Propiedades del modelo, se debe simular el HVAC como **'compacto'** en vez de como **'simple'**. El modo simple no nos permite modificar los parámetros de la maquina HVAC con el EnergyPlus, mientras que en el modo compacto si (de esta manera podemos sacar los reportes de consumo y modificar los termostatos desde el archivo IDF correspondiente).

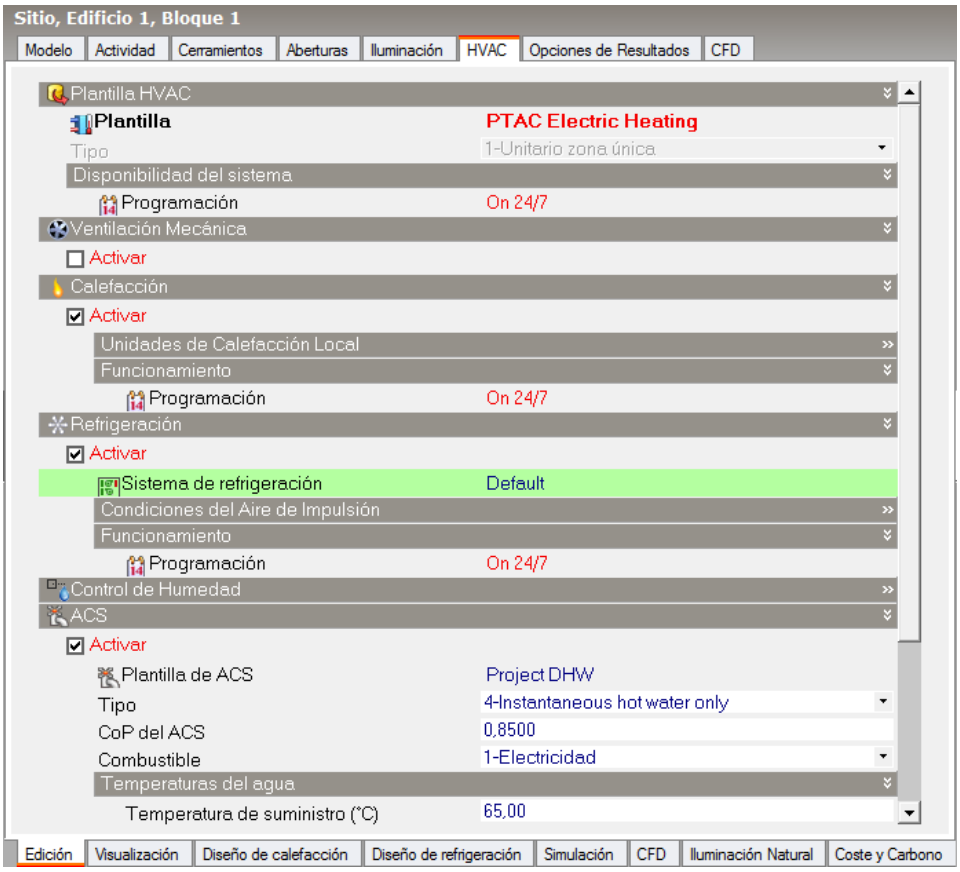

Figura 13. Pestaña HVAC

Realizamos una simulación previa empleando el motor de EnergyPlus dentro de DesignBuilder, obteniendo los resultados según los setpoints fijados en el programa. De las diferentes versiones que existen emplearemos la 7.2.0 debido a problemas de compatibilidad con el toolbox de Matlab en versiones más recientes. El programa nos permite descargar directamente el ejecutable desde sus servidores para llevar a cabo la simulación.

En los parámetros a configurar, programamos el día de inicio y de fin y activamos todas las casillas de reporte. En el tipo de control seleccionamos **'Temperatura Operativa'.** Los resultados finales se muestran en la figura 14.

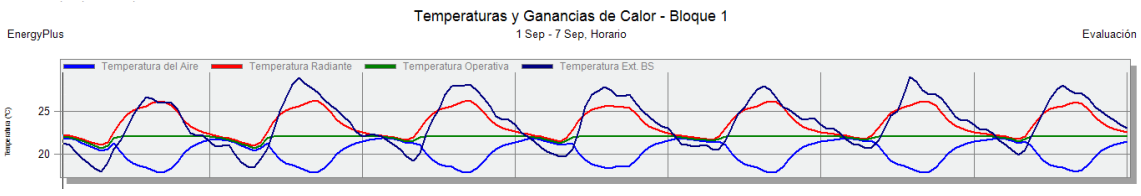

Figura 14. Temperatura simulada para una semana.

Solo nos queda exportar nuestro proyecto a un archivo IDF, con el cual trabajaremos a partir de ahora. El modo de exportación debe ser para diseño de simulación (no confundir con calefacción o refrigeración).

4.1.2. EnergyPlus-ExternalInterface

Para poder seleccionar los valores de setpoints de calor y frio hace falta crear un entorno de Co-simulación entre EnergyPlus y un programa externo.

Con el objeto de lograr ese fin, es necesario activar la ExternalInterface dentro del archivo IDF. Para ello emplearemos el programa IDFEditor, que no es más que una herramienta gráfica para hacer modificaciones sobre el archivo IDF de manera más cómoda y sencilla que un editor de texto estándar (Notepad++).

La función de ExternalInterface fue creada en principio para utilizarla junto al programa BCVTB (Building Controls Virtual Test Bed) y dentro de la misma con Ptolemy. No obstante, se puede emplear otro programa que realice la función de simulación.

El archivo generado por DesignBuilder es abierto con el IDFEditor (por defecto con EnergyPlus) mostrando una primera imagen como se refleja en la Figura 15. En primer lugar hace falta establecer unos parámetros iniciales de simulación que se irán modificando a lo largo del desarrollo del proyecto. Se debe seleccionar en la pestaña Timestep el paso de simulación de una hora. De momento lo dejaremos en un valor de 3 (cada paso de simulación se establece en 20 minutos).

A continuación programamos un día cualquiera en la pestaña RunPeriod. Esta pestaña deberá ser modificada de manera manual para simular distintos intervalos de tiempo, teniendo en cuenta que lo mínimo a simular es un día, y que las simulaciones solo se pueden realizar de días completos (no se puede cortar a una hora en concreto). Dejamos el resto de opciones por defecto.

Las pestañas de ‘Schedule’ corresponden a los diferentes comportamientos de los elementos que intervienen en el modelo. No se debe de modificar si se ha establecido la función **‘Always ON’** en DesignBuilder. Todos los parámetros de simulación energética del programa están controlados por una variable ‘Schedule’, pudiendo fijar límites según la hora o comportamientos específicos para días festivos o de fin de semana.

Las siguientes pestañas hacen referencia a la parte de diseño de la estructura, por lo que no se debe modificar si se ha realizado correctamente el modelo en DesignBuilder. Comprobamos que la actividad humana se encuentra desactivada (no hay gente trabajando en las simulaciones) en el apartado Zone:People y más concreto en su valor límite de Schedule.

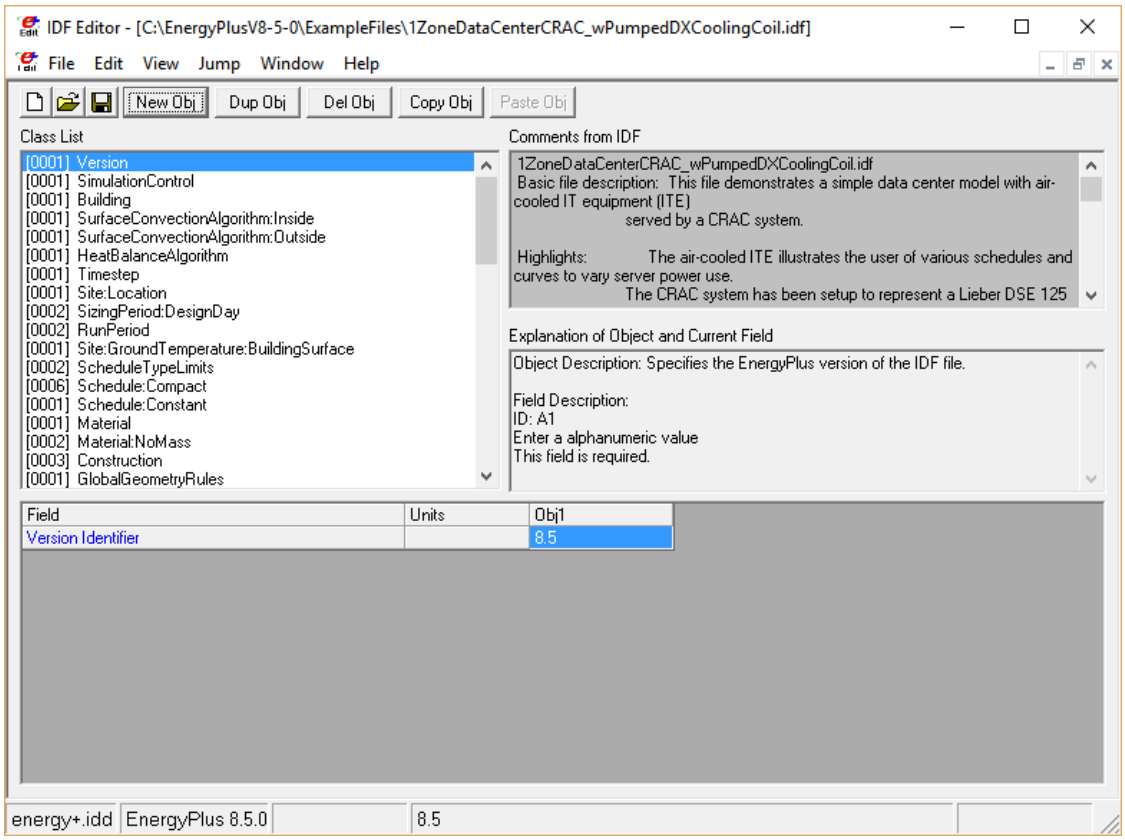

Figura 15. Interfaz Gráfica IDF Editor

Dentro del apartado ExternalInterface, crearemos un objeto para activar dicha función, estableciendo el valor en PtolemyServer. Cabe señalar que se puede importar un FMU para realizar la Co-simulación, aunque se ha optado por emplear las funcionalidades básicas para realizar el control predictivo. De esta manera, nos encontramos con 3 posibles candidatos:

- **ExternalInterface (Schedule):** Esta será la función a emplear para controlar el termostato de la máquina de climatización, debido a que solo contiene un único valor que se usa en cada paso de tiempo, seleccionado por el usuario. Se puede emplear para seleccionar consignas o variar la disponibilidad de la máquina.
- **ExternalInterface (Variable):** La función variable puede ser modificada durante la simulación a lo largo de cada paso. No obstante, no puede modificar los setpoint establecidos, por lo que se descarta como funcionalidad a emplear.
- **ExternalInterface (Actuator):** Esta función sirve para utilizar actuadores. No se emplea para el control.

El control de la máquina HVAC puede establecerse con dos termostatos para frio y calor respectivamente, o un único termostato con función dual. Independientemente

de cual se elija, se necesitarán dos valores de ExternalInterface:Schedule para modificar tanto la consigna de frio como la de calor. Por lo tanto se crean dos objetos, y se establece el límite de los valores que se pueden elegir (por defecto cualquier valor).

Field	Units	Obj1	Obj2
Name		HeatingSP	CoolingSP
Schedule Type Limits Name		Any Number	Any Number
Initial Value		20	20

Figura 16. Variables a controlar en la ExternalInterface

El valor inicial es importante establecerlo en un valor por defecto ya que EnergyPlus realizar una presimulación de 15 días para ver el punto de funcionamiento en la hora en la que se inicia nuestra simulación. Si se establece un valor de apagado (setpoint de calor y frio en 0 y 40 grados respectivamente) el comportamiento de la máquina puede originar problemas. Lo normal es modificar el valor según se realicen pruebas y ya en el control inicializar el valor de consigna en el mismo valor seleccionado en este punto.

Por defecto, el IDF creado a partir de DesignBuilder viene establecido con un termostato dual (se puede configurar en DesignBuilder), por lo que si nos vamos a dicha pestaña, se observará que el objeto ya está creado. En caso de que no lo estuviera, se necesitaría crear uno a partir de cero. Es importante realizar el control sobre la temperatura operativa de la zona (ZoneControl:Thermostat:OperativeTemperature), para lo que se tenía que haber seleccionado previamente durante la exportación del IDF (*Véase 4.1.1.Diseño)*.

En la pestaña de ThermostatSetpoint:DualSetpoint hay un objeto por el cual se puede seleccionar el valor Schedule tanto para frio como para calor. Se escribe de manera manual los valores creados anteriormente en la ExternalInterface para que a la hora de modificar dichos parámetros, éstos afecten directamente al termostato y seleccionen la consigna designada. En cada paso de simulación, EnergyPlus consultará dichos valores, pudiendo modificarse según qué condiciones.

Field	Units	Obj1
Name		319 Thermostat Dual SP Control
Heating Setpoint Temperature Schedule Name		HeatingSP
Cooling Setpoint Temperature Schedule Name		CoolingSP

Figura 17. Termostato a controlar

Por último, necesitamos designar los valores reportados por EnergyPlus. El sistema cuenta con 3 entradas (Temperatura Ambiente, Radiación Solar y Setpoint) y 2 salidas (Temperatura Operativa y Consumo). Por defecto, los objetos creados en

ExternalInterface son reportados automáticamente. Para las otras variables, hace falta crear un objeto específico para cada una de ellas en la pestaña Output:Variable y para el consumo en la pestaña Output:Meter.

REPORTE	Output:Variable			Output:Meter
Key Value	319	*	*	----------
Variable Name	Zone Operative Temperature	Outdoor Dry Bulb	Direct Solar	Electricty:HVAC
Reporting Frequency	Timestep	Timestep	Timestep	Timestep

Tabla 1. Variables a reportar

- **Key Value:** Valor por el cual se establece la zona donde se quiere medir la variable. Por defecto, DesignBuilder nombra numéricamente las zonas. En este caso la '319' corresponde con el bloque principal mientras que la '355' corresponde con el tejado.
- **Variable Name:** En este apartado se selecciona el nombre de la variable a reportar. Dentro del manual de EnergyPlus viene un listado con los diferentes nombres para cada una de ellas. La temperatura del bulbo corresponde con la temperatura ambiente.
- **Reporting Frequency:** Aquí se establece la frecuencia de reporte, en la cual se designará que sea para cada paso de tiempo por si este es modificado por el usuario.

Después de realizar todas estas modificaciones, ya disponemos de un IDF preparado para ser manipulado por MLE+.

4.2. MLE+ (Matlab EnergyPlus)

Con el IDF listo para la simulación, procedemos a descargar el toolbox MLE+ de la página principal, y la instalamos sobre la versión de Matlab 2011b (por temas de compatibilidad). Se arranca la interfaz con el comando **'mlep'**.

En la pestaña principal se muestra un esquema estándar de lo que se puede realizar con esta herramienta. Para el proyecto no se emplearán utilidades como la

identificación del sistema o la implementación de reguladores. Para comenzar, seleccionaremos el IDF en la ventana desplegable correspondiente junto con el archivo de clima a emplear en la simulación.

A continuación nos dirigimos a la pestaña de Control y hacemos clic en el botón '**Variables**'. Se nos abrirá una nueva pestaña donde designaremos tanto las variables de entrada al EnergyPlus, como las variables de salida extraídas del mismo. Para ello pulsamos en **'Load IDF'** y nos aparecerá un listado de las entradas y salidas. Estas variables serán empleadas en el archivo de control de Matlab. Como solo vamos a modificar los Setpoints de frio y de calor, los introducimos al sistema y les designamos un Alias que se empleará más adelante. Finalmente pulsamos sobre el botón '**write variables.cfg'** el cual nos creará un archivo de texto que habilitará las variables para ser utilizadas por la ExternalInterface.

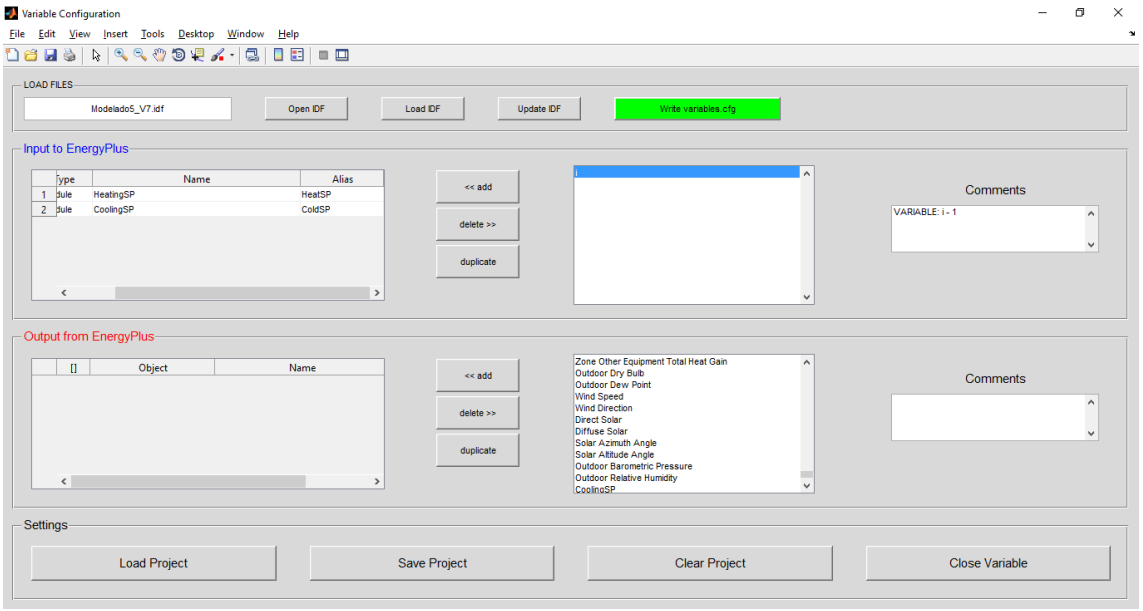

Figura 18. Pestaña de configuración de E/S

Una vez designadas las entradas, cerramos la pestaña y pulsamos sobre **'Create Control File'**, el cual nos generará un archivo .m que utilizará la toolbox para el control del sistema. En el botón **'Edit Control File'** se nos abrirá el editor de Matlab. En el archivo viene creado por defecto la función que se utilizará en la simulación.

El archivo de control viene definido de tal forma que según el valor de cmd ejecutaremos una parte u otra.

- Valores iniciales ('init'): En esta parte del código se inicializan las variables a manipular con un valor. Solo se ejecuta una vez y al principio de la simulación.

- Cuerpo del código ('normal'): En la segunda parte se escribe el código principal del programa. Se ejecuta de manera cíclica tantos pasos de simulación tengamos.

Para extraer y modificar las variables que introduciremos en EnergyPlus emplearemos la función **‘eplus_in_curr.(el alias de la variable)’**, la cual igualaremos al valor que nosotros deseemos establecer (en nuestro caso el valor de consigna. Para realizar una simulación con la máquina HVAC desactivada, escribiremos las siguientes líneas en la inicialización de las variables:

```
eplus_in_curr.HeatSP = 0;
eplus_in_curr.ColdSP = 40;
```

Al establecer unos valores de consigna lejos de los límites de temperatura de la zona (de manera aproximada), la máquina nunca será activada (en el caso anterior la temperatura tendría que caer de 0 grados o superar los 40 grados para que funcionase).

Guardamos la función y cerramos el editor. Nos vamos a la pestaña de Simulación y pulsamos en **‘Run Simulation’.** Se nos abrirá una consola donde se ve reflejado que hace en cada momento el motor de simulación. En el caso de que haya algún error a la hora de configurar, la consola nos indicará que la simulación ha fallado. En este caso se debe revisar tanto la ventana de comandos de Matlab como un archivo .err donde se recogen los errores. Dicho archivo se crea por defecto en una carpeta Output en el mismo directorio donde se encuentra el IDF.

Realizando correctamente la configuración, los resultados deberían ser como se muestran en la figura 19 (dependiendo de la fecha a simular)

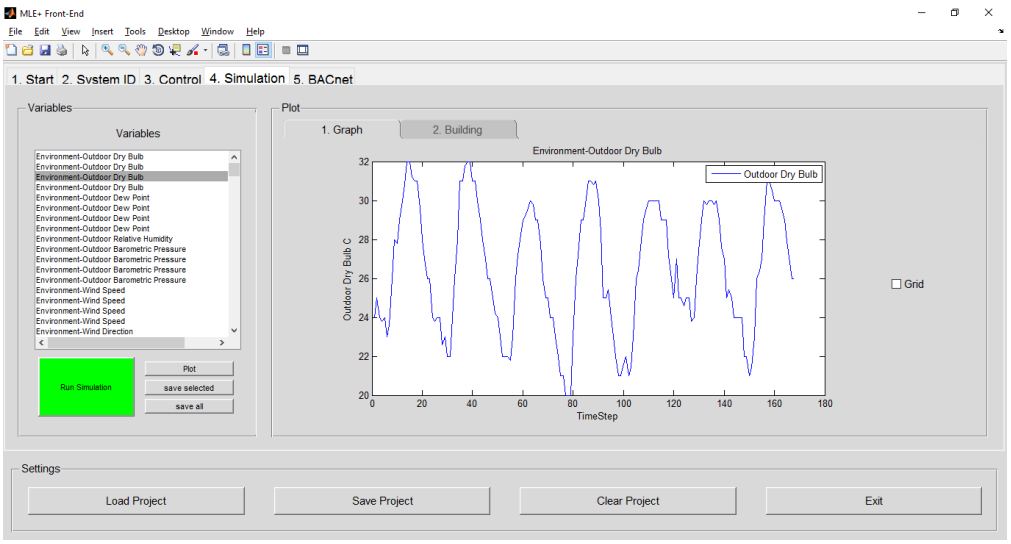

Figura 19. Ejemplo de simulación de una semana de Julio.

En la ventana de la izquierda, se muestran todas las variables configuradas en el apartado Output:Variable y Output:Meter del IDF simulado, mientras que en la ventana derecha se muestra la gráfica de la variable seleccionada una vez pulsado el botón **'plot'.**

Observamos que la temperatura oscila correctamente, siendo los picos más altos los correspondientes a las horas de mayor calor. Según el valor del Timestep designado, sabremos el valor de la variable para cada paso de tiempo (si seleccionamos un valor de '1' cada Timestep se corresponde a 1 hora).

Ahora realizaremos la misma prueba pero activando la climatización de la zona. Para ello hace falta modificar el archivo de control para que tome los valores que nosotros le escribamos en el programa. Creamos dos variables denominadas **'HeatSP'** y **'ColdSP'** y les fijamos un valor numérico de 23 en ambas. Esto quiere decir que la máquina forzará la temperatura operativa de la zona a que sea constante en 23 grados. Es necesario actualizar la variable en cada paso de tiempo añadiendo al final las dos líneas siguientes:

```
eplus_in_curr.HeatSP = HeatSP;
eplus_in_curr.ColdSP = ColdSP;
```

También se puede establecer un control de banda, por ejemplo poniendo los valores de los setpoints de calor y frio en 19 y 23 respectivamente. De esta manera la temperatura oscilará entre esas dos temperaturas y solo se activará la máquina cuando sobrepase dichos límites. Los resultados con la máquina activada se muestran en la figura 20.

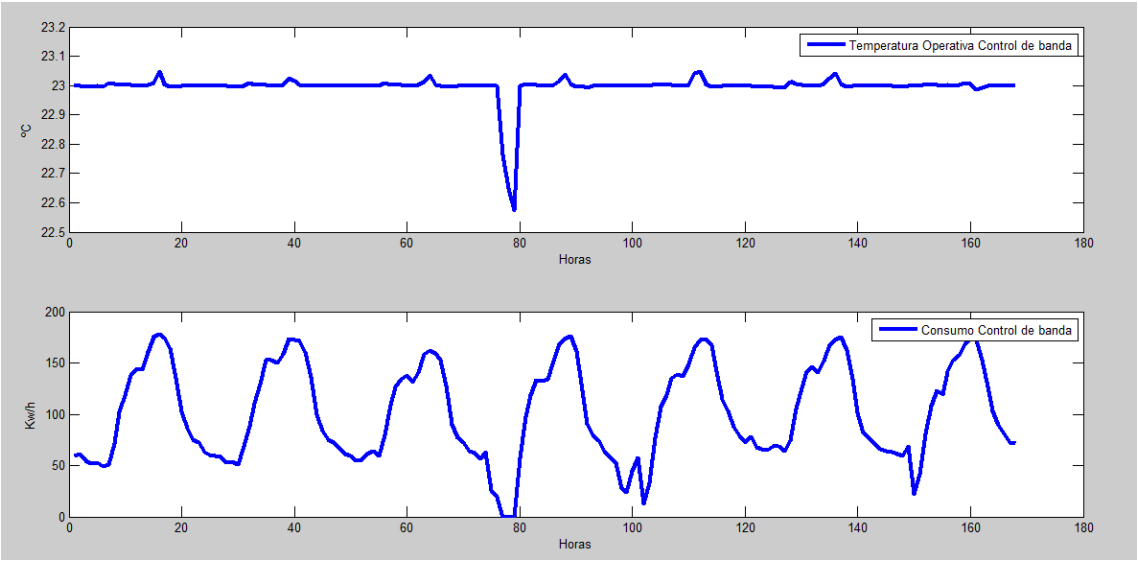

Figura 20. Temperatura y Consumo de salida

4.3. Identificación del Sistema

Una vez creado el entorno de trabajo, procedemos a la identificación del sistema para usar el modelo real en el control predictivo. Para realizar la identificación, necesitamos en primer lugar definir las entradas y salidas del sistema:

- 3 Entradas: Temperatura Ambiente, Radiación Solar y Setpoint.
- 2 Salidas: Temperatura Operativa y Consumo.

Se ha seleccionado únicamente las entradas con mayor peso e influencia sobre las dos salidas, ya que otras variables como la humedad, radiación difusa, ángulo Azimuth (el ángulo que forma el son con la tierra) también son tenidas en cuenta por EnergyPlus, aunque su influencia no es tan significativa como las anteriormente nombradas.

Como previamente se creó el fichero de EnergyPlus sin actividad ninguna, no hay que tener en consideración variables como la cantidad de personas trabajando o los equipos electrónicos conectados y que desprenden calor.

Antes de comenzar hay que tener en cuenta una limitación del motor de simulación en cuanto a la variable de radiación solar. Para calcular su valor final, se emplea la siguiente fórmula:

$$Radiación = Rs * cos\theta + Rd$$

donde Rs representa la irradiación directa proveniente de los rayos del anillo solar, Rd la radiación difusa en una superficie horizontal y θ el ángulo que forma el sol con la superficie de la tierra.

La radiación solar (o radiación normal) es la cantidad de radiación en unidades de Wh/m^2 directa proveniente del sol sobre una superficie perpendicular a él durante el número de minutos indicado. El problema reside en el ángulo solar y la radiación difusa, ya que toman un valor de cero durante las horas de noche, tal y como se observa en la Figura 21.

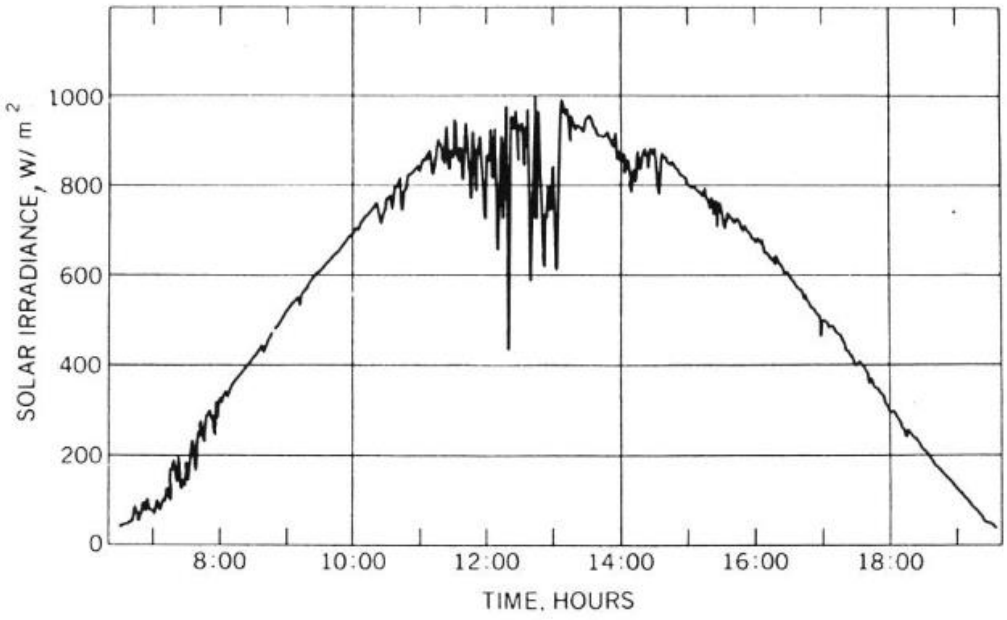

Figura 21. Radiación Solar directa durante el día.

La solución aparente al problema es someter al sistema a escalones únicamente durante el día, de manera que durante la noche su valor sea cero. Para que la variable no afecte al resto de identificaciones (ya que aunque se fije en un valor constante éste dependerá de la hora de simulación) se debe poner un valor bajo como por ejemplo 1 Wh/m^2.

4.3.1. Modificación archivos climáticos

De manera práctica, únicamente se puede variar el setpoint al que se somete el sistema, siendo los valores de temperatura y radiación proporcionados por el archivo de clima. No obstante, para realizar la identificación del modelo es necesario modificar las variables de entrada introduciendo escalones en cada una de ellas y dejando el resto constante para ver su influencia en el sistema (se espera un comportamiento lineal de primer o segundo orden).

Como ya se había descrito anteriormente, los archivos climáticos en realidad son archivos de extensión .csv, por lo que se pueden modificar con el uso de un programa externo. El elegido para tal labor es el Ron's Editor, que con una interfaz muy similar a la de Excel nos permite modificar los valores de las celdas según a las columnas que pertenezcan. Previamente, seleccionaremos una semana que nos servirá como semana

de simulación (la elegida ha sido del 1 de Septiembre al 7 de Septiembre) la cual debe configurarse dentro del archivo IDF para que se simule.

A continuación seguiremos una serie de pasos para identificar cada una de las FDT que afectan al modelo:

- Creación del fichero de clima para cada variable, sometiendo cada una de ellas a diferentes valores de escalones y dejando el resto constante.
- Simulación del modelo con los ficheros creados para cada una de las salidas, teniendo en cuenta si la máquina debe estar encendida o apagada.
- Exportar los datos al Matlab, graficar los valores para ver que son correctos y guardarlos en matrices de columnas para usarlos con el toolbox.
- Abrir el Identification Toolbox e introducir las variables de entrada y salida, estimar una FDT y validar los datos.
- Si el resultado ha sido favorable, exportar los datos para utilizarlos en el modelo de Simulink.

En total realizaremos 8 identificaciones distintas, que se reparten de la siguiente forma:

- Temperatura Ambiente → Temperatura Operativa
- Radiación Solar → Temperatura Operativa
- Setpoint de Frio → Temperatura Operativa (Opcional)
- Temperatura Ambiente → Consumo
- Radiación Solar → Consumo
- Setpoint de Frio → Consumo
- Arranque HVAC → Temperatura Operativa*
- Arranque HVAC → Consumo*

*La identificación del arranque se realiza conmutando el Setpoint entre Encendido/Apagado.

4.3.2. Identificación Temperatura Ambiente sobre la Temperatura Operativa

Una vez convertido el fichero de clima en formato CSV, lo abrimos con el Ron's editor y modificamos las columnas desde el día 1 de Septiembre hasta el 7 de

Septiembre de tal manera que los valores de las constantes sean 0 (o 1 en el caso de radiaciones). A continuación, crearemos una secuencia de 4 escalones de temperatura tal y como se muestra en la figura 22.

A continuación, configuramos el RunPeriod para la semana modificada y seleccionamos un Timestep de 1 (pasos de 1 hora) dentro del archivo IDF. Ya dentro de MLE+ y con la configuración ya establecida, dejamos el archivo de control de tal manera que la máquina de clima este apagada (*Véase 4.2.MLE+)* y pulsamos sobre el botón simular. Seleccionamos ahora las siguientes variables y las guardaremos con **'save selected plot'** en una carpeta: Outdoot Dry Bulb, Direct Solar, 319 Zone Mean Operative Temperature, Electricity:HVAC y ColdSP.

Con el script adjunto en el Anexo I del apartado Anexos, extraemos los datos de EnergyPlus y graficaremos los mismos para ver que se han exportado de manera correcta (*Véase Figura 22)*.

Con la información guardada en una carpeta con formato .mat, ejecutamos un segundo script para extraer las variables por separado, denominando a las entradas con nomenclatura u* y a las salidas con nomenclatura y*, donde el asterisco corresponde al número de variable que hayamos designado. Finalmente, el script ejecutará el toolbox de identificación.

Una vez abierta la interfaz, en primer lugar cargaremos los datos de entrada de temperatura ambiente y salida de temperatura operativa seleccionando para ello el botón *import data>time data*. En la parte de entrada escribimos **'u2'** y en la parte de salida escribimos **'y1'**. Dejamos por defecto el tiempo de muestreo y el inicio en 1. Una vez cargada la gráfica comprobamos que es la correcta marcando la casilla **'time plot'**.

Si estimamos un modelo en el cual los valores de escalón son de alto valor, podemos tener complicaciones a la hora de obtener sus parámetros y validarlo, por lo que es conveniente realizar un preproceso a la señal en la cual quitaremos la media de las señales de entrada y salida. Esto quiere decir que el valor medio de la señal se ajustará de tal manera que su resultado sea 0. Una vez obtenida la señal procesada, la dibujamos para verificarla.

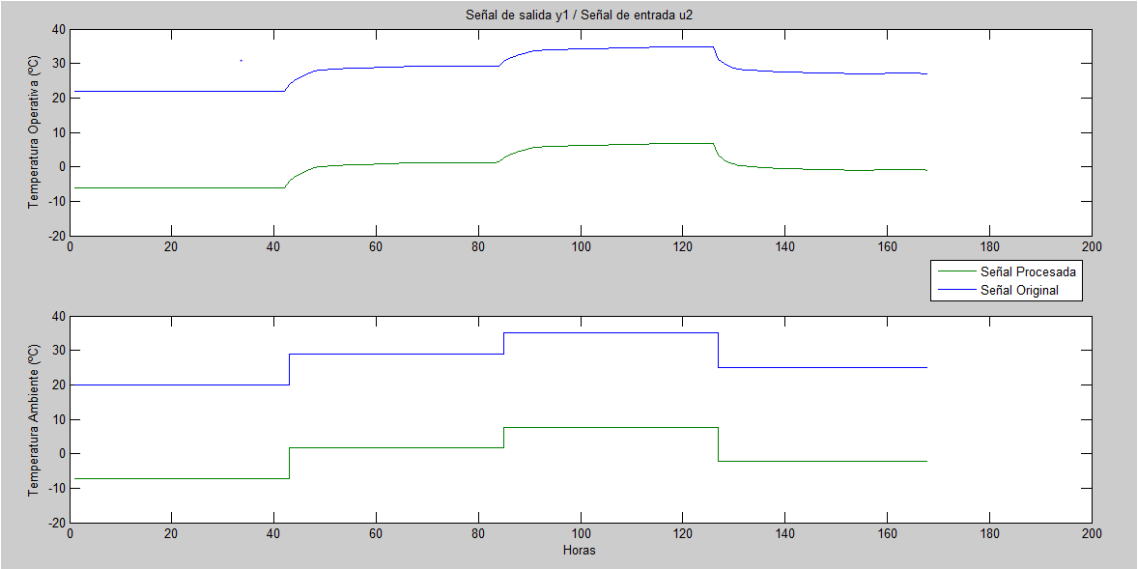

Figura 22. Señal original y procesada con media en 0

Ahora únicamente nos queda la estimación del proceso, así que para ello arrastramos la nueva señal a la casilla de '**working data'** y a la casilla de **'validation data'**. A continuación seleccionamos en *estimate>linear model* y se nos abrirá una nueva ventana donde podremos añadir polos y ceros para la estimación del modelo. Como se ha podido deducir ya por las gráficas, en nuestro caso observamos que el comportamiento del sistema corresponde a un primer orden con retardo, así que marcamos la casilla de '**delay'** y dejamos por defecto en automática la búsqueda de los parámetros. Pulsamos en **'estimate'** y el algoritmo se pondrá en marcha, realizando un número de iteraciones hasta obtener el resultado deseado o llegar a su límite (fijado en 20).

Finalmente y con el modelo estimado seleccionado, marcamos la casilla **'model output'** para comprobar el % de aproximación del modelo con el real. Como se observa en la figura 23, la coincidencia entre el modelo estimado y el real es de 85.62. Es un buen resultado, por lo que lo daremos por válido. Observamos principalmente que sigue las tendencias de temperatura y que no se aleja en exceso del valor de salida (error tolerable al ser temperatura).

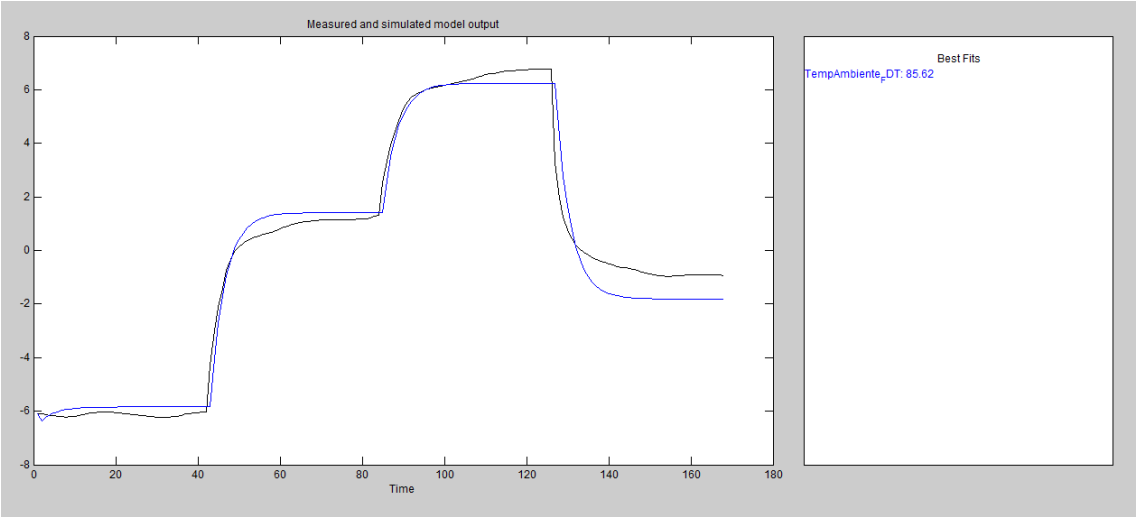

Figura 23. Modelo estimado para u2>y1

Se puede emplear otro conjunto de datos para verificar que la solución es correcta. Para ello basta con repetir los pasos anteriores y arrastrar la nueva gráfica a la casilla de verificación, donde se nos mostrará la salida de dicha entrada con respecto a nuestro modelo y su comparación con la salida real.

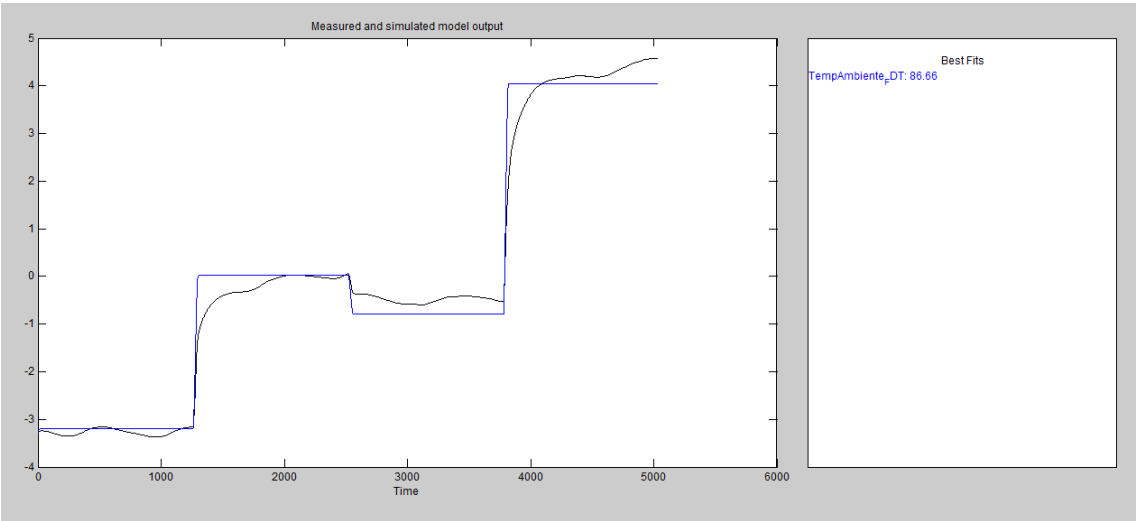

Figura 24. Validación del modelo resultante

Ya para terminar, exportamos los datos del modelo arrastrándolo a la casilla de **'to workspace'**, donde se nos creará una variable de tipo **<iddproc>** que más tarde emplearemos en la construcción del modelo de simulink.

4.3.3. Identificación Radiación Solar sobre la Temperatura Operativa

Una vez convertido el fichero de clima en formato CSV, lo abrimos con el Ron's editor y modificamos las columnas desde el día 1 de Septiembre hasta el 7 de Septiembre de tal manera que los valores de las constantes sean 0 (o 1 en el caso de radiaciones). A continuación, crearemos una secuencia de 7 escalones de radiación tal y como se muestra en la figura 25.

Hay que recordar que el cálculo de la radiación depende de la hora de simulación, por lo que aunque se coloque un valor en una hora de noche, este no se tendrá en cuenta en el cálculo de la radiación, por lo que emplearemos el día para introducir un escalón completo.

Ya dentro de MLE+ y con la configuración ya establecida anteriormente, dejamos el archivo de control de tal manera que la máquina de clima este apagada (*Véase 4.2 MLE+)* y pulsamos sobre el botón simular. Seleccionamos ahora las siguientes variables y las guardaremos con **'save selected plot'** en una carpeta: Outdoot Dry Bulb, Direct Solar, 319 Zone Mean Operative Temperature, Electricity:HVAC y ColdSP.

Con el script adjunto en el Anexo I del apartado Anexos, extraemos los datos de EnergyPlus y graficaremos los mismos para ver que se han exportado de manera correcta (*Véase Figura 25)*.

Con la información guardada en una carpeta con formato .mat, ejecutamos un segundo script para extraer las variables por separado. Finalmente, el script ejecutará el toolbox de identificación.

Una vez abierta la interfaz, en primer lugar cargaremos los datos de entrada de radiación solar y salida de temperatura operativa seleccionando para ello el botón *import data>time data*. En la parte de entrada escribimos **'u3'** y en la parte de salida escribimos **'y1'**. Dejamos por defecto el tiempo de muestreo y el inicio en 1. Una vez cargada la gráfica comprobamos que es la correcta marcando la casilla **'time plot'**.

Realizamos el preproceso de quitar la media a la señal importada, de tal manera que podamos estimar correctamente el modelo.

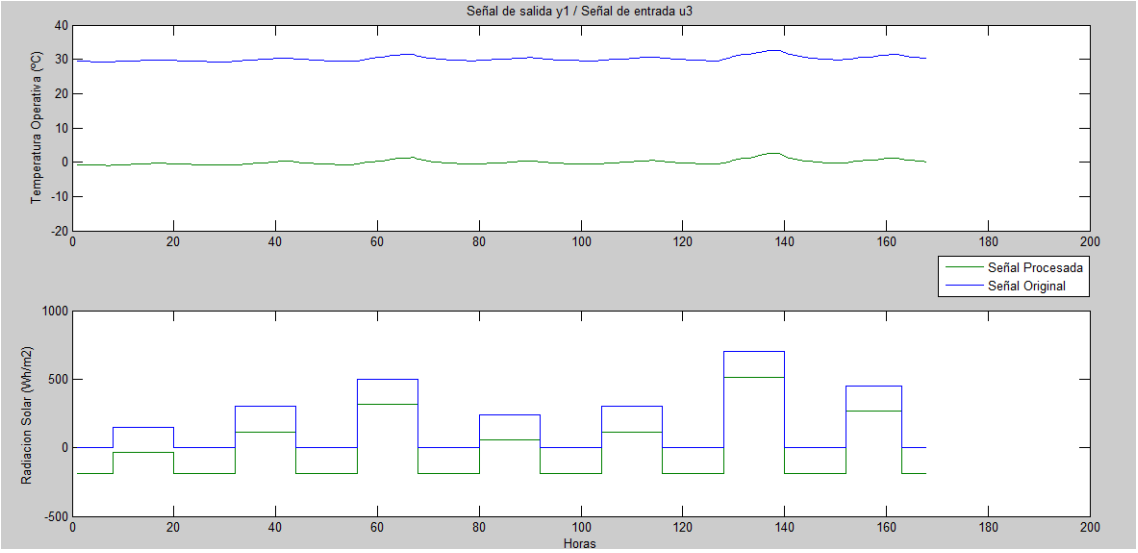

Figura 25. Señal original y procesada con media en 0

Ahora únicamente nos queda la estimación del proceso, así que para ello arrastramos la nueva señal a la casilla de '**working data'** y a la casilla de **'validation data'**. A continuación seleccionamos en *estimate>linear model* y se nos abrirá una nueva ventana donde podremos añadir polos y ceros para la estimación del modelo. En este caso nos encontramos con que la señal tiene un comportamiento diferente al de la temperatura, por lo que podemos estudiar diferentes casos de FDT. Para empezar, estimaremos un modelo de primer orden con retardo. A continuación, repetimos el proceso pero añadiendo un cero en vez de un retardo.

Finalmente y con el modelo estimado seleccionado, marcamos la casilla **'model output'** para comprobar el % de aproximación del modelo con el real. Como se observa en la figura 26, la coincidencia entre el modelo estimado y el real es de 75.73 para él caso del primer orden con retardo y de 82.93 para el caso del primer orden con un cero. Como podemos observar, el resultado es bastante similar en un caso que el otro, por lo que seleccionaremos el más sencillo a simular, es decir, el modelo del primer orden con retardo.

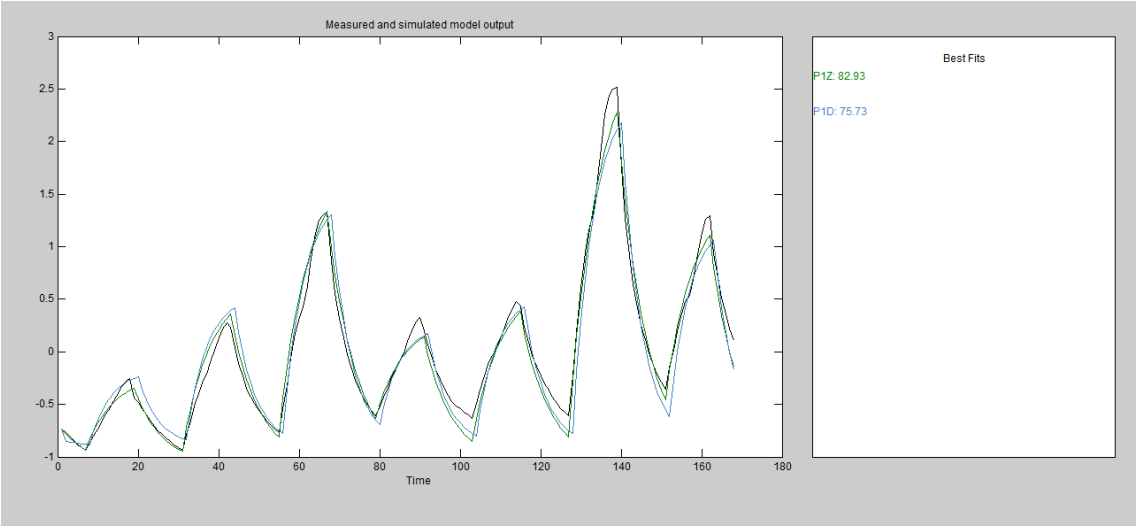

Figura 26. Comparación entre dos modelos estimados frente al real.

Validamos el modelo elegido con un conjunto de datos diferentes al de entrada, comprobando que los resultados sean satisfactorios.

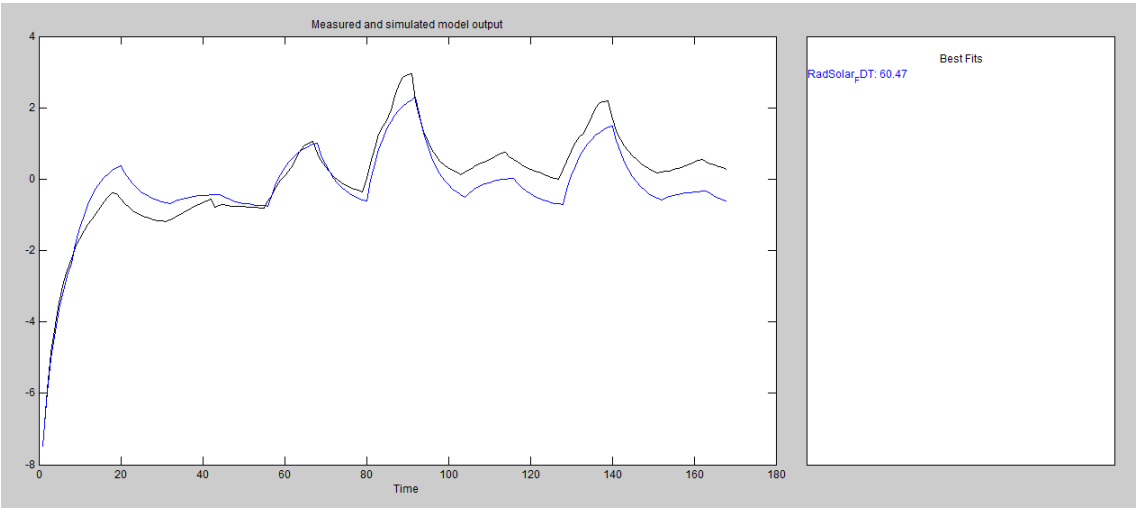

Figura 27. Validación de los datos con el modelo estimado.

En este caso vemos que los resultados no son tan buenos como en el caso de la temperatura. No obstante, esto se debe a que los datos de validación fueron introducidos con un estado inicial diferente al de los datos originales, dando lugar a un ligero error que se va arrastrando. Aun así la coincidencia no es del todo mala, por lo que lo damos por bueno.

Ya para terminar, exportamos los datos del modelo arrastrándolo a la casilla de **'to workspace'**, donde se nos creará una variable de tipo **<iddproc>** que más tarde emplearemos en la construcción del modelo de simulink.

4.3.4. Identificación Setpoint sobre la Temperatura Operativa (Opcional)

Una vez convertido el fichero de clima estándar en formato CSV, lo abrimos con el Ron's editor y modificamos las columnas desde el día 1 de Septiembre hasta el 7 de Septiembre de tal manera que los valores de las constantes sean 0 (o 1 en el caso de radiaciones). En este caso, los escalones de consigna se introducirán en el archivo de control del MLEP tal y como se muestran en la figura 28.

Ya dentro de MLEP y con la configuración ya establecida anteriormente, dejamos el archivo de control de tal manera que la máquina de clima varíe su consigna fijando los dos setpoints en un valor fijo, variándolo para que nos queden 4 escalones. Basta con crear un vector con longitud los 7 días x 24 horas y establecer las consignas para cada uno de ellos, quedando de la siguiente manera:

```
SP(:,1:42) = meter el valor que toque;
SP(:,43:84) = meter el valor que toque;
SP(:,85:126) = meter el valor que toque;
SP(:,127:169) = meter el valor que toque;

HeatSP=SP(steptime);
ColdSP=SP(steptime);
```

De esta manera para cada paso de tiempo se asignará el valor correspondiente a la posición del vector de setpoints creado (en este caso similar para calor y frío). Se le añade 1 parámetro de más debido a que tiene en consideración el valor inicial.

Pulsamos sobre el botón simular. Seleccionamos ahora las siguientes variables y las guardaremos con **'save selected plot'** en una carpeta: Outdoot Dry Bulb, Direct Solar, 319 Zone Mean Operative Temperature, Electricity:HVAC y ColdSP.

Con el script adjunto en el Anexo I del apartado Anexos, extraemos los datos de EnergyPlus y graficaremos los mismos para ver que se han exportado de manera correcta (*Véase Figura 28)*.

Con la información guardada en una carpeta con formato .mat, ejecutamos un segundo script para extraer las variables por separado. Finalmente, el script ejecutará el toolbox de identificación.

Una vez abierta la interfaz, en primer lugar cargaremos los datos de entrada de radiación solar y salida de temperatura operativa seleccionando para ello el botón *import data>time data*. En la parte de entrada escribimos **'u1'** y en la parte de salida escribimos **'y1'**. Dejamos por defecto el tiempo de muestreo y el inicio en 1. Una vez cargada la gráfica comprobamos que es la correcta marcando la casilla **'time plot'**.

Realizamos el preproceso de quitar la media a la señal importada, de tal manera que podamos estimar correctamente el modelo.

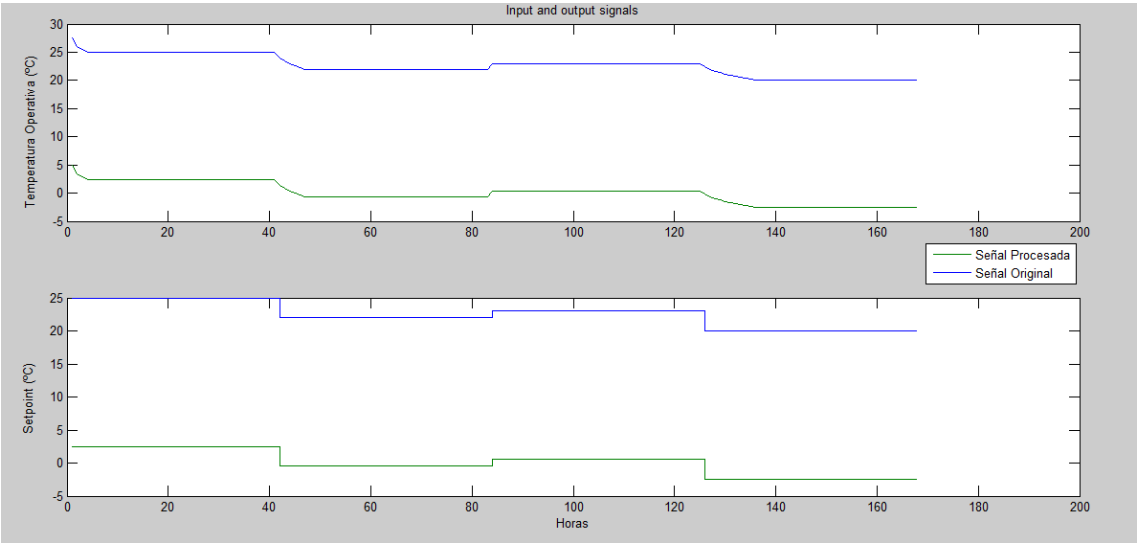

Figura 28. Señal original y procesada con media en 0.

Ahora únicamente nos queda la estimación del proceso, así que para ello arrastramos la nueva señal a la casilla de '**working data'** y a la casilla de **'validation data'**. A continuación seleccionamos en *estimate>linear model* y se nos abrirá una nueva ventana donde podremos añadir polos y ceros para la estimación del modelo. De nuevo observamos un comportamiento de primer orden con retardo, así que dejamos por defecto la casilla **'delay'** marcada. Observamos que el comportamiento es más rápido que en caso de la temperatura ambiente.

Finalmente y con el modelo estimado seleccionado, marcamos la casilla **'model output'** para comprobar el % de aproximación del modelo con el real. Como se observa en la figura 29, la coincidencia entre el modelo estimado y el real es de 87.87. Por lo tanto la coincidencia es bastante buena así que procedemos a validar los datos con otro conjunto de escalones de entradas.

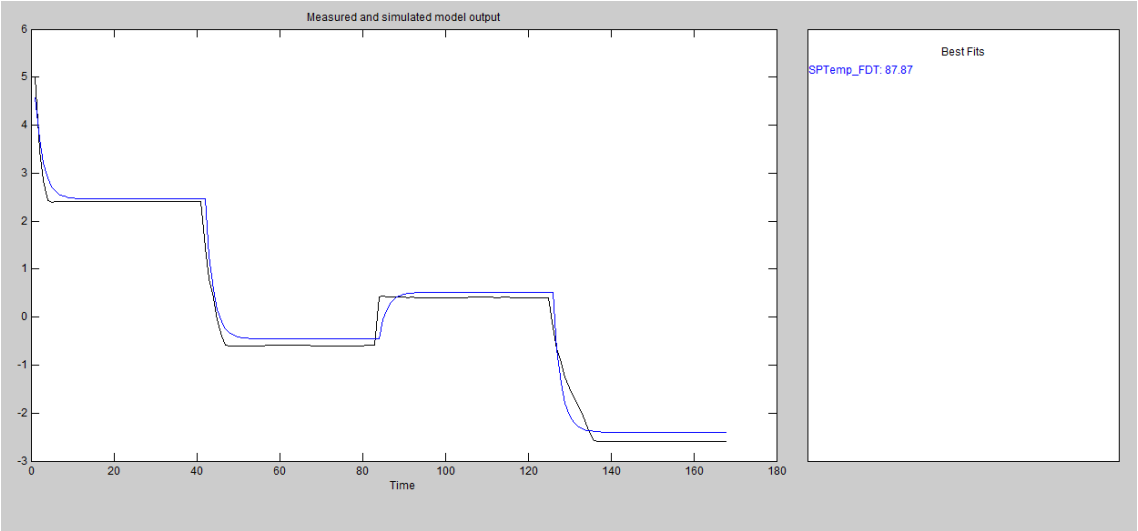

Figura 29. Modelo real frente al estimado

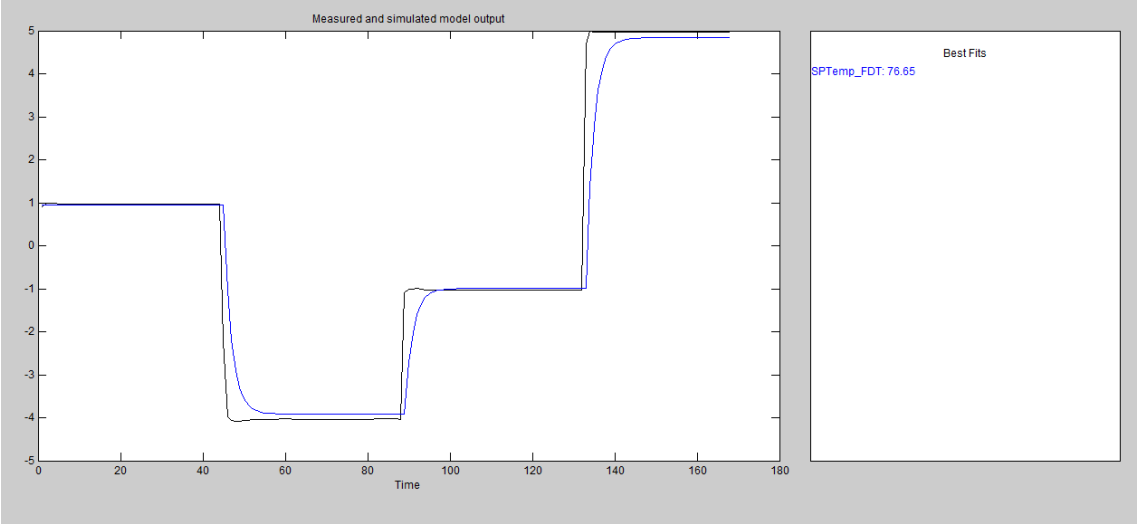

Figura 30. Validación del modelo.

Se ha comprobado que el sistema alcanza el valor muy lentamente cuando se simulan cambios entre horas. Se ha observado que entre el rango en el que nos movemos (de 19 a 23 grados) los cambios de temperatura se realizan antes de 1 hora, por lo que se puede seleccionar directamente el valor de consigna como el valor final de temperatura, la cual no tiene error alguno. Por lo tanto, no es necesario implementar esta parte. No obstante, se ha incluido de igual manera en el diagrama del modelo para usarse si se necesita en un futuro.

4.3.5. Identificación de Temperatura Ambiente sobre el consumo

Una vez convertido el fichero de clima estándar en formato CSV, lo abrimos con el Ron's editor y modificamos las columnas desde el día 1 de Septiembre hasta el 7 de Septiembre de tal manera que los valores de las constantes sean 0 (o 1 en el caso de radiaciones). Se puede emplear el mismo fichero de clima que para la identificación de la temperatura operativa. Es aconsejable fijar constantes los valores anteriores al día 1 de Septiembre para evitar un cambio brusco de temperatura (EnergyPlus realiza una presimulación de 15 días).

En este caso, necesitamos que el valor de consigna sea fijo durante toda la simulación, para ver cómo afecta la variación de temperatura sobre el consumo de la máquina en un valor fijo. Basta con crear un vector con longitud los 7 días x 24 horas y establecer las consignas constantes en un valor medio de por ejemplo 21 grados

```
SP(:,1:169) = 21;

HeatSP=SP(steptime);
ColdSP=SP(steptime);
```

De esta manera para cada paso de tiempo se asignará el valor correspondiente a la posición del vector de setpoints creado (en este caso similar para calor y frío). Se le añade 1 parámetro de más debido a que tiene en consideración el valor inicial, el cual debe ser fijado también en 21 grados para que desde el inicio se mantenga constante en un valor.

Pulsamos sobre el botón simular. Seleccionamos ahora las siguientes variables y las guardaremos con **'save selected plot'** en una carpeta: Outdoot Dry Bulb, Direct Solar, 319 Zone Mean Operative Temperature, Electricity:HVAC y ColdSP.

Con el script adjunto en el Anexo I del apartado Anexos, extraemos los datos de EnergyPlus y graficaremos los mismos para ver que se han exportado de manera correcta (*Véase Figura 31)*.

Con la información guardada en una carpeta con formato .mat, ejecutamos un segundo script para extraer las variables por separado. Finalmente, el script ejecutará el toolbox de identificación.

Una vez abierta la interfaz, en primer lugar cargaremos los datos de entrada de radiación solar y salida de temperatura operativa seleccionando para ello el botón *import data>time data*. En la parte de entrada escribimos **'u2'** y en la parte de salida escribimos

‘y2’. Dejamos por defecto el tiempo de muestreo y el inicio en 1. Una vez cargada la gráfica comprobamos que es la correcta marcando la casilla **‘time plot’**.

Realizamos el preproceso de quitar la media a la señal importada, de tal manera que podamos estimar correctamente el modelo.

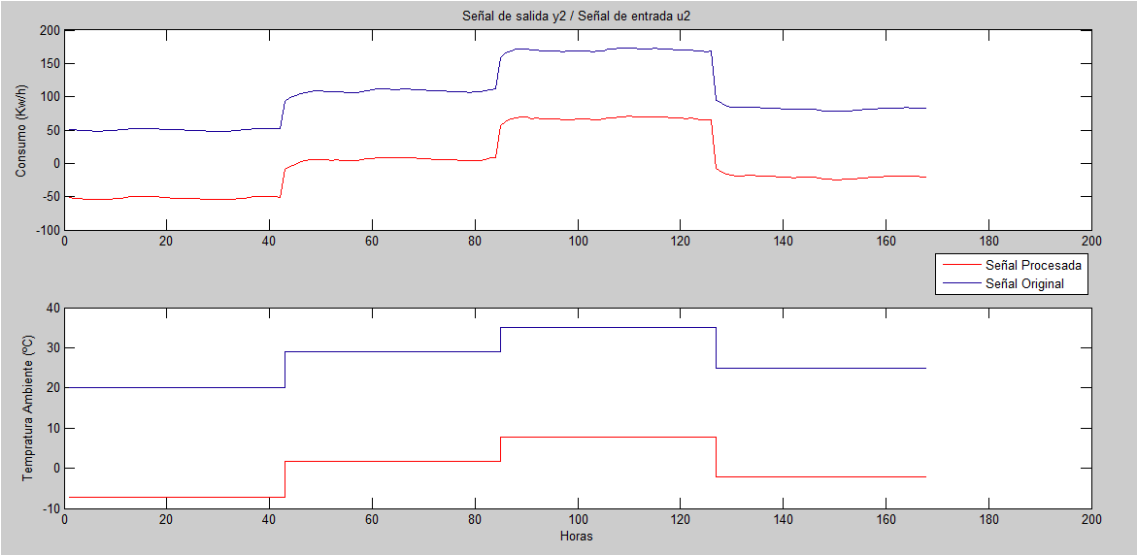

Figura 31. Señal original y Señal con media de 0.

Ahora únicamente nos queda la estimación del proceso, así que para ello arrastramos la nueva señal a la casilla de ‘**working data’** y a la casilla de **‘validation data’**. A continuación seleccionamos en *estimate>linear model* y se nos abrirá una nueva ventana donde podremos añadir polos y ceros para la estimación del modelo. De manera similar a la la identificación de la otra salida, el comportamiento de la temperatura ambiente sobre el sistema es de primer orden con retardo, así que dejamos por defecto la casilla **‘delay’** marcada.

Finalmente y con el modelo estimado seleccionado, marcamos la casilla **‘model output’** para comprobar el % de aproximación del modelo con el real. Como se observa en la figura 32, la coincidencia entre el modelo estimado y el real es de 77.71. Por lo tanto la coincidencia es bastante buena así que procedemos a validar los datos con otro conjunto de escalones de entradas.

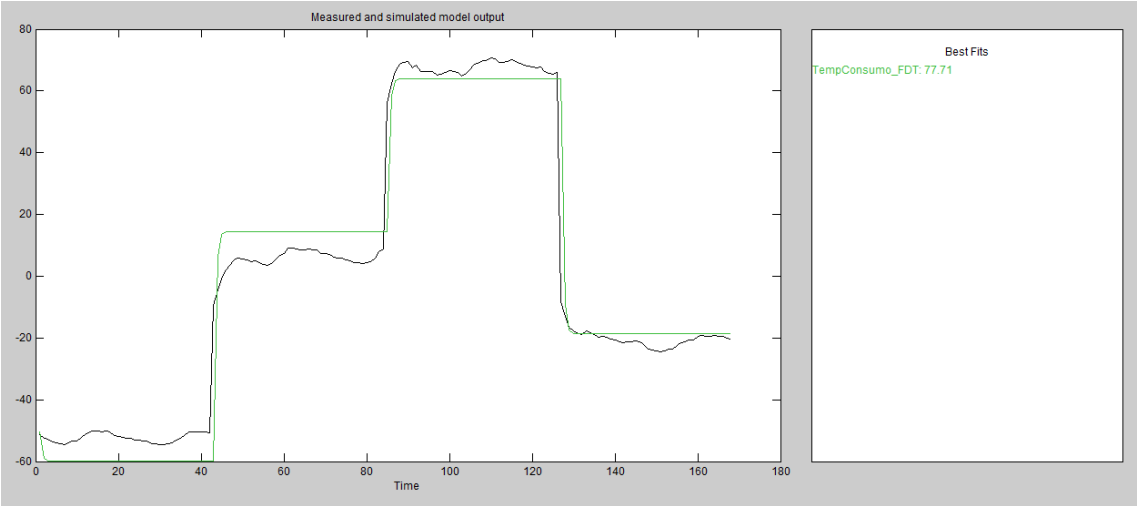

Figura 32. Estimación del modelo

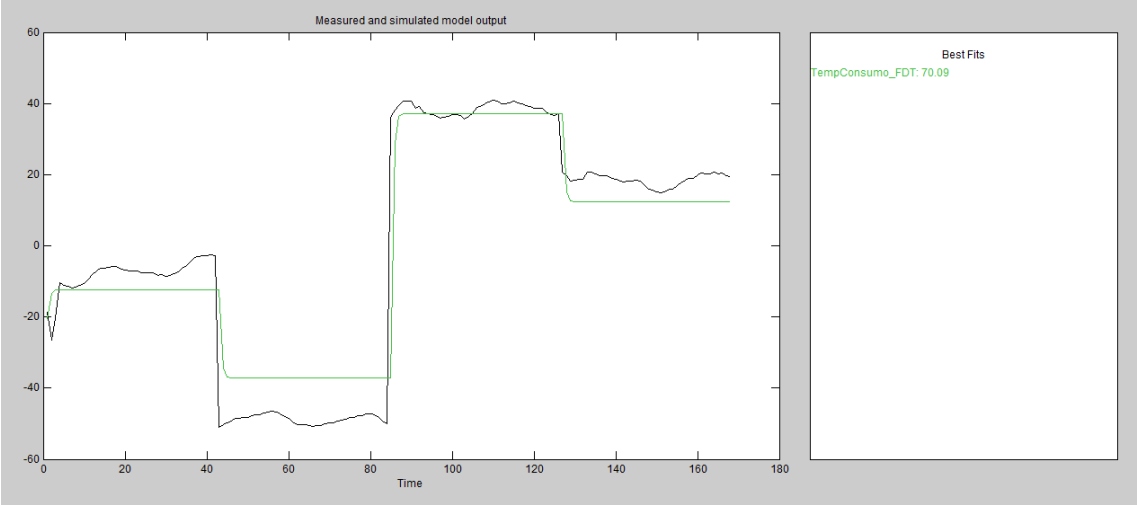

Figura 33. Validación del modelo con otros datos

En realidad existe una ligera diferencia del comportamiento de la temperatura sobre el consumo para un valor de 19 grados de consigna con respecto al otro extremo de 23 grados, pero el cambio es tan insignificante que se ha optado por escoger el valor medio de ambas y usarlo de manera global para todos los setpoints.

4.3.6. Identificación de Radiación Solar sobre el Consumo

Una vez convertido el fichero de clima estándar en formato CSV, lo abrimos con el Ron's editor y modificamos las columnas desde el día 1 de Septiembre hasta el 7 de Septiembre de tal manera que los valores de las constantes sean 0 (o 1 en el caso de radiaciones). Se puede emplear el archivo de clima utilizado en la identificación de temperatura operativa, de nuevo teniendo en consideración que durante las horas nocturnas el valor de la radiación es siempre 0 y que pese a que se fije el escalón en un valor constante, la influencia sobre la salida será mayor en las horas intermedias.

Con el MLEP ya configurado, dejamos el archivo de control de tal manera que la máquina de clima mantenga su valor de consigna en un valor fijo. Basta con crear un vector con longitud los 7 días x 24 horas y establecer las consignas para cada uno de ellos. Podemos utilizar el mismo fichero de control que en el apartado anterior.

De esta manera para cada paso de tiempo se asignará el valor correspondiente a la posición del vector de setpoints creado (en este caso similar para calor y frío). Se le añade 1 parámetro de más debido a que tiene en consideración el valor inicial.

Pulsamos sobre el botón simular. Seleccionamos ahora las siguientes variables y las guardaremos con **'save selected plot'** en una carpeta: Outdoot Dry Bulb, Direct Solar, 319 Zone Mean Operative Temperature, Electricity:HVAC y ColdSP.

Con el script adjunto en el Anexo I del apartado Anexos, extraemos los datos de EnergyPlus y graficaremos los mismos para ver que se han exportado de manera correcta (*Véase Figura 34)*.

Con la información guardada en una carpeta con formato .mat, ejecutamos un segundo script para extraer las variables por separado. Finalmente, el script ejecutará el toolbox de identificación.

Una vez abierta la interfaz, en primer lugar cargaremos los datos de entrada de radiación solar y salida de temperatura operativa seleccionando para ello el botón *import data>time data*. En la parte de entrada escribimos **'u3'** y en la parte de salida escribimos **'y2'**. Dejamos por defecto el tiempo de muestreo y el inicio en 1. Una vez cargada la gráfica comprobamos que es la correcta marcando la casilla **'time plot'**.

Realizamos el preproceso de quitar la media a la señal importada, de tal manera que podamos estimar correctamente el modelo.

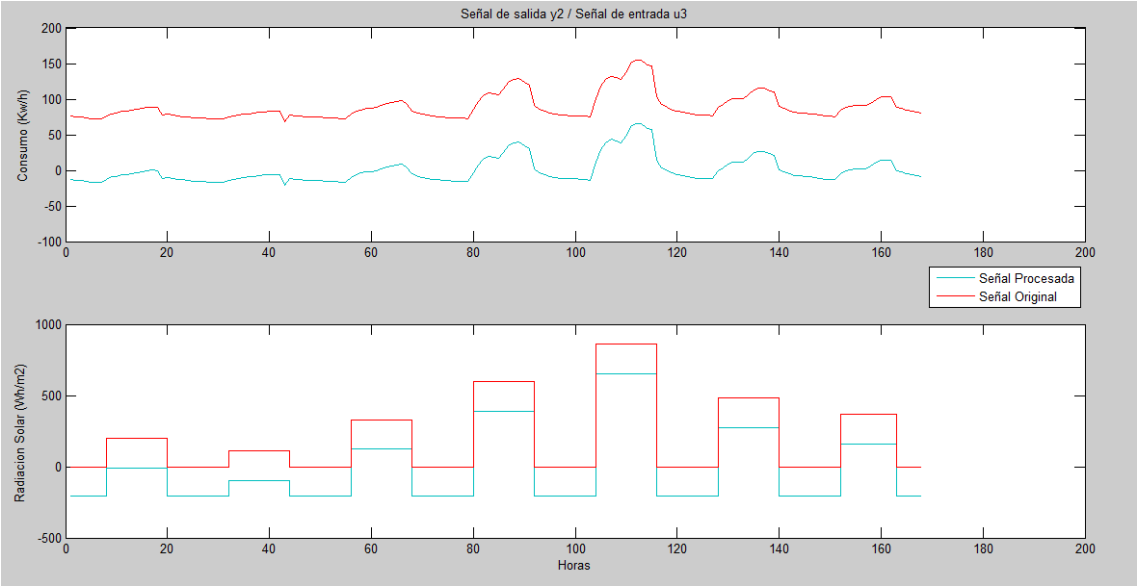

Figura 34. Señal procesada frente a la señal original

Ahora únicamente nos queda la estimación del proceso, así que para ello arrastramos la nueva señal a la casilla de '**working data'** y a la casilla de **'validation data'**. A continuación seleccionamos en *estimate>linear model* y se nos abrirá una nueva ventana donde podremos añadir polos y ceros para la estimación del modelo. En este caso, se nos plantea de nuevo la opción de usar un primer orden con retardo o con un cero. Seleccionamos la opción del cero ya que sabemos que nos va a estimar mejor el modelo y los picos de consumo.

Finalmente y con el modelo estimado seleccionado, marcamos la casilla **'model output'** para comprobar el % de aproximación del modelo con el real. Como se observa en la figura 35, la coincidencia entre el modelo estimado y el real es de 62.55.

Hay que tener en cuenta de nuevo que para un valor de consigna de 19 grados la influencia de la radiación será diferente a la de un valor de consigna de 23 grados. Pero de nuevo la variación es tan pequeña que se ha tomado un valor medio de 21 grados como en el caso anterior para cualquier setpoint dentro de ese rango.

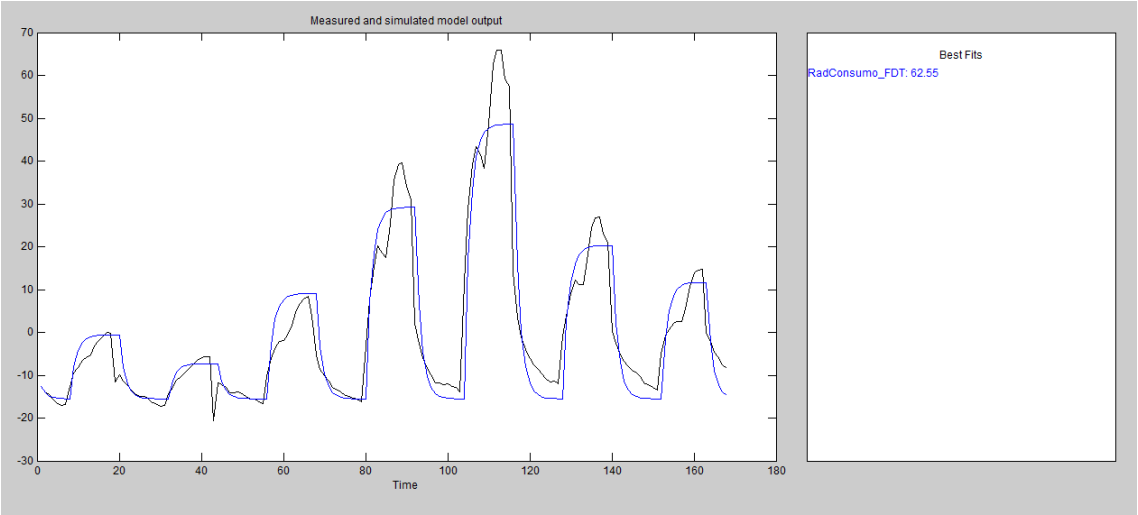

Figura 35. Gráfica del modelo estimado

Como se puede observar, la identificación no es del todo buena (se ha estimado con un cero). No obstante, identificar los picos de consumo correspondientes se antoja un tanto complicado y como sigue tendencias lo daremos por buena ya que no vamos a obtener un resultado mejor con un modelo lineal.

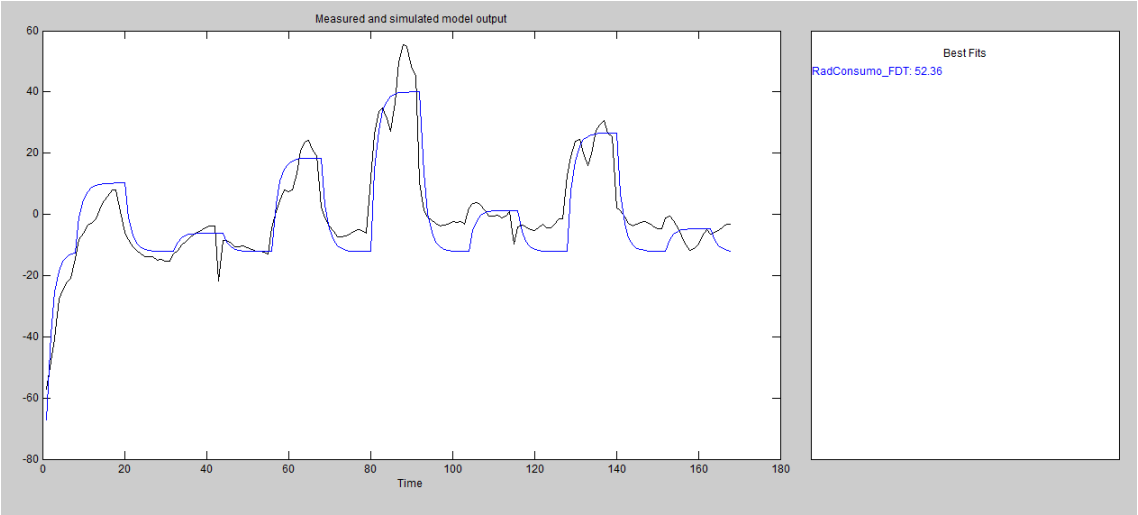

Figura 36. Validación del modelo estimado

La validación tiene un porcentaje de similitud del 52.36 %. No es un valor demasiado bueno que digamos. No obstante, no se puede mejorar la identificación a no ser que empleemos un modelo no lineal. De todas formas, la precisión del modelo del consumo no tiene que ser excesivamente buena (con que siga más o menos los valores y tendencias es suficiente para aplicar la optimización del algoritmo).

4.3.7. Identificación de Setpoint sobre el Consumo

Una vez convertido el fichero de clima estándar en formato CSV, lo abrimos con el Ron's editor y modificamos las columnas desde el día 1 de Septiembre hasta el 7 de Septiembre de tal manera que los valores de las constantes sean 0 (o 1 en el caso de radiaciones). Se puede emplear el archivo de clima utilizado en la identificación de temperatura operativa, que no es más que un fichero con todas las variables constantes en un valor.

Con el MLEP ya configurado, dejamos el archivo de control de tal manera que la máquina de clima genere una serie de escalones de entrada (pueden ser los mismos que para la identificación de la temperatura). Basta con crear un vector con longitud los 7 días x 24 horas y establecer las consignas para cada uno de ellos.

De esta manera para cada paso de tiempo se asignará el valor correspondiente a la posición del vector de setpoints creado (en este caso similar para calor y frío). Se le añade 1 parámetro de más debido a que tiene en consideración el valor inicial.

Pulsamos sobre el botón simular. Seleccionamos ahora las siguientes variables y las guardaremos con **'save selected plot'** en una carpeta: Outdoot Dry Bulb, Direct Solar, 319 Zone Mean Operative Temperature, Electricity:HVAC y ColdSP.

Con el script adjunto en el Anexo I del apartado Anexos, extraemos los datos de EnergyPlus y graficaremos los mismos para ver que se han exportado de manera correcta (*Véase Figura 37)*.

Con la información guardada en una carpeta con formato .mat, ejecutamos un segundo script para extraer las variables por separado. Finalmente, el script ejecutará el toolbox de identificación.

Una vez abierta la interfaz, en primer lugar cargaremos los datos de entrada de radiación solar y salida de temperatura operativa seleccionando para ello el botón *import data>time data*. En la parte de entrada escribimos **'u1'** y en la parte de salida escribimos **'y1'**. Dejamos por defecto el tiempo de muestreo y el inicio en 1. Una vez cargada la gráfica comprobamos que es la correcta marcando la casilla **'time plot'**.

Realizamos el preproceso de quitar la media a la señal importada, de tal manera que podamos estimar correctamente el modelo.

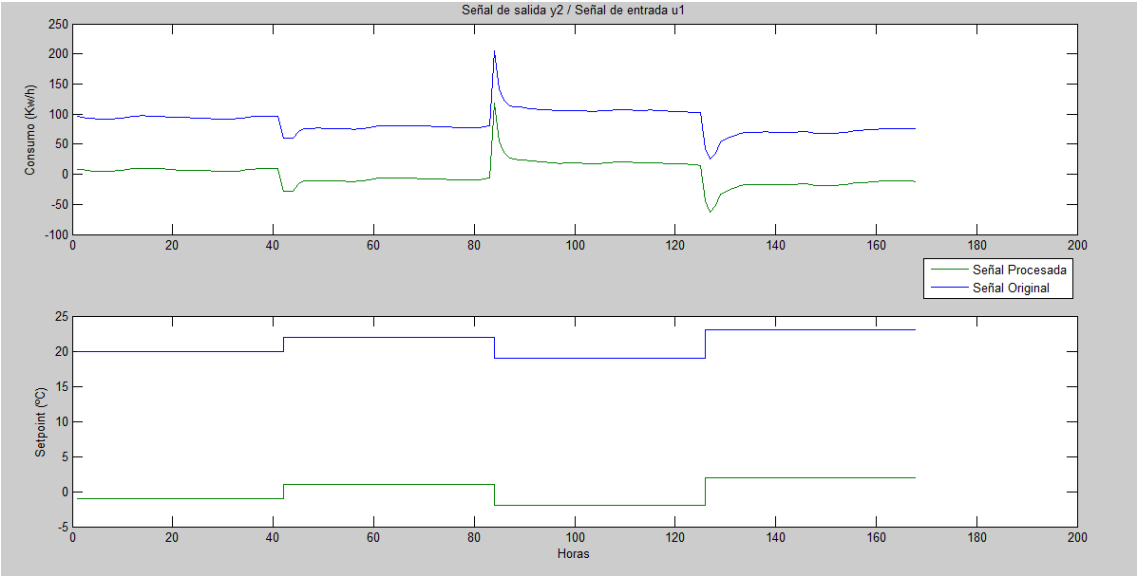

Figura 37. Señal procesada y original

Ahora únicamente nos queda la estimación del proceso, así que para ello arrastramos la nueva señal a la casilla de '**working data'** y a la casilla de **'validation data'**. A continuación seleccionamos en *estimate>linear model* y se nos abrirá una nueva ventana donde podremos añadir polos y ceros para la estimación del modelo. Se observa claramente como el comportamiento del sistema precisa de un cero, por lo que marcamos la casilla correspondiente y ejecutamos el algoritmo de estimación.

Finalmente y con el modelo estimado seleccionado, marcamos la casilla **'model output'** para comprobar el % de aproximación del modelo con el real. Como se observa en la figura 38, la coincidencia entre el modelo estimado y el real es de 70.66. De nuevo se observa un buen porcentaje, así que daremos por buena la identificación.

Se observa que la peor parte es la identificación del cambio de consigna. Esos picos se deben a que la máquina cambia de estado de trabajo de manera que cuando cambiamos el setpoint ésta se da prisa en alcanzar dicho valor. Para el caso de cambio a un valor más elevado, la máquina no se apaga del todo sino que reduce su potencia, produciendo una ligera oscilación.

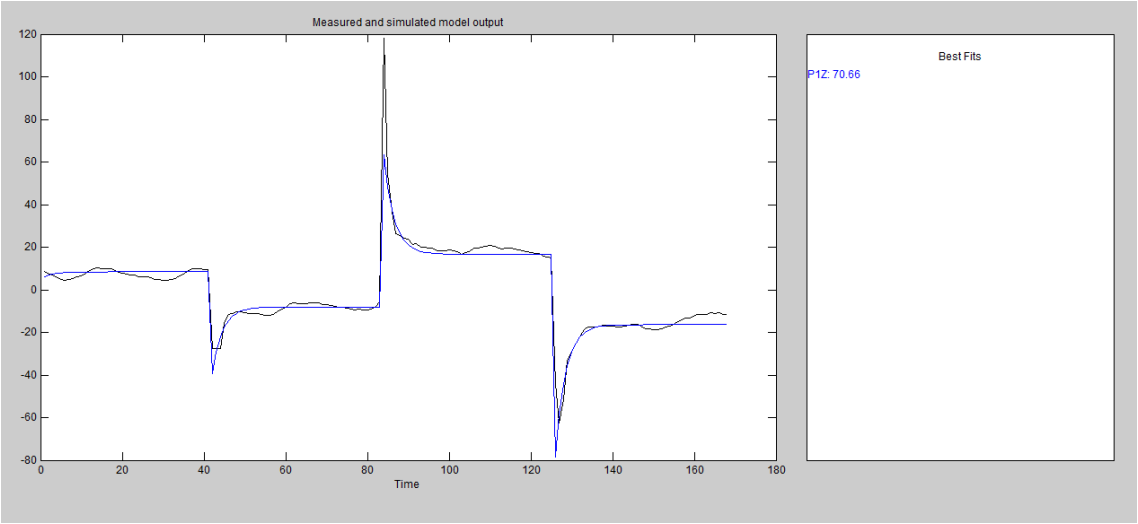

Figura 38. Estimación del modelo

Para validar los datos, empleamos un nuevo conjunto de datos de setpoint, siendo los resultados obtenidos representados en la Figura 32. Se observa una estimación normal, aunque las tendencias son buenas, por lo que damos por válida la identificación.

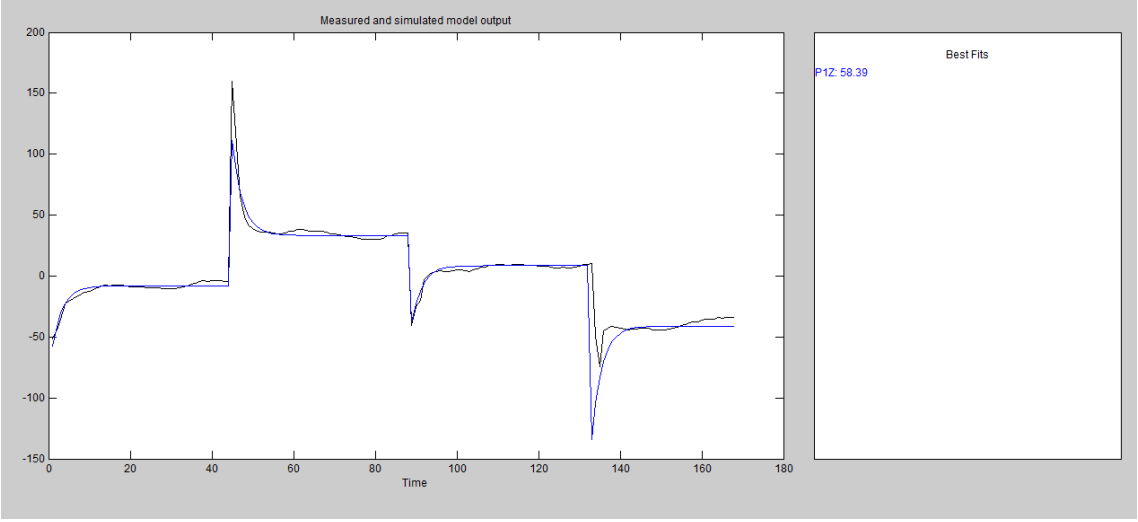

Figura 39. Validación del modelo

4.3.8. Identificación del arranque de la máquina HVAC sobre la temperatura operativa y sobre el consumo

Adicionalmente a las 6 identificaciones anteriores, se ha visto necesario realizar dos identificaciones más debido al comportamiento del sistema cuando se produce un cambio de estado de ON/OFF. Los picos de temperatura y de consumo dependerán directamente del setpoint que se encuentre establecido en ese momento, aunque el rango en el que nos movemos es bastante similar en cuanto a los valores a la hora de apagar la máquina, existen ligeras diferencias que hacen que el modelo sea menos preciso.

El problema principal reside en que no es lo mismo un apagado con 25 grados que con 40 grados, ya que en el segundo caso la temperatura operativa intentará alcanzar el valor de 40 grados de manera mucho más rápida que con 25. Para solucionar esto, se ha realizado la identificación a la hora de cambiar el setpoint de encendido a apagado (recordemos que el apagado de la refrigeración se realiza seleccionando un valor muy alto de consigna). Para abarcar la mayoría de valores de temperatura, se ha establecido el valor de temperatura ambiente en un punto muy alto.

Por tanto, esta identificación se antoja un tanto necesaria para mejorar el comportamiento del modelo simulado y dotarlo de un mayor realismo. Así pues, se seguirán los pasos anteriormente descritos en primer lugar para el consumo.

Una vez convertido el fichero de clima estándar en formato CSV, lo abrimos con el Ron's editor y modificamos las columnas desde el día 1 de Septiembre hasta el 7 de Septiembre de tal manera que los valores de las constantes sean 0 (o 1 en el caso de radiaciones). Se puede emplear el archivo de clima utilizado en la identificación del setpoint, cambiando el valor de temperatura ambiente a un valor muy alto (>35 ºC), que no es más que un fichero con todas las variables constantes en un valor.

Con el MLEP ya configurado, dejamos el archivo de control de tal manera que la máquina se vaya apagando y encendiendo. Como en este caso no nos interesa identificar valores distintos de escalones, suponemos un único escalón el cual corregiremos con un offset posteriormente en el Simulink.

De esta manera para cada paso de tiempo se asignará el valor correspondiente a la posición del vector de setpoints creado (en este caso similar para calor y frío). Se le añade 1 parámetro de más debido a que tiene en consideración el valor inicial.

Pulsamos sobre el botón simular. Seleccionamos ahora las siguientes variables y las guardaremos con **'save selected plot'** en una carpeta: Outdoot Dry Bulb, Direct Solar, 319 Zone Mean Operative Temperature, Electricity:HVAC y ColdSP.

Con el script adjunto en el Anexo I del apartado Anexos, extraemos los datos de EnergyPlus y graficaremos los mismos para ver que se han exportado de manera correcta (*Véase Figura 40)*.

Con la información guardada en una carpeta con formato .mat, ejecutamos un segundo script para extraer las variables por separado. Finalmente, el script ejecutará el toolbox de identificación.

Una vez abierta la interfaz, en primer lugar cargaremos los datos de entrada de radiación solar y salida de temperatura operativa seleccionando para ello el botón *import data>time data*. En la parte de entrada escribimos **'u1'** y en la parte de salida escribimos **'y1'**. Dejamos por defecto el tiempo de muestreo y el inicio en 1. Una vez cargada la gráfica comprobamos que es la correcta marcando la casilla **'time plot'**.

Realizamos el preproceso de quitar la media a la señal importada, de tal manera que podamos estimar correctamente el modelo.

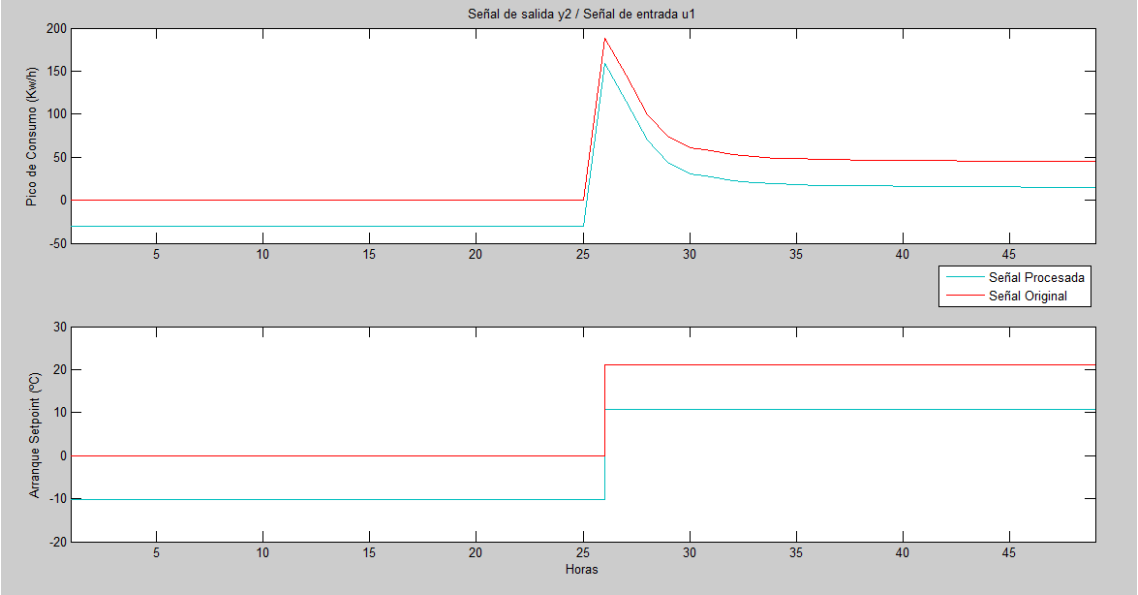

Figura 40. Señal procesada y señal original (solo un escalón).

Ahora únicamente nos queda la estimación del proceso, así que para ello arrastramos la nueva señal a la casilla de '**working data**' y a la casilla de **'validation data'**. A continuación seleccionamos en *estimate>linear model* y se nos abrirá una nueva ventana donde podremos añadir polos y ceros para la estimación del modelo. Se

observa claramente como el comportamiento del sistema precisa de un cero, por lo que marcamos la casilla correspondiente y ejecutamos el algoritmo de estimación.

Finalmente y con el modelo estimado seleccionado, marcamos la casilla **'model output'** para comprobar el % de aproximación del modelo con el real. Como se observa en la figura 41, la coincidencia entre el modelo estimado y el real es de 80.54. Observamos como el resultado es satisfactorio y el consumo se ha identificado correctamente.

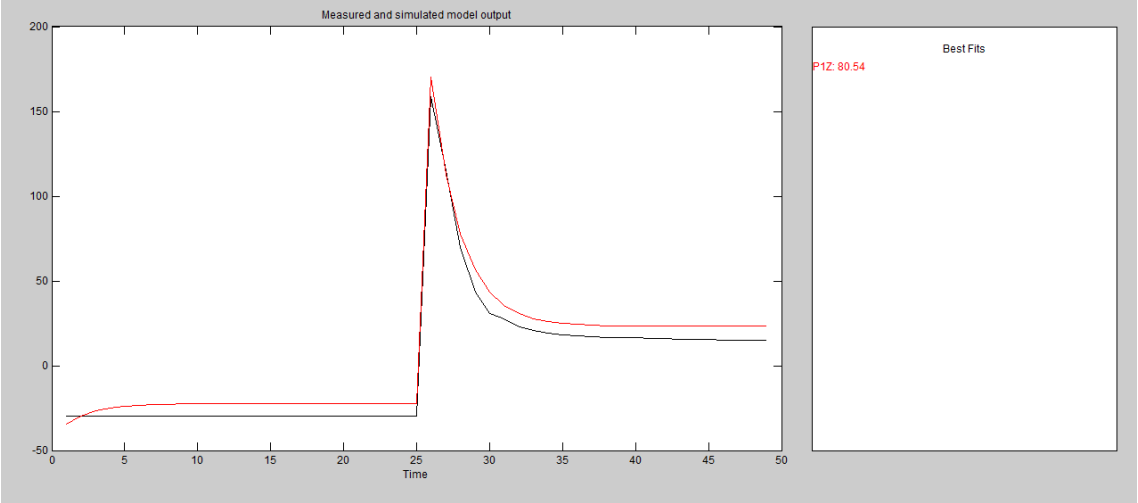

Figura 41. Estimación del modelo

Aunque el resultado no hubiese sido del todo bueno, solo nos interesa que identifique correctamente el pico de subida, ya que será la parte con la que nos quedaremos a la hora de construir el consumo final del modelo (recordemos que dependerá de diferentes estados).

Análogamente al consumo, realizamos los mismos pasos para identificar el pico de temperatura. De nuevo hay que resaltar que la recuperación de temperatura al valor ambiente es bastante similar en el rango en el que nos movemos, por lo que seleccionando un valor alto de temperatura ambiente alto y un setpoint medio de 21 grados conseguiremos el cambio de temperatura deseado. Los resultados se pueden ver en las Figuras 42 y 43.

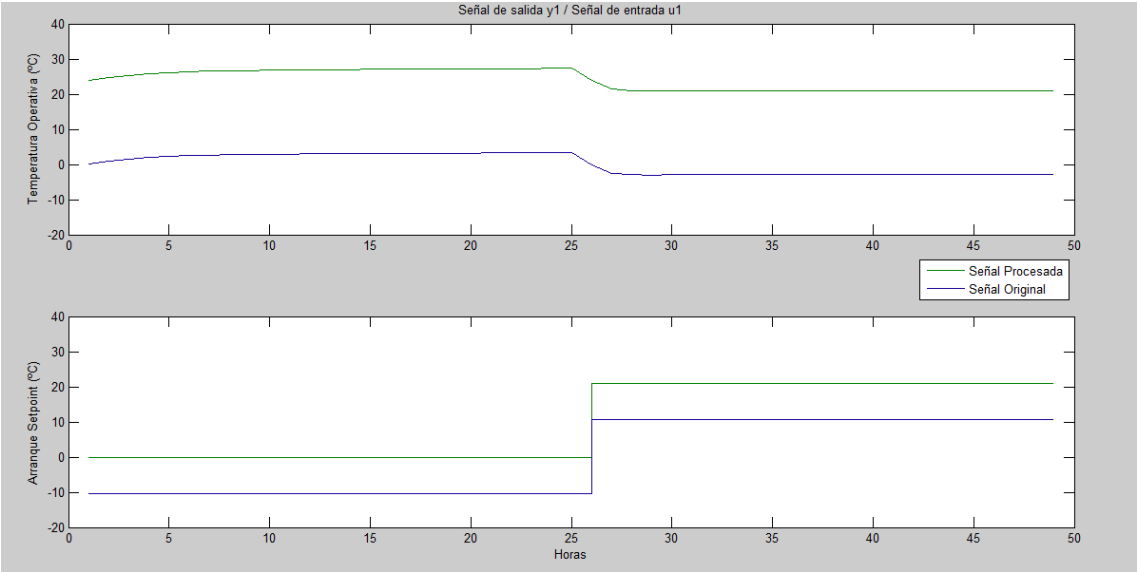

Figura 42. Señal procesada y real de la temperatura operativa.

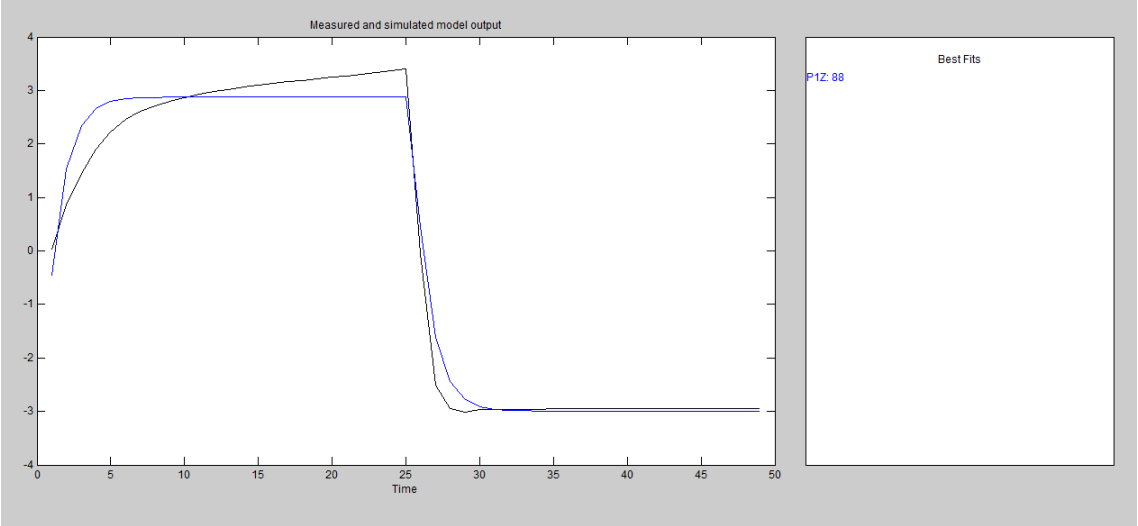

Figura 43. Modelo estimado para la temperatura operativa.

Cabe señalar que estos dos estados únicamente se producirán cuando se realice un cambio de ON/OFF de la máquina de climatización, de manera que el comportamiento normal vendrá dado por las identificaciones de las entradas. Con un bucle de evaluación del estado, se seleccionará la salida de consumo original o de arranque y del mismo modo con la temperatura, elaborando de esta manera la gráfica final resultante del modelo simulado.

4.3.9. Extracción de las Funciones de Transferencia

El formato de guardado por defecto para los modelos estimados es un formato de tipo <idproc>. Si hacemos clic sobre cada uno de ellos, veremos que integra diferentes parámetros entre los que se encuentran los valores de ganancia, retardo, parte derivativa, etc…

Para extraer dichas características emplearemos las siguientes líneas de comando:

```
Kp  = (nombre de la funcion).Kp.value;
Td* = (nombre de la funcion).Td.value;
Tp  = (nombre de la funcion).Tp1.value;
Tz* = (nombre de la funcion).Tz.value;
```

*para los valores de Td y Tz, dependerá si la función tiene retardo o algún cero, por lo que los sistemas tendrán o una u otra en función de lo estimado.

Una vez obtenidos los parámetros, construimos las funciones de transferencia con el comando '**tf'**, empleando la extensión <iodelay> para introducir el retardo a las FDTs. En el *Anexo IV* encontramos más información acerca de la construcción de las funciones.

La tabla 2 refleja las G obtenidas durante el proceso de modelado del sistema.

$g11 = Kp11/(1 + Tp11 * s)$	$g12 = Kp12/(1 + Tp12 * s)$
$g13 = Kp13/(1 + Tp13 * s)$	$g14 = Kp14 * (1 + Tz14 * s)/(1 + Tp14 * s)$
$g21 = Kp21 * (1 + Tz21 * s)/(1 + Tp21 * s)$	$g22 = Kp22/(1 + Tp22 * s)$

$g23 = Kp23 * (1 + Tz23 * s)/(1 + Tp23 * s)$	$g24 = Kp24 * (1 + Tz24 * s)/(1 + Tp24 * s)$

Tabla 2. Funciones de transferencia del sistema

4.4. Construcción del Modelo

A la hora de elaborar el modelo que se va a emplear en el algoritmo evolutivo, con el fin de optimizar el consumo en un rango de temperatura determinado, emplearemos la herramienta Simulink que viene incorporada junto a Matlab.

Gracias a Simulink, podremos realizar el diagrama de bloques que componen el sistema, simular las entradas en un determinado espacio de tiempo y exportar las variables al workspace para poder trabajar con ellas.

Debido a que el comportamiento del sistema no empieza desde un valor inicial de 0, sino que parte de un estado previo (en EnergyPlus realiza una simulación previa), necesitaremos emplear un **'espacio de estados'** en vez de una función de transferencia para cada una de las entradas al sistema.

4.4.1. Espacio de Estados

Un espacio de estados no es más que un modelo matemático empleado en ingeniería de control. En su conjunto lo forma una serie de entradas y salidas junto con una variable de estado, relacionadas entre sí por medio de ecuaciones diferenciales de primer orden.

El conjunto de entradas y salidas se almacenan de manera vectorial. Si n es el número de entradas y m el número de salidas, necesitaríamos $n\,x\,m$ veces la transformada de Laplace para procesar la información del sistema.

Las variables de estado que forman el sistema son un subconjunto más pequeño de variables que pueden representar su comportamiento y estado dinámico en un

instante t. Son linealmente independientes entre sí. Si queremos representar una función de transferencia, el número de variables de estados es igual al orden del denominador.

Las ecuaciones que definen al sistema de estados son las siguientes:

$$\partial x = A(t)x(t) + B(t)u(t)$$

$$y(t) = C(t)x(t) + D(t)U(t) \qquad (1)$$

Donde x es el vector de estados, y es el vector de salida, u el vector de control y los parámetros $A, B, C\ y\ D$ son matrices (por simplificación se considera D=0).

Por tanto, necesitaremos realizar una transformación previa de las funciones de transferencia extraídas en las identificaciones a espacios de estados para poder usarlas en Simulink. Para realizar el cambio emplearemos la siguiente línea de código para cada una de las funciones de transferencia:

```
[a,b,c,d]=tf2ss(num,den);
```

donde $a, b, c\ y\ d$ son las matrices de parámetros del espacio de estados y $num\ y\ den$ son el numerador y denominador de la FDT a transformar. Dentro del *Anexo IV* se encuentran detalladas las transformaciones al completo.

4.4.2. Obtención de estados iniciales y offset.

Como ya se comentó con anterioridad, el sistema no parte de un estado de 'reposo' en el que los valores iniciales son 0, sino que viene de una temperatura y consumo anteriores. Para calcular el estado inicial de la simulación, EnergyPlus emplea una presimulación de 15 días de manera que determina el comportamiento del sistema sin ningún problema.

Por otro lado, en las identificaciones se utilizó un preproceso para aclimatar las señales de entrada/salida que consistió en quitarles la media de la función para dejarla en 0. Por esa misma razón, a la hora de simular el proceso, las entradas al sistema tienen que tener dicho preproceso para simular correctamente la salida. No obstante, la señal resultante necesitará de sumarse un offset para ajustarla a su valor real (calcular el punto de funcionamiento del sistema).

Para la obtención del estado inicial, necesitaremos los dos vectores de salida del sistema real, por lo que hará falta realizar una simulación en EnergyPlus para extraer

dichos parámetros y emplearlos en Simulink. Bastará con realizar una simulación estándar de una semana y extraer las funciones de entrada y salida que emplearemos en el modelo simulado. A continuación, aplicaremos el comando **'detrend'** para dejar el valor medio en 0 (también para las entradas) tanto del consumo como de la temperatura operativa, y extraemos el primer valor del vector de datos, el cual lo emplearemos como estado inicial en el bloque de espacio de estados (hará falta realizar un ajuste en el mismo).

Para obtener los offsets, simplemente extraemos el valor inicial de las funciones de salida originales para sumárselo a la suma de salidas para cada una de las entradas. En el caso de la salida de la consigna, emplearemos un offset correspondiente al valor inicial del vector de entradas de consignas generado a posteriori por el algoritmo.

La extracción de dichos parámetros se encuentra reflejada y comentada en el *Anexo IV*.

4.4.3. Construcción del diagrama de bloques

Una vez abierto Simulink, para la construcción del diagrama de bloques necesitaremos los siguientes:

- Bloque **'From WorkSpace'**: Con este bloque introduciremos las entradas del sistema. Solo acepta una matriz con los vectores de tiempo y valores de la variable.
- Bloque **'State-Space':** Este es el bloque que emplearemos para introducir las variables de estado.
- Bloque **'Transport Delay':** Para evitar problemas, para las funciones que tengan delay emplearemos este bloque con un estado inicial.
- Bloque **'Sum':** Este bloque nos permitirá sumar las diferentes entradas y los valores de offset a una señal.
- Bloque **'Variable':** Con este bloque se pueden recoger directamente variables del Workspace. Se emplea para introducir los Offsets.
- Bloque **'To WorkSpace':** Se emplea para guardar las señales de salidas en el Workspace.
- Bloque **'Scope'**: Bloque opcional que nos permite ver una representación gráfica del estado del sistema donde esté situado. Se puede emplear para extraer variables al WorkSpace.

Para realizar una mejor organización, crearemos dos subsistemas: Uno para la temperatura operativa y otro para el consumo. Dentro de cada uno de ellos irán las 3 variables que afectan al sistema. Hay que recordar que dos de ellas permanecen invariables y que se obtienen por medio de predicciones meteorológicas (en nuestro caso por medio del fichero de clima).

En el subsistema de Temperatura, crearemos dos líneas de trabajo, una para extraer la temperatura operativa a partir de la temperatura ambiente y la radiación, y la otra para obtener los picos de temperatura cuando se produce un cambio de estado ON/OFF en la máquina HVAC. Se ha dejado una tercera línea opcional con la temperatura operativa cuando el setpoint está activo en un valor. No obstante, como ya se comentó anteriormente, el cambio de temperatura entre 19 y 23 grados se produce en pasos inferiores a 1 hora, por lo que resulta más preciso seleccionar directamente el valor de setpoint como valor de temperatura final. El esquema resultante se muestra en la Figura 44. Recordemos que los bloques de 'Transport Delay' se añadirán solo a los sistemas con retardo.

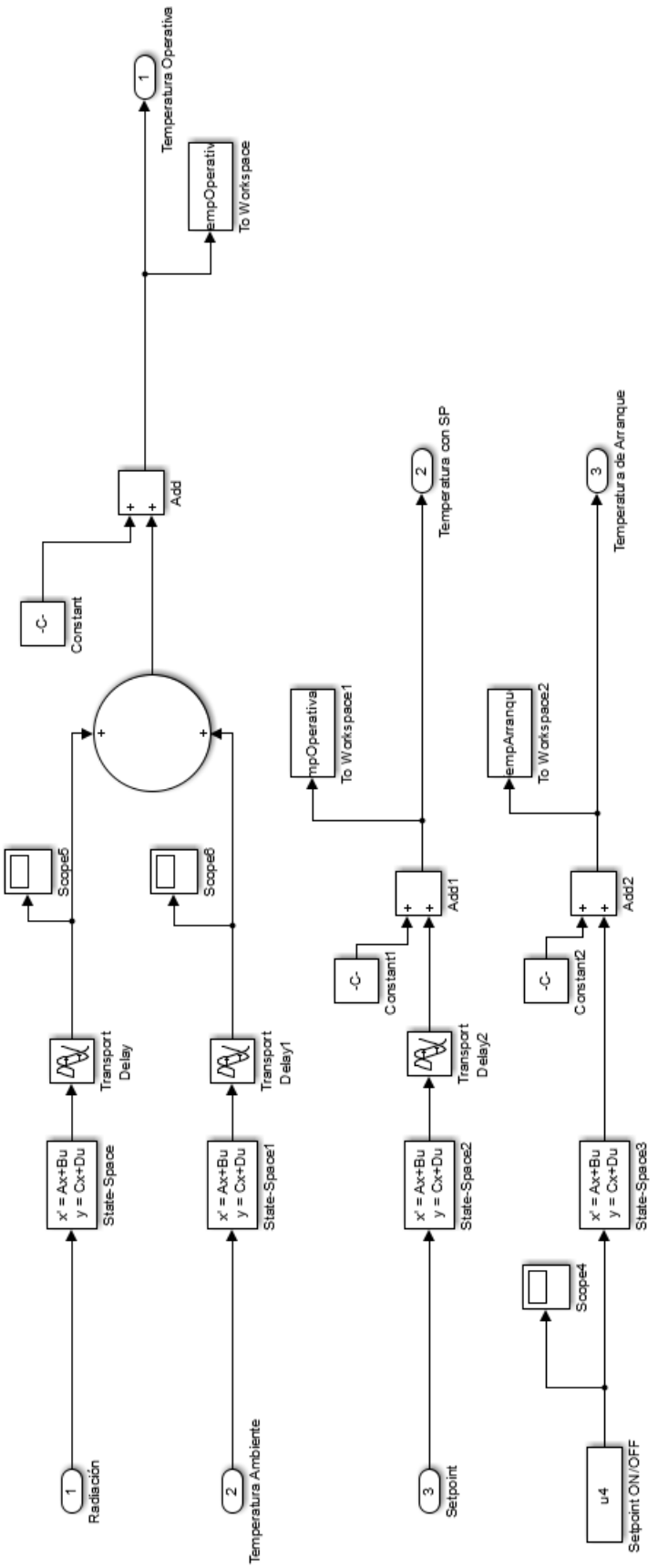

Figura 44. Esquema de Temperatura Operativa del modelo simulado

Como se puede observar, tenemos 3 tipos de temperatura que se extraen a Matlab. El valor final de temperatura dependerá del estado en el que nos encontremos, siendo definido de la siguiente manera:

- Máquina Apagada: En este estado, la salida de temperatura es la suma de la temperatura ambiente junto a la radiación solar.
- Máquina Encendida: La salida de temperatura es directamente el valor de la consigna fijado. Se puede tomar el valor de Simulink (más impreciso) o seleccionar directamente el valor de consigna.
- Cambio ON/OFF: Se produce cuando se apaga o enciende la máquina. En este caso el valor de temperatura es igual a la rampa de subida o bajada siempre y cuando no supere el valor de la temperatura operativa con la máquina apagada (*Véase 4.3.8. Identificación del arranque de la máquina HVAC sobre la temperatura operativa y sobre el consumo)*

A la hora de ejecutar el algoritmo de optimización, se realizará una máquina de estados por medio de funciones para seleccionar uno u otro valor según corresponda (sistema de prioridades).

Los valores de Offset son los siguientes:

- Offset Temperatura Operativa: Es el valor inicial del vector de salida de Temperatura Operativa con la máquina apagada extraída de EnergyPlus.
- Offset Temperatura con SP: En este caso el offset es directamente el primer valor del vector de consignas dado por el algoritmo (valor inicial fijado en 20).
- Offset Temperatura de Arranque: Para que la rampa de temperatura parta del valor inicial correcto, se le suma el setpoint anterior al apagado de la máquina para que parta de dicho valor.

En el subsistema de Consumo, la premisa es la misma que para la Temperatura. Emplearemos los bloques de '**state-space'** para cada una de las funciones de entrada (incluyendo la de arranque del consumo), junto con el bloque de delay si es necesario.

El resultado final se refleja en la Figura 45. Como podemos ver, en este caso la suma del consumo final es producto de las 4 entradas juntas. El valor de arranque se ha introducido para mejorar la precisión del sistema, ya que debido a la dinámica de la misma, la rampa de consumo al encender la máquina sobrepasaba al consumo original, provocando fallos e imprecisiones en el modelo.

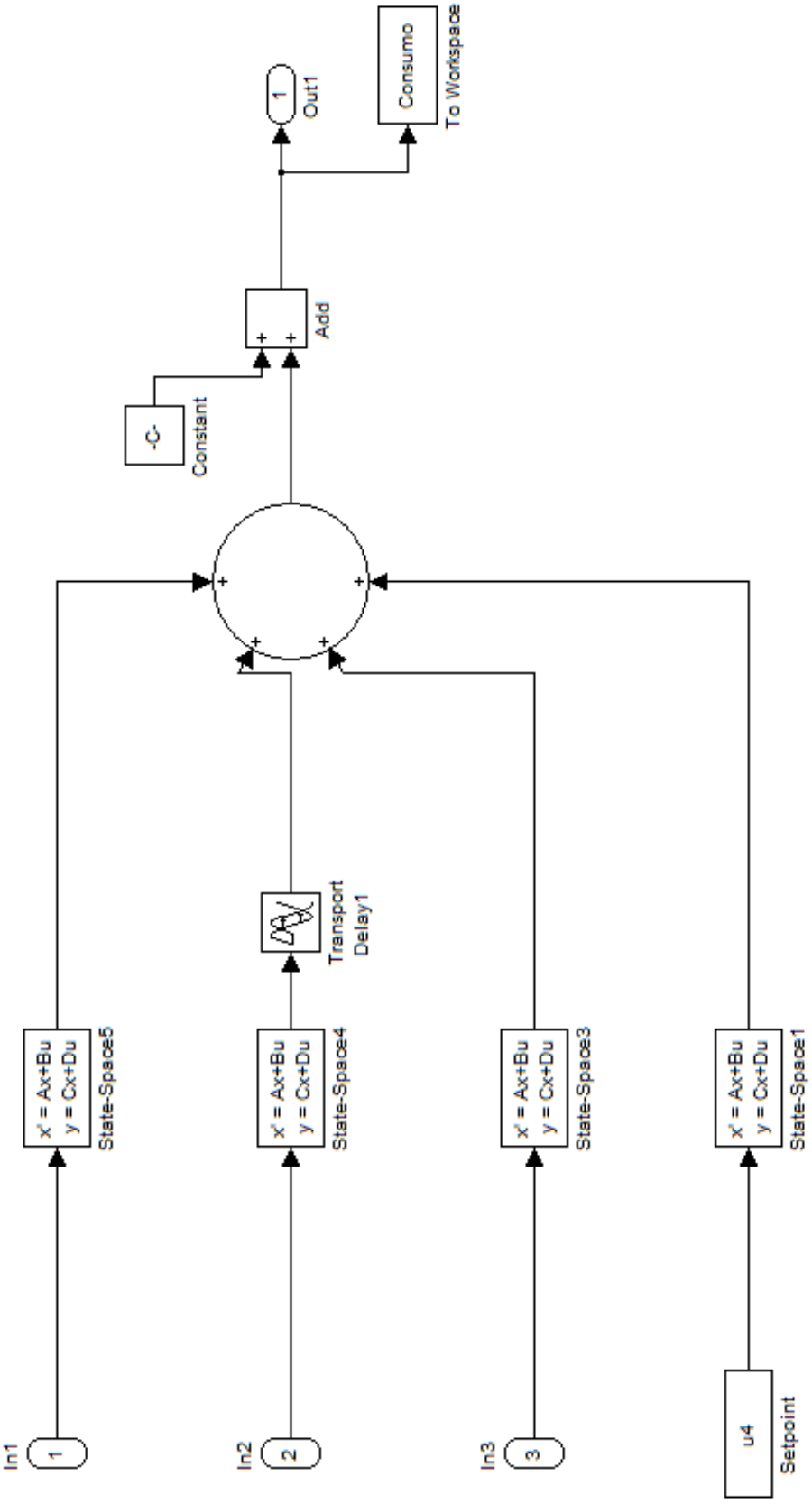

Figura 45. Subconjunto de Consumo del modelo simulado

En este caso contamos con dos estados distintos de consumo, definidos de la siguiente forma:

- Consumo Final: Es el consumo resultante de sumar la influencia de las 3 entradas. Si el Setpoint es cero, la salida total será cero. Se ha introducido la rampa de consumo para simular la dinámica del sistema a la hora de encender la máquina.
- Consumo de Arranque: Se utiliza para identificar los picos de arranque de consumo de la máquina HVAC cuando se encuentra en reposo.

De la misma manera que para la temperatura, la salida de consumo final dependerá de una máquina de estados ejecutada con el algoritmo de optimización.

Los valores de Offset son los siguientes:

- Offset Consumo Final: Es el valor inicial del consumo de salida obtenido al realizar la simulación previa en EnergyPlus. Calcula el horizonte de predicción.
- Offset Consumo de Arranque: Es el mismo Offset que para el consumo final con un pequeño margen para evitar errores entre un consumo y otro (mismo valor en la máquina de estados).

Para definir el estado inicial, hace falta resolver la ecuación (1). Para ello dividimos el valor inicial de la señal de salida correspondiente con la media quitada entre el parámetro c de la ecuación de estados (donde $d = 0$).

4.4.4. Resultados del modelo vs real

Para ver si nuestro modelo es correcto, realizaremos una serie de pruebas con las que observaremos si las salidas del modelo simulado se asemejan a las del modelo real.

En primer lugar realizaremos una simulación en EnergyPlus de unos días de verano cualesquiera. Una vez simulada, extraeremos los datos a ficheros .m que cargaremos con el código adjunto del Anexo IV, añadiendo la entrada de SP que estará definida en EnergyPlus (comentada en el código).

Para una sucesión de 3 días típica de Julio, los resultados de Temperatura con la máquina apagada son los representados en la Figura 46 (el consumo es 0).

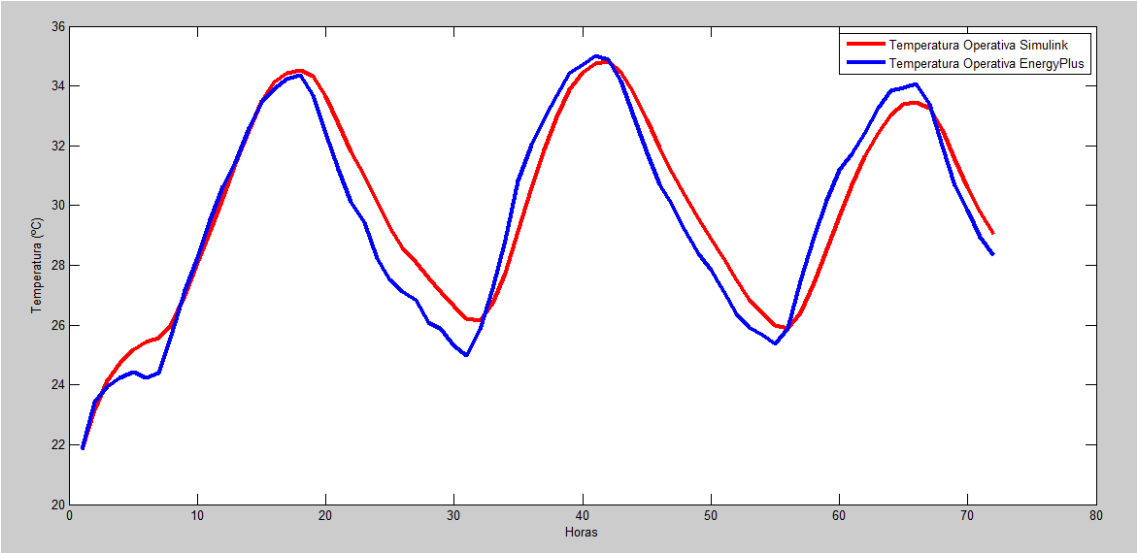

Figura 46. Comparación de temperaturas.

Como podemos observar, el resultado es bastante bueno. La señal simulada sigue tendencias y se aproxima bastante a la señal real. No obstante, se observa un pequeño error en algunas partes del modelo en los que el modelo simulado no es capaz de actuar de manera similar al real, debido a imprecisiones en la identificación del mismo.

Si introducimos una serie de escalones de consigna, la temperatura operativa de salida, pasando por la máquina de estados, es la mostrada en la Figura 47. Se ve que prácticamente están superpuestas.

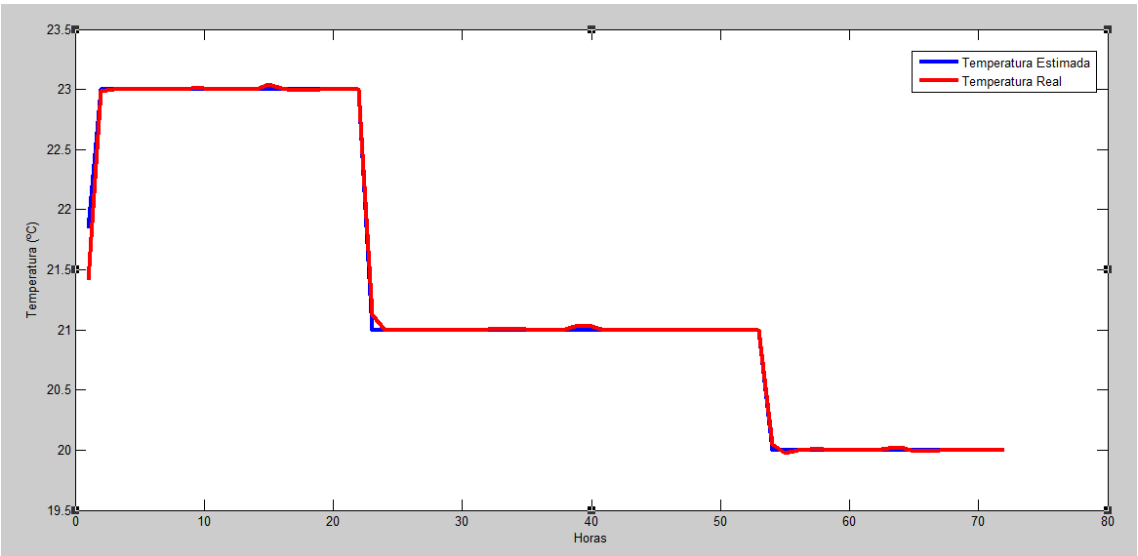

Figura 47. Temperatura de consigna real y estimada

En el caso del consumo, el resultado final se muestra en la Figura 48. Al igual que la temperatura, tiene que construirse mediante la máquina de estados.

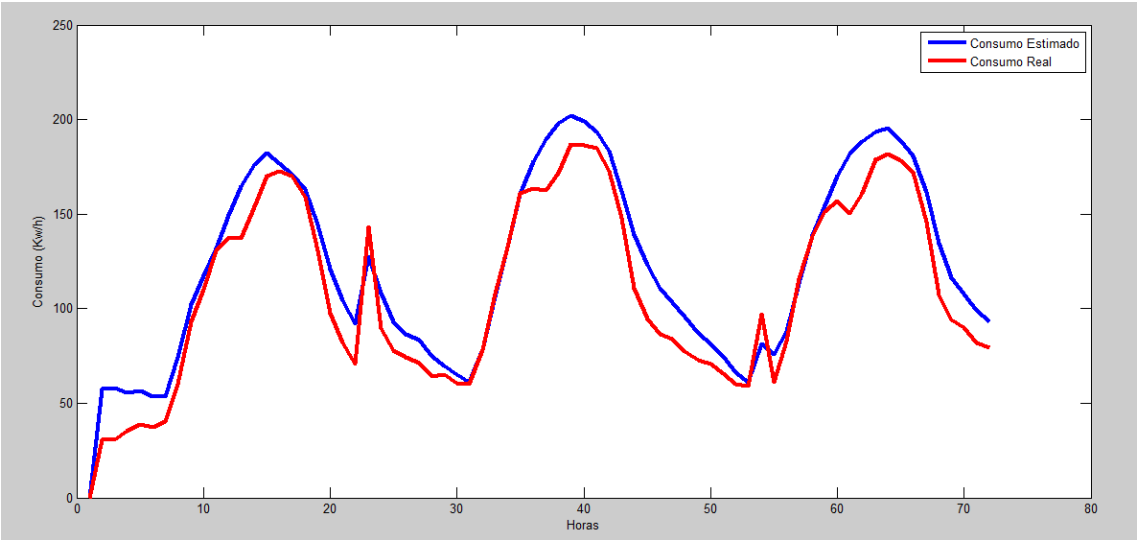

Figura 48. Consumo Estimado frente al real

Como podemos observar, las tendencias son muy parecidas, incluso los picos de cambio de Setpoint los identifica correctamente. Existen diferencias propias de la imprecisión del modelo debido a la identificación. No obstante, el resultado es bastante bueno.

De manera similar, podemos identificar de igual manera el cambio de estado ON/OFF de la máquina HVAC con la línea adicional de arranque añadida en Simulink. Si introducimos una sucesión de Setpoints intercalados con apagados y encendidos aleatorios, observamos que se asemeja bastante a la realidad. Los resultados obtenidos se representan en la Figura 49.

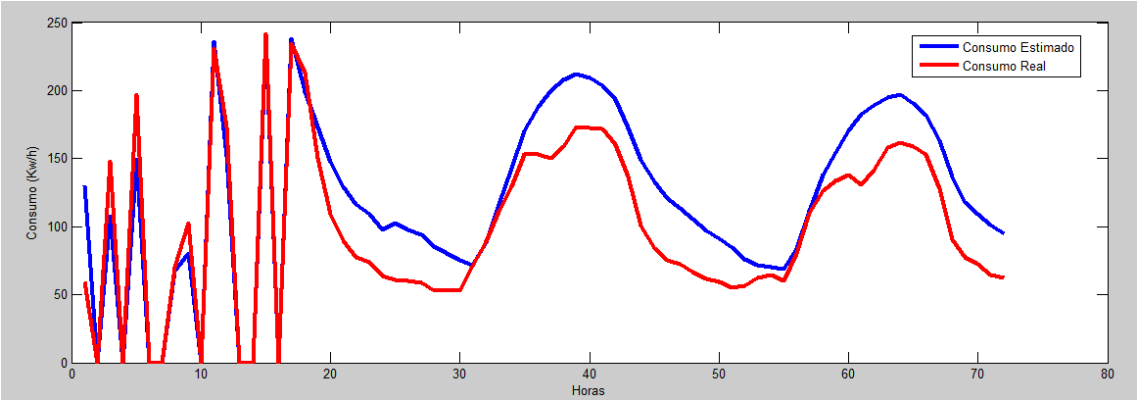

Figura 49. Consumo Estimado frente al real

En este último ejemplo se ha representado una semana de Junio. Como se observa los resultados no son tan buenos como para la otra semana. No obstante los

picos de consumo se encuentran correctamente identificados y las tendencias son buenas.

4.4.5. lsim como comando de simulación

Ya que el toolbox de EnergyPlus se basa en diagramas simulink para procesar los datos energéticos, dentro del archivo propio de control no podemos llamar a otro bloque de simulación, por lo que habrá que realizar una transcripción del diagrama Simulink a código en Matlab. Para llevar a cabo dicha tarea emplearemos el comando de simulación **'lsim'**.

El comando lsim nos permite realizar una simulación de un proceso lineal, incluido un sistema de espacio de estados. Para ello emplearemos la siguiente línea en la ventana de comandos:

```
Y=lsim(Sys,U,T,X0);
```

donde Sys se corresponde a la función de transferencia o espacio de estados a simular, U es la entrada al sistema, T el vector de tiempos con los pasos de simulación (longitud similar a U) y X0 el estado inicial del sistema.

Hace falta crear una función de estado pasándole los parámetros a,b,c y d anteriormente obtenidos, para lo que emplearemos el código del Anexo IV.2. Los resultados obtenidos son similares a los obtenidos por Simulink. En la figura 50 se representa la temperatura de salida obtenida, resultando ser similar a la de la figura 46.

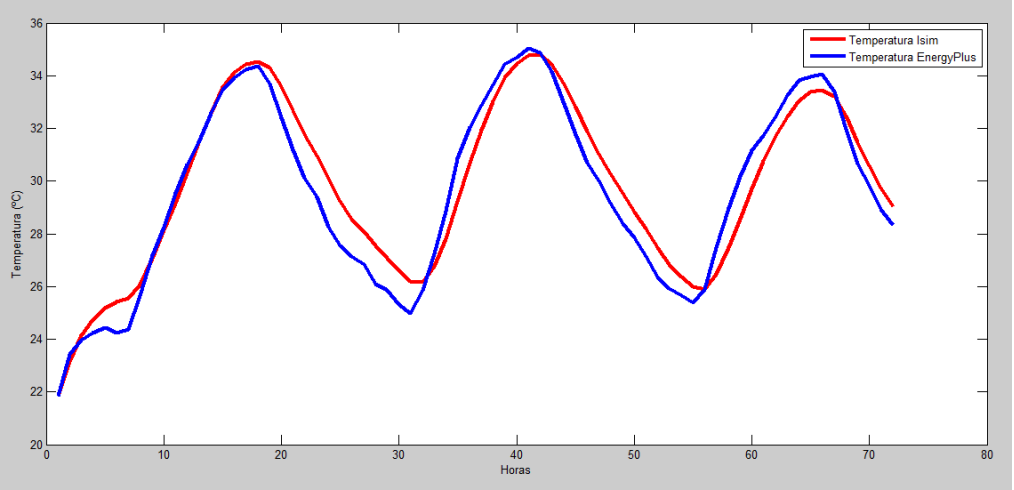

Figura 50. Temperatura con lsim vs Temperatura EnergyPlus

Para las pruebas venideras, se ha empleado según el caso la simulación por bloques Simulink o la simulación por comando lsim. No obstante, como ya se comentó, para la prueba final del MPC se ha empleado este último por temas de incompatibilidad con la herramienta toolbox empleada.

También se debe señalar la velocidad de ejecución de la simulación. Si bien es cierto que Simulink es un programa mucho más completo que admite múltiples bloques y funcionalidades que en Matlab nos ocuparían muchas líneas de código, la velocidad a la que ejecutaba la simulación era bastante superior que empleando el comando lsim para nuestro caso de simulación de modelos de primer orden lineales.

5. MPC (Model Predictive Control)

Con la identificación correctamente realizada y preparada, se procede a la implementación del control predictivo, para lo que realizaremos una serie de pruebas básicas para comprobar cómo funciona, derivando finalmente a una implementación completa del control sobre el modelo real o modelo EnergyPlus.

En primer lugar configuraremos el algoritmo genético y crearemos la función de coste a optimizar, dentro de la cual se debe simular nuestro modelo estimado para calcular la predicción de la salida. Las condiciones de optimización vendrán dadas por las necesidades del proyecto (en nuestro caso necesitamos minimizar el consumo para un rango determinado de temperaturas).

Una vez tengamos todo preparado para la realización de pruebas, cogeremos en primer lugar un caso simple de optimización seleccionando distintos setpoints sin apagar la máquina. De esta manera se estudiará el efecto del control y se comparará sobre la selección de un setpoint fijo en un valor determinado en el modelo real.

Como segunda prueba, dejaremos el setpoint fijo en un valor e introduciremos estados de ON/OFF de la máquina, comparándolo con el caso anterior y observando si mejora o no el sistema.

Ya para finalizar, implementaremos la selección de setpoints junto al ON/OFF en un único algoritmo, de manera que genere aleatoriamente estados de encendido y apagado de la máquina, y cuando elija encendido, seleccione un setpoint dentro de nuestro rango a controlar. Los resultados se compararán con un control de setpoint por banda, de manera que la temperatura se mantendría siempre entre un valor de rango designado, pero sin ningún tipo de control aplicable.

La estrategia de optimización difiere al uso clásico del MPC, ya que se optimizará una función de coste propia, creando un criterio propio y por tanto resultando un tanto diferente.

5.1. Criterio de optimización

El MPC tratará de resolver el problema planteado a continuación, donde la variable x es la salida del algoritmo y por tanto la variable que se empleará en nuestra función de coste.

La ecuación que relaciona la salida con x es la siguiente:

$$Consumo = u21 + u22 + u23 + u24$$

donde:

- $u21 = radiación * G21$
- $u22 = temperatura * G22$
- $u23 = x\ (Setpoint) * G23$
- $u24 = x_{on} * G24$

x_{on} es una variable modificada que se corresponde al comportamiento de la máquina cuando es apagada y vuelta a encender.

La función de coste a optimizar por tanto, sigue el siguiente criterio base:

$$J(X) = \sum_{k=1}^{10} Consumo(x)$$

Los límites a los que se tiene que restringir el algoritmo, se corresponden a la salida de temperatura, definida por:

$$Temperatura = u11 + u12 + u13 + u14*$$

donde:

- $u11 = radiación * G11$
- $u12 = temperatura * G12$
- $u13 = x\ (Setpoint) * G13$
- $u14 = x_{on} * G14$

*Nota: La salida de temperatura es modificada en la máquina de estados donde toma directamente la salida u13 cuando se fija un valor de SP.

De esta manera, quedarían definidos los límites de salida del GA como:

$$\begin{cases} si\ T \in (19{,}23) & penalti = 0 \\ si\ no\ T \in (19{,}23) & penalti = 10^9 \end{cases}$$

5.2. Configuración del basic GA

Como se había comentado anteriormente, el MPC se basa en predicciones futuras de las entradas para calcular un horizonte de predicción que nos muestra el comportamiento futuro de un sistema y actuar en consecuencia a ello. Para estudiar las predicciones, se empleará un algoritmo genético evolutivo para generar soluciones candidatas al sistema. El algoritmo seleccionado es el **'Basic GA'** del instituto AI2 de la Universidad Politécnica de Valencia.

La estructura principal del mismo se encuentra en el Anexo III. Se puede observar como realiza las funciones de evolución genética explicadas en el apartado de teoría. No se entrará en excesivos detalles en la forma de actuar del algoritmo y si nos centraremos más en la parte de configuración del mismo junto con la definición de la función de coste a optimizar.

Para ejecutar el algoritmo, emplearemos un script para configurar parámetros como la población inicial, el nº de generaciones, la función de coste a optimizar y variables que se quieran introducir en la misma. El resultado del algoritmo se almacena en una variable **'xmin'** siendo el mejor de los resultados obtenidos con los criterios seleccionados.

Para configurar el número de generaciones y la población inicial, dependerá del tiempo en el que se tarde en simular el modelo estimado junto a la precisión a la hora de obtener el resultado final, por lo que finalmente se debe hacer un cálculo aproximado según las necesidades. Por defecto y para la realización de las pruebas se ha dejado en 50 la población inicial y en 5 el nº de generaciones máximas para la obtención de los SP.

Complementariamente al algoritmo, tenemos dos funciones que nos van mostrando en pantalla el resultado obtenido en cada iteración (generación) junto con un resultado final completo una vez finalizada la ejecución del mismo.

Una modificación propia que se le ha realizado al algoritmo es el redondeo de los posibles candidatos a un valor más coherente con el tipo de variable que estamos

tratando. Por defecto los candidatos salen con un número elevado de decimales, pero los Setpoint de temperatura rara vez admiten más de un decimal. Se ha optado por tanto por redondear a la primera décima todos los valores obtenidos antes de evaluarlos en el modelo predictivo.

5.3. Función de coste

5.3.1. Estructura principal

La función de coste consiste en una función que nos devuelve un resultado el cual el algoritmo evolutivo pretende optimizar según una serie de criterios. En este caso se ha considerado que la mejor solución al sistema consistirá en el menor valor de salida de la función correspondiente al sumatorio de consumos futuros. Todo esto debe cumplir que la temperatura de salida del modelo no supere la banda de 19-23 grados. En nuestro caso como la simulación se realiza para verano, nos interesa que no supere los 23 grados ya que rara vez va a bajar de 19, y aunque fuera el caso no se tiene ningún control diseñado para el mismo.

Lo primero que haremos dentro de la función de coste es cargar los datos necesarios para realizar la simulación, para lo que emplearemos el script correspondiente a cada caso y que se basa en el descrito en el Anexo IV.1. Hay que recordar que en este caso los setpoint no provienen de un fichero en concreto con las predicciones, sino que será un valor designado por el GA. Por lo tanto, configuraremos las entradas de Setpoint normal y Setpoint de arranque para posteriormente emplearlas en la simulación.

Con las salidas preparadas, simulamos con Simulink/Isim el modelo estimado, obteniendo las siguientes salidas: Temperatura Operativa sin HVAC, Temperatura de Arranque, Consumo Total. Todas ellas serán procesadas por una máquina de estados que construirá las dos salidas finales de Temperatura y Consumo que se analizarán posteriormente.

5.3.2. *Máquina de estados*

Los criterios para la construcción de las salidas finales del sistema vendrán dados por una serie de características que debemos asumir de la siguiente manera según la salida.

Para la temperatura operativa:

- Si la consigna está fijada en un valor:
  - ❖ Si la consigna es mayor que la temperatura operativa, entonces la salida del sistema es la temperatura operativa.
  - ❖ Si la consigna es menor que la temperatura operativa, entonces la salida del sistema es directamente el valor de consigna.
- Si la consigna es 0:
  - La temperatura de salida es igual a la rampa de arranque de temperatura, siempre y cuando su valor sea inferior a la de la temperatura operativa, en cuyo caso adoptaría dicho estado

Para el consumo:

- Si la consigna está fijada en un valor:
  - Se toma el valor de consumo obtenido directamente de la simulación
- Si la consigna es 0:
  - Se toma el valor de salida de consumo como 0, ya que este no se tiene en cuenta en la salida final, aunque si se considera el valor de arranque.

5.3.3. *Criterio de optimización*

Como criterio de optimización, existen muchos tipos como minimizar el error cuadrático, el error medio, etc.… En nuestro caso como no queremos tratar un error sino el valor de consumo de la máquina, sumaremos los valores futuros de consumo en Kw/h y obtendremos los Kw/h de media para la candidata. Ese valor será procesado por el MPC, de manera que si resulta el más pequeño se considerará como solución a la entrada de Setpoints designada (función fitness).

Para que el sistema se mantenga dentro del rango de temperaturas deseado, hace falta una penalización de resultado cuando se supere dichos límites. Para ello basta con crear una variable **'penalti'** con un valor suficientemente alto para que cuando se le sume al resultado final, el GA lo descarte como solución óptima. Para añadir el penalti simplemente analizamos la temperatura de salida y si algún valor excede el rango, se le añade al sumatorio de consumos.

Hay que destacar que no se puede ajustar de manera precisa el rango de temperaturas ya que el modelo es impreciso y muchas veces tiene un error ligeramente superior a medio grado, por lo que es conveniente añadir un margen de tolerancia si se permitiese de más o menos medio grado, ya que a veces se descartan soluciones optimas en el sistema real que luego en el simulado no lo son.

5.3.4. Diagrama de flujo

Como idea principal, se sugiere el diagrama de flujo representado en la Figura 51,

Para el estudio del algoritmo se ha empleado un diagrama más sencillo tal y como se muestra en la figura 52, sin tener en cuenta el modelo real con el objeto de agilizar las pruebas. No se ha empleado el factor de corrección, basando el resultado únicamente en las tendencias del sistema.

Para este caso se ha prescindido del uso del modelo real por complicaciones con Simulink, de manera que el estado inicial de ajuste se realiza a partir del modelo simulado teniendo en cuenta las predicciones pasadas para saber desde donde viene el sistema. Este esquema se ha empleado para las propuestas de control 1 y 2, mientras que para la propuesta 3 se recoge dentro del propio apartado su flujograma con la lógica correspondiente.

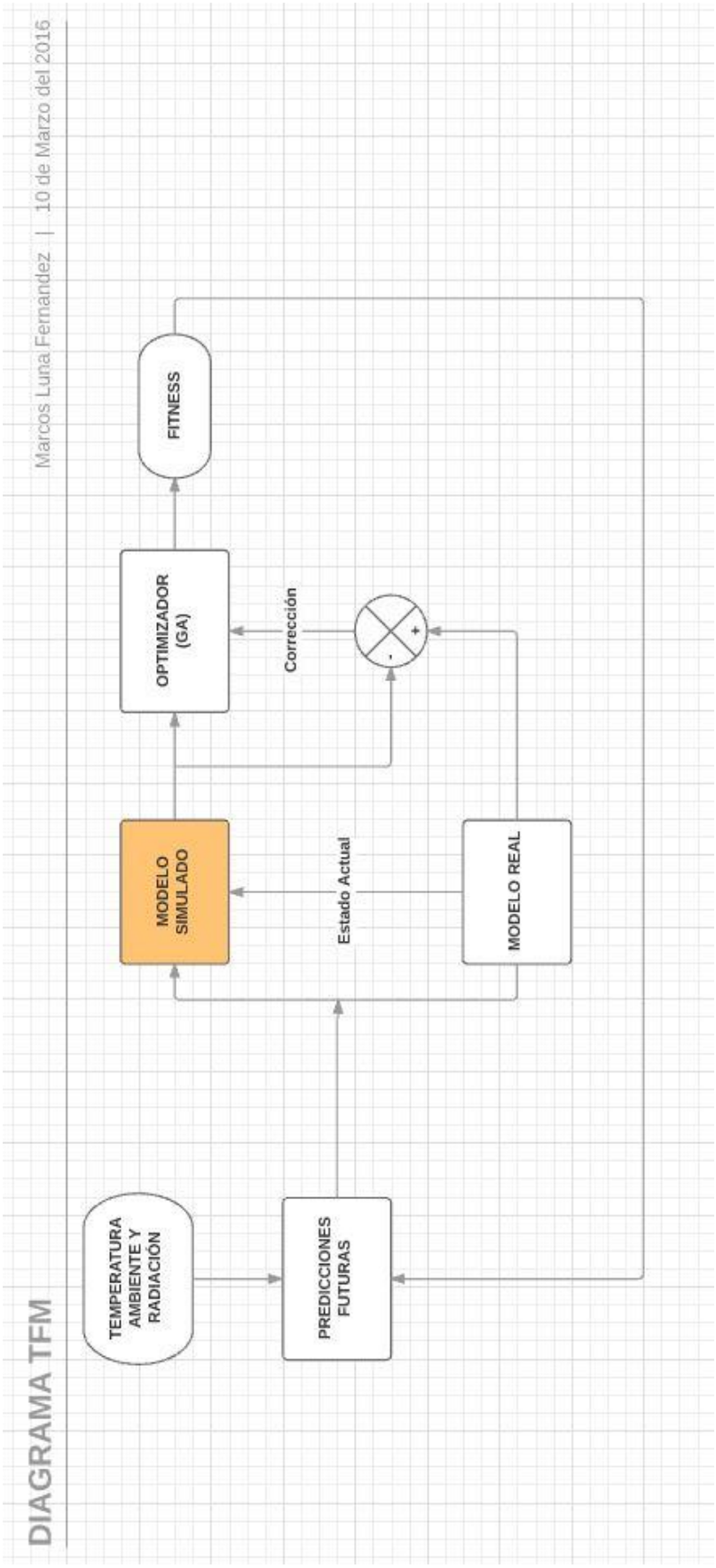

Figura 51. Diagrama principal MPC

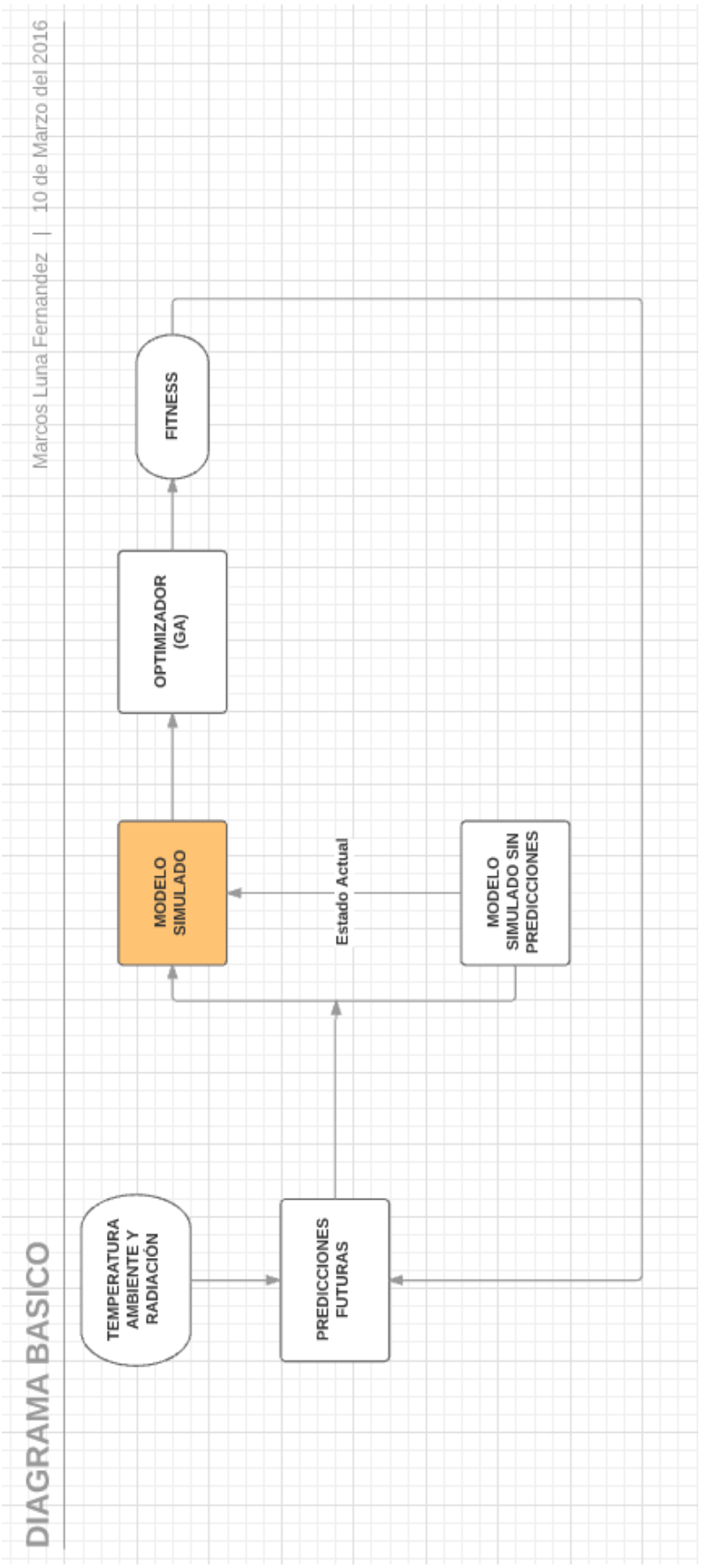

Figura 52. Diagrama básico

6. Resultados

En este apartado se han recogido los diferentes resultados obtenidos a lo largo de la elaboración del proyecto, el cual se ha sometido a diferentes propuestas para determinar su comportamiento frente a un control predictivo. Por lo tanto, hemos sometido al sistema a una serie de pruebas con diferentes condiciones, reflejando los resultados y sacando una conclusión.

6.1. Propuesta de control 1. Primera parte

En esta propuesta, realizaremos una toma de contacto con el MPC. Para ello, emplearemos el algoritmo de manera que nos calcule una predicción de setpoints a lo largo de un día. En este caso dejaremos fijados los primeros setpoints en un valor fijo ya que de todas maneras no se aplicaría un control en esas horas debido a que las pruebas se han realizado con una temperatura que no supera los 23 grados durante ese periodo.

6.1.1. Configuración del algoritmo

Los parámetros que emplearemos para lanzar el algoritmo son los siguientes:

- Nº individuos: El número total de individuos de población inicial la dejaremos en un valor alto entre 50 y 100.
- Nº generaciones: Si no establecemos ningún criterio de convergencia, el algoritmo finalizará su cálculo en la 5ª generación.
- Función de Coste: Se encuentra en el Anexo VI.1 y VI.2
- Criterio de optimización: minimizar el sumatorio de consumos para las 20 horas siguientes.

6.1.2. *Ejecución del algoritmo*

Antes de la prueba, realizaremos una simulación previa en EnergyPlus de un día cualquiera de Agosto, y extraemos las variables hacia Matlab, para tener las predicciones deseadas y el punto de funcionamiento del sistema.

Una vez seleccionado y preparado todo, lanzamos el archivo '**EjecutarGA.m'** de manera que el algoritmo comience a trabajar. En este caso, seleccionará una lista de 20 setpoints que resulten los más óptimos a la hora de minimizar el consumo. Si hemos escogido un mes caluroso como Agosto, en la que cualquiera de sus días las temperaturas superan los 30 grados, la tendencia natural del algoritmo de predicción es fijar los setpoints en 23, ya que de entre 19 y 23 grados que se pueden seleccionar, esta última sería la opción que menos consumiría.

Esta aplicación no recoge los criterios iterativos del GA, de manera que solo obtenemos una única predicción en base a lo que ocurriría en 24 horas, lo que no resulta muy preciso ya que las condiciones de entrada podrían cambiar en tan poco tiempo y además no tenemos en cuenta el factor de corrección.

La prueba se ha ejecutado durante el día 7 de Julio correspondiente al archivo EPW de Valencia. Las variables de entrada (predicciones) se muestran en la figura 53.

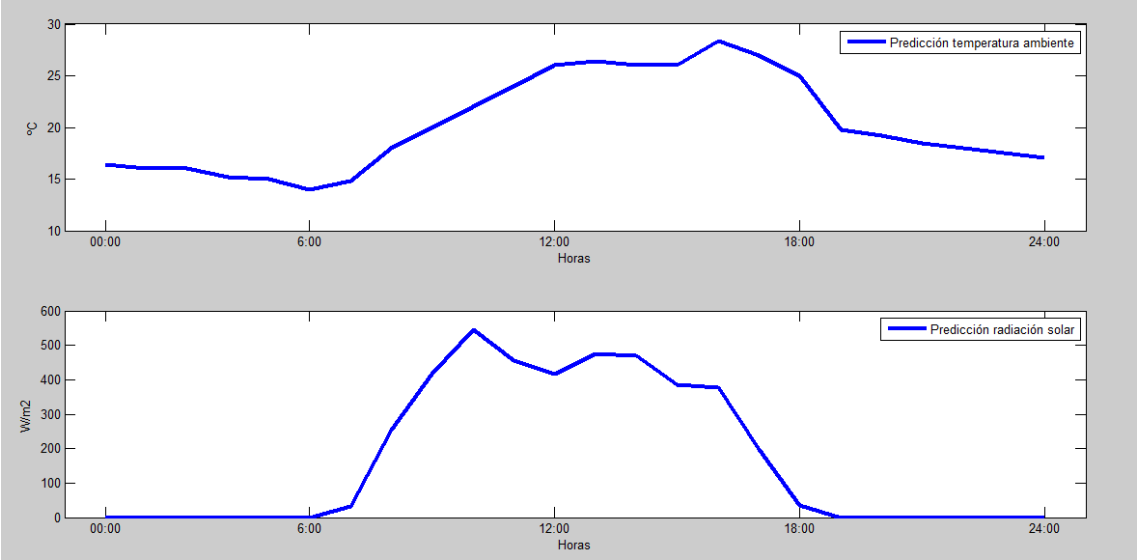

Figura 53. Variables de entrada

Estas predicciones simularían las entradas que serían tomadas en base a predicciones en tiempo real que pueden ser consultadas de manera online, por lo que no se asemejan a un día real sino que 'simula serlo'.

6.1.3. Resultados

Los resultados de temperatura operativa se omitirán ya que los setpoints van a oscilar entre 19 y 23 grados y nunca se van a salir de rango.

El algoritmo ha tardado aproximadamente 50 minutos y el resultado final ha sido:

```
######    RESULT    #########

   Objective function for xmin: 1507.9526

   xmin: [23 23 22 19 19 21 19 22 23 19 21 23 23 23 19 23 21 23
22 23]
```

El resultado del sumatorio de consumos para EnergyPlus manteniendo el SP fijo en un valor de 21 fue de 1648.1 KW.

La media de consumos entre las 5:00 AM y las 24:00 AM ha sido de 82.4 Kw/h para el consumo sin optimizar y de 75.39 Kw/h para el consumo optimizado.

La gráfica resultado de los consumos se representa en la Figura 54.

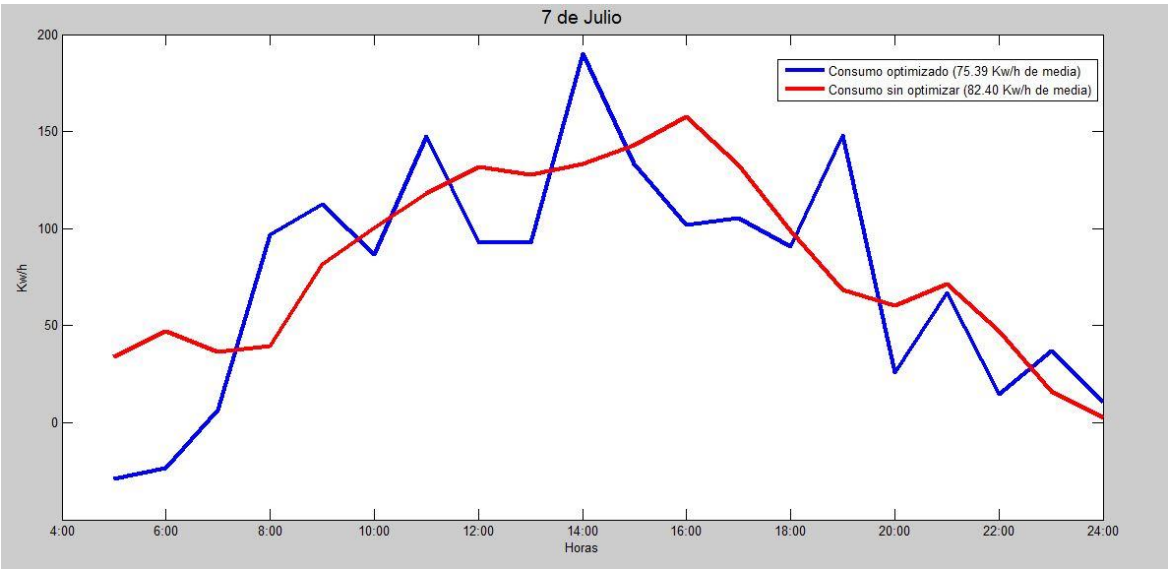

Figura 54. Comparativa de Consumo

6.1.4. Conclusiones

Los resultados obtenidos reflejan una clara tendencia positiva a la hora de variar los setpoints según el consumo de salida.

Teniendo en cuenta que nuestro sistema solo puede adoptar valores entre 19 y 23 grados, es lógico pensar que si la máquina está enfriando y busca el menor consumo posible, la tendencia sea a elegir consignas de valor alto. Observamos que efectivamente el valor que más se repite es el '23'.

El algoritmo no tiene en cuenta diversos factores como los nuevos horizontes de predicción ya que calcula de golpe la lista de setpoints que se debe asignar a la máquina, todo ello suponiendo que no varíen las condiciones de las predicciones (cosa muy poco probable).

Se observa una mejora de más o menos 7 Kw/h de media en el consumo a lo largo del día que fijando una consigna fija.

6.2. Propuesta de control 1. Segunda Parte

Continuando con la estrategia desarrollada en la primera parte, vamos a introducir lo que sería el uso clásico del MPC y que mejores resultados da, ya que asigna un setpoint para cada hora de simulación y calcula el nuevo horizonte de predicción en base a los resultados anteriormente obtenidos. Al no tener en cuenta el modelo real el control no sería del todo completo.

6.2.1. Configuración del algoritmo

Los parámetros que emplearemos para lanzar el algoritmo son los siguientes:

- Nº individuos: El número total de individuos de población inicial se ha reducido entre 25-50.
- Nº generaciones: En este caso se ha seleccionado un número máximo de generaciones de 3.
- Función de Coste: Se encuentra en el Anexo VI.1 y VI.2 (Modificaciones comentadas)

- Criterio de optimización: minimizar el sumatorio de consumos para las 5 horas siguientes.

6.2.2. Ejecución del algoritmo

Con los mismos datos de entrada usados en la primera parte, en este caso introduciremos una función adicional que simule pasos de 1 hora, para los cuales en cada uno de ellos ejecutará el mismo algoritmo pero con la salvedad de que solo estudiaremos el sumatorio de consumos de las siguientes 5 horas de simulación.

Para tal labor, el MPC calculará únicamente los 3 setpoints correspondientes al momento actual y a las siguientes 2 horas de simulación, seleccionando el resto de valores iguales al tercero.

Además, se debe tener en cuenta los setpoints anteriores para estudiar el comportamiento del consumo y de la temperatura, aunque en este caso no debería existir problema ya que esto únicamente afecta a estados de encendido y apagado anteriores.

La prueba se ha realizado el mismo día que en el anterior caso, exceptuando que en esta vez se ha optimizado el consumo durante una jornada laboral estándar debido a que las muestras de predicciones se han tomado de 24 horas, por lo que únicamente se puede optimizar hasta la hora 20 ya que tomamos en cuenta las 5 horas siguientes y reducir el margen podría suponer un comportamiento no completo. Esto se solucionaría fácilmente ampliando las predicciones de tiempo aunque luego el control se realice sobre un margen más pequeño.

Se ha añadido una línea adicional para garantizar que el consumo sea 0 si el setpoint elegido resulta mayor que la temperatura operativa de la zona.

6.2.3. Resultados

Los resultados de temperatura operativa se omitirán ya que los setpoints van a oscilar entre 19 y 23 grados y nunca se van a salir de rango.

El algoritmo finalmente se ha limitado a una población de 10 individuos iniciales y 3 generaciones máximo, reduciendo el tiempo entre iteración de horas a aproximadamente 5 minutos,

El resultado es el siguiente:

```
######    RESULT    #########

   Objective function for xmin: 1000.7

   xmin: [20 20 20 20 20 20 23 19 20 22 23 23 21 22 20 23 23 22
23 21 20 20 20 20]
```

El resultado del sumatorio de consumos para EnergyPlus manteniendo el SP fijo en un valor de 21 fue de 1290.9 KW.

La media de consumos entre las 5:00 AM y las 20:00 AM ha sido de 80.68 Kw/h para el consumo sin optimizar y de 62.54 Kw/h para el consumo optimizado.

La gráfica resultado de los consumos se representa en la Figura 55.

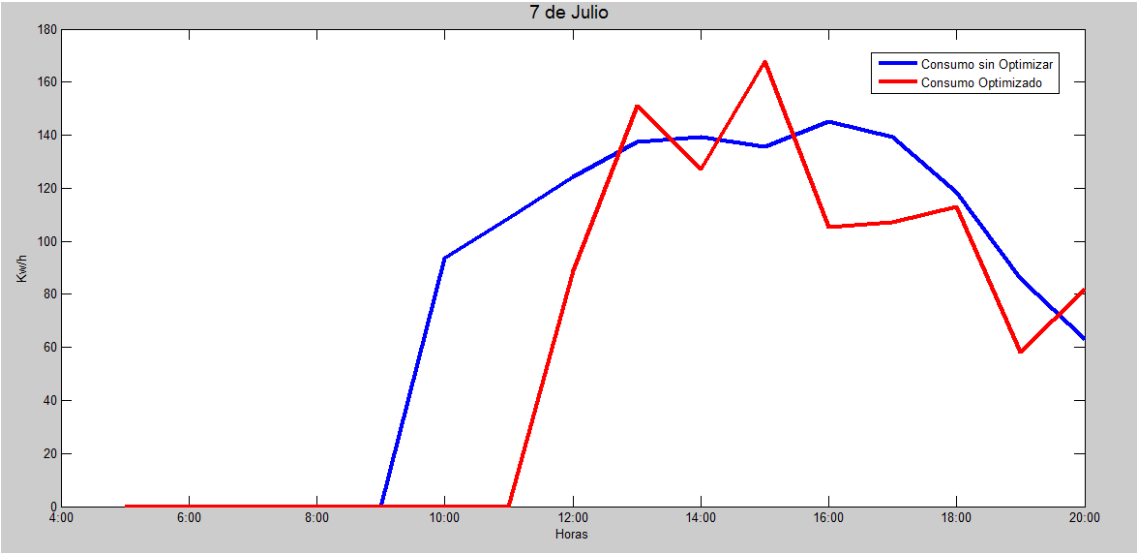

Figura 55. Comparativa de consumos

La temperatura operativa de salida se representa en la figura 56. Se observa como al principio no toma los valores de setpoint.

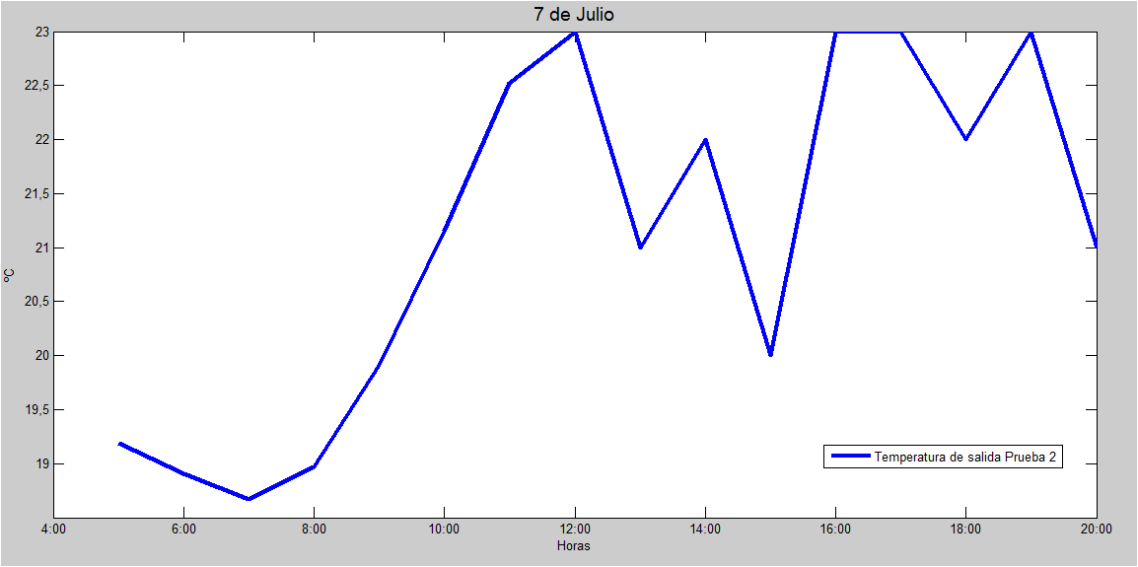

Figura 56. Temperatura de salida

6.2.4. Conclusiones

En la primera parte se puso los dos setpoints (frío y calor) en un valor fijo de manera que la máquina siempre estaba funcionando. En este caso solo se ha tenido en cuenta el setpoint de frío, de manera que la temperatura operativa inferior al setpoint designado hace que la máquina se mantenga apagada.

Hasta ahora hemos empleado el modelo para realizar comparativas y ver si el algoritmo funcionaba correctamente. Se observa que la optimización es mayor debido a los procedimientos adoptados.

Para las siguientes pruebas, por tanto, emplearemos el script de iteración y usaremos el modelo real para comparar resultados. Para la correcta ejecución del mismo, se precisa de identificar el arranque tanto de consumo como de temperatura, ya que como se observa en las pruebas realizadas no se ha tenido en cuenta. Si introdujéramos los valores en EnergyPlus las discrepancias con el modelo estimado serían mayores, y más si tenemos en cuenta el encendido y apagado de setpoints.

6.3. Propuesta de control 2

En esta segunda propuesta de control, visto los resultados anteriormente vistos, introduciremos la identificación del arranque de la máquina para ver cómo afecta al consumo y a la temperatura operativa.

Para poder comprobarlo, añadiremos en este caso una secuencia de apagado y encendido de la máquina sin variar el setpoint, y lo compararemos con el resultado de dejar la máquina fija en el mismo setpoint sin apagar ni encender de EnergyPlus.

Extenderemos las pruebas a 1 semana y emplearemos un periodo en el que las temperaturas estén siempre por encima de 23 grados, de manera que la máquina siempre se encuentre u operativa o apagada, pero nunca porque la temperatura operativa sea inferior al setpoint dado.

Para esta prueba hemos escogido 2 de los 3 setpoints resultados del MPC en vez de 1 solo, ya que de esta manera nos aseguramos que elija una secuencia de apagado considerando todos los factores posibles. En el futuro se dará prioridad a que el primer setpoint contenga el apagado y encendido de la máquina.

6.3.1. Configuración del algoritmo

Los parámetros que emplearemos para lanzar el algoritmo son los siguientes:

- Nº individuos: El número total de individuos de población inicial se ha reducido entre 25-50.
- Nº generaciones: En este caso se ha seleccionado un número máximo de generaciones de 3-5.
- Función de Coste: Se encuentra en el Anexo VII.1 y VII.2
- Criterio de optimización: minimizar el sumatorio de consumos para las 10 horas siguientes.

6.3.2. Ejecución del algoritmo

Para la siguiente propuesta, evaluaremos el efecto que tiene el encendido y apagado de la máquina sobre el consumo. En este caso ha sido necesario introducir un comportamiento distinto a las anteriores pruebas debido al efecto del arranque de la máquina sobre las variables de salida.

Lo primero de todo es que cuando la máquina recibe el setpoint de apagado (recordemos que en EnergyPlus el apagado para frío es seleccionar un setpoint muy elevado como por ejemplo 40-45) la temperatura 'recupera' su estado habitual. En este caso se presentarían dos situaciones: Que la temperatura a la que debería estar el edificio fuese mayor a la actual por el setpoint y al revés.

En cuanto al consumo, ocurre lo mismo que para la temperatura. Cuando la máquina parte de un estado de reposo, el consumo se dispara en un 'pico de arranque' para luego recuperar su tendencia habitual. Cuando se apaga, el consumo instantáneo en ese momento es de 0.

Para tratar estas 'perturbaciones' (las denominaremos así ya que no poseen un comportamiento lineal sobre el sistema) introduciremos una máquina de estados en las cuales evaluando las condiciones de la temperatura, consumo y el setpoint actual, se tomará un valor u otro en función del mismo.

En cuanto a las variables que intervienen, se deben construir de la siguiente manera:

- En primer lugar tendremos un vector que contenga los setpoints sin los apagados de la máquina. Si el setpoint tiene un 0 originalmente, este adoptará el valor anterior.
- Debido a que en la identificación se observó que el pico de arranque era muy similar dentro del rango de 19-23 grados, se creará otro vector con los setpoints fijados en un valor medio (21 grados) y se le introducirán los 0 o apagados de la máquina (en Simulink usaremos el estado 0 para el apagado en vez de 40-45, luego se procesa para EnergyPlus).
- Se creará un tercer vector de setpoints que contengan los setpoints originales junto con los apagados de la máquina para crear el offset que determine el arranque de temperatura (no es lo mismo que empiece en 19 grados que en 23 grados).

Por lo tanto, dentro del esquema Simulink, tendremos varias líneas de simulación de las cuales obtendremos las variables que intervienen en la máquina de estados. Para la temperatura Operativa emplearemos las predicciones de Temperatura Ambiente y Radiación Solar. Para la Temperatura de Arranque, emplearemos la variable 'SP' que contiene los apagados del HVAC.

Para el consumo, únicamente se tendrá una sola línea donde se sumen la influencia de la Temperatura Ambiente, la Radiación Solar, el Setpoint siempre encendido, y el setpoint con los apagados. Todo ello, junto al offset correspondiente nos dará como resultado el consumo final de la máquina lo más asemejado posible a la realidad.

Adicionalmente, se han añadido unos limitadores para cuando haya problemas a la hora de simular los 10 pasos siguientes, como por ejemplo si quisiésemos simular 72 horas y nos encontrásemos en la hora 66, no podemos simular hasta las 76 horas si solo tenemos las predicciones en el rango anterior. Esto se puede solucionar aumentando las predicciones o usando el limitador por código.

6.3.3. Resultados

El algoritmo finalmente se ha limitado a una población de 50 individuos iniciales y 3 generaciones máximo, reduciendo el tiempo entre iteración de horas a aproximadamente 20 minutos,

El resultado es el siguiente (Solo se contabiliza de 8:00 a 20:00 durante los 7 días):

```
######   RESULT   #########

   Objective function for xmin: 3162,003

   xmin: [0 1 0 1 0 1 1 1 0 1 0 1 0] [0 0 1 1 0 1 0 1 0 1 0 1 0]
[0 1 0 1 0 1 0 1 1 0 1 0 1] [0 0 1 0 1 0 1 0 1 0 1 0 1] [1 1 1 0
1 0 1 1 0 1 1 0 1] [1 0 0 1 1 1 0 1 1 1 0 1 0] [0 0 1 0 1 1 1 0 1
0 1 1 0]
```

Cada uno de los vectores se corresponde con 1 día de una semana de 7 días. Dentro de ese día, el rango de hora va desde las 8:00 AM hasta las 20:00 PM. Los 0

representan el apagado de la máquina, y los 1 el setpoint fijado en un valor constante (21).

El resultado del sumatorio de consumos para EnergyPlus manteniendo el SP fijo en un valor de 21 e introduciendo los apagados de la máquina fue de 3383,34 KW.

La media de consumo por día para el modelo real sin optimizar y optimizado se reflejan en la Figura 57.

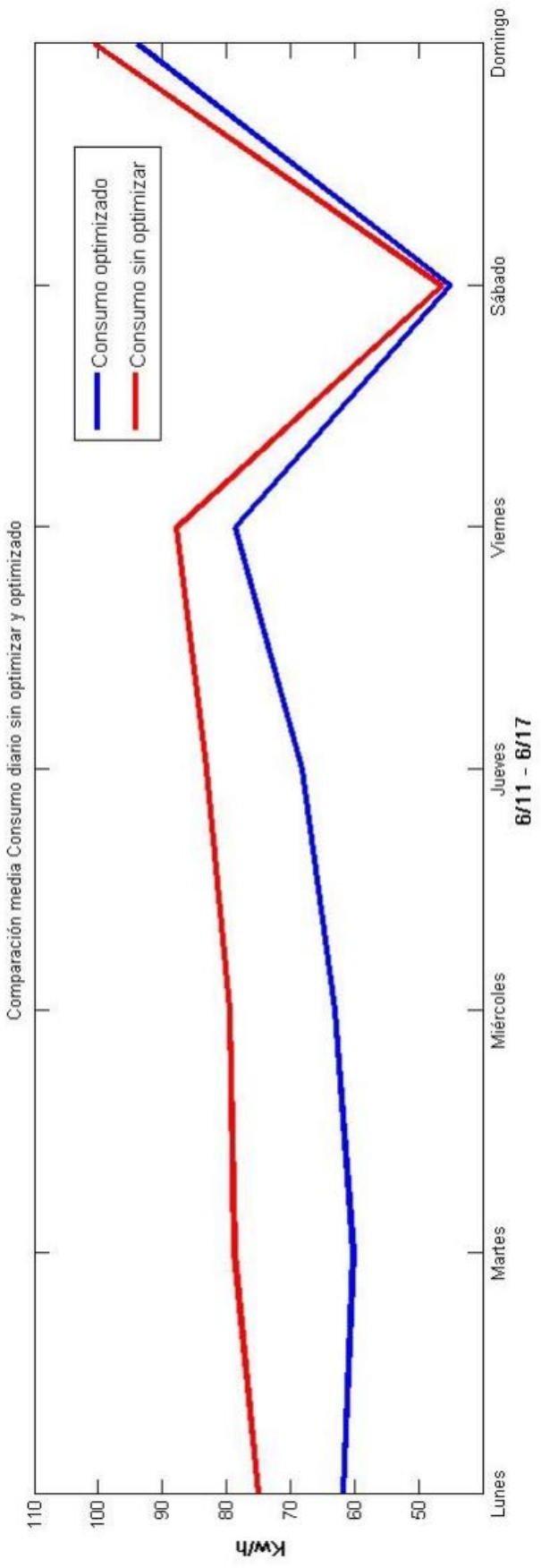

Figura 57. Comparación de consumos a lo largo de una semana

6.3.4. Conclusiones

Como se puede observar, existe una mejora considerable en el consumo medio por día introduciendo secuencias de apagado y encendido de la máquina manteniendo en un rango de temperaturas de 19 y 23 grados el sistema. La mejora es más notable en unos días respecto a otros (se observa que el sábado apenas se produce variación).

En cuanto a las temperaturas, en la Figura 58 se encuentran la temperatura operativa con la máquina apagada y la temperatura operativa con la optimización aplicada de la máquina HVAC. Se ha añadido un 'grid' en 23 grados para ver que la temperatura en algunos casos supera dicha franja. Esto se debe a que en la función de coste, tal y como viene descrita en el Anexo VII.2, se ha añadido un pequeño margen de tolerancia de más o menos 1 grado, ya que la discrepancia entre el modelo real y el estimado puede tener errores.

Esto se puede solucionar empleando un factor de corrección comparando la temperatura de salida para cada hora en el sistema real después de aplicar el setpoint predicho, cosa que se ha introducido en la siguiente propuesta de control en la que se pretenderá introducir el MPC al completo.

Pese a que solo se haya realizado un control ON/OFF, vemos que ya resulta más óptimo que dejar en un valor constante la máquina. Los picos de arranque se compensan con las horas en las que la máquina está apagada, de manera que a la larga resulta más conveniente.

No obstante, el control por banda sí que resulta más óptimo, ya que en función de la temperatura la máquina tomará la decisión de apagarse o encenderse para mantenerse en el rango de temperaturas deseado. Es en ese momento donde el predictivo podría (o no) resultar más conveniente debido a que se puede adelantar al comportamiento térmico del modelo.

Para esclarecer finalmente las dudas presentadas, implementaremos lo ya descrito en el siguiente apartado.

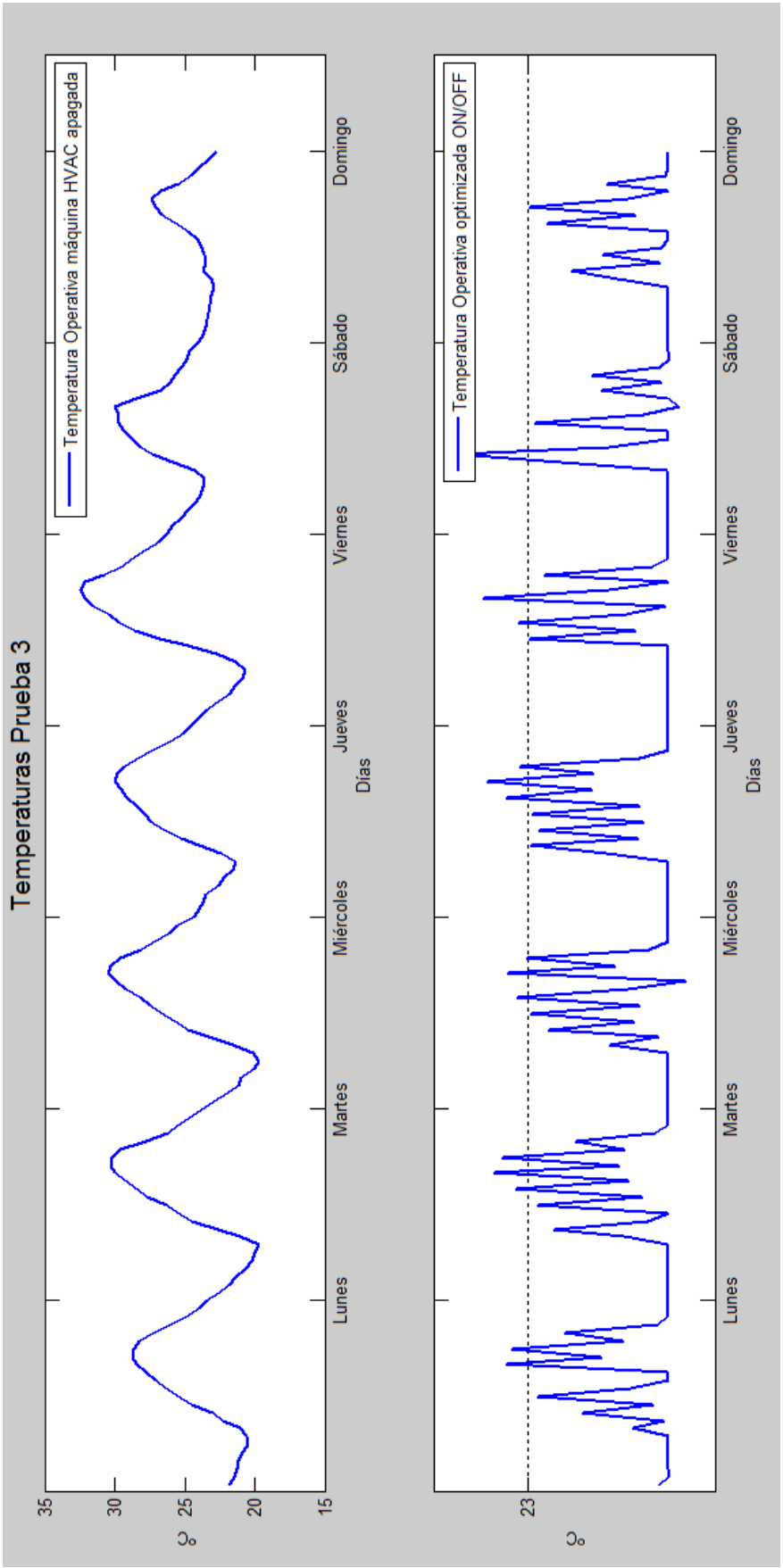

Figura 58. Temperaturas de salida

6.4. Propuesta de control 3. Primera Parte

El nacimiento de esta primera parte se origina de la imposibilidad de simular un conjunto de bloques Simulink dentro de otro conjunto. La base sobre la que trabaja MLE+ impide utilizar el comando 'sim' dentro de archivo de control que nos proporciona el toolbox.

Esto se debe a que MLE+ está desarrollado sobre Simulink, y por tanto no podemos simular un modelo Simulink dentro de otro, ya que Matlab no será capaz de procesarlo y nos lanzará un mensaje de error.

Ya que hasta ahora no se había planteado el uso del comando lsim, se desarrolló esta prueba con el objeto de probar el funcionamiento del GA utilizando como modelo lineal de entrada (modelo real) el propio modelo estimado anteriormente.

Debido a tal consecuencia, aplicar un factor de corrección, tal y como se establece en el MPC, es irrelevante ya que la salida de ambos modelos va a ser la misma, por lo que se arrastrará el error a lo largo de la simulación de forma invariante.

A pesar de ello, la prueba nos permite implementar los conocimientos obtenidos de las anteriores y ver el comportamiento y funcionamiento del control predictivo, así como la importancia de emplear el modelo real junto con un factor de corrección que ajuste los parámetros de salida de nuestro modelo.

6.4.1. Configuración del algoritmo

Los parámetros que emplearemos para lanzar el algoritmo son los siguientes:

- Nº individuos: El número total de individuos de población inicial se ha reducido entre 25-50.
- Nº generaciones: En este caso se ha seleccionado un número máximo de generaciones de 3-5.
- Función de Coste: Se encuentra en el Anexo VIII.1 y VIII.2
- Criterio de optimización: minimizar el sumatorio de consumos para las 10 horas siguientes.

6.4.2. Ejecución del algoritmo

Para ejecutar el algoritmo, emplearemos un script iterativo tal y como hicimos en la Prueba 3, adjunto en el Anexo V. El script simulará cada hora, en la cual ejecutará una función de inicio del GA para que optimice la función de coste en el instante t en el que se encuentre el bucle.

Como resultado del GA, guardaremos los 2 SP resultantes, es decir, el SP para t y para t+1. Esto se realiza debido a los problemas presentados cuando el valor estimado de SP para el instante t+1 resulta un apagado de la máquina.

En cuanto a la función de coste a optimizar, sufre una serie de variaciones con respecto a la Prueba 3. La primera de ellas resulta de los límites introducidos en el GA. Se establece que para el rango entre 17-19 y 23-25, el SP de salida sea 0, mientras que entre 19 y 23 el SP tenga el valor acordado. De esta manera hay un 50 % de posibilidades tanto de encendido como de apagado de la máquina.

La lógica resultante es algo más compleja que lo visto hasta ahora, de manera que el flujograma descrito en la Figura 59 intenta sintetizar toda la información y los criterios de construcción de las salidas y optimización de las variables.

Para simular las salidas, emplearemos los resultados obtenidos del modelo como referencia de valor inicial o punto de funcionamiento (principalmente para saber dónde nos encontramos).

El criterio de optimización es el mismo, minimizar el consumo para las siguientes 10 horas y manteniendo el rango de temperatura

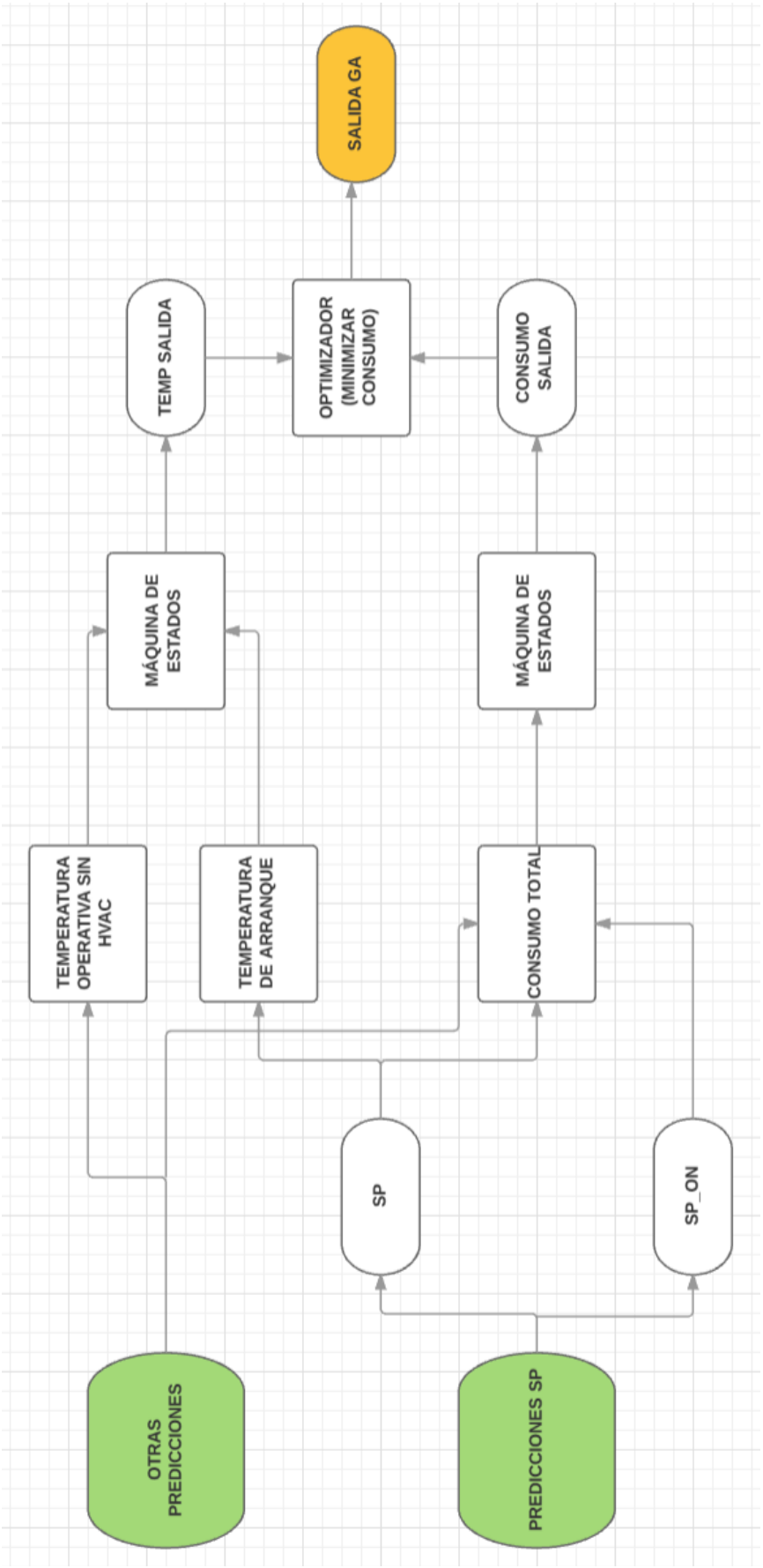

Figura 59. Flujograma Prueba 4

6.4.3. Resultados

El algoritmo finalmente se ha limitado a una población de 50 individuos iniciales y 5 generaciones máximo.

Se ha realizado pruebas para dos intervalos de 3 días de los meses de Junio y Julio.

Del 24 al 26 de Junio

El resultado es el siguiente (Solo se contabiliza desde las 6:00 AM del primer día hasta las 23:00 PM del último día):

```
######    RESULT    #########

   Objective function for xmin: 5829.1

   xmin: [20 20 0 0 22.9 0 22.8 0 20.6 23 19 0 22.2 22.8 19.3 0
22.2 21.7 19.2 0 22.8 22.1 19.1 0 21.1 22.7 0 0 0 0 22.9 22.3 19.5
0 19.4 ...]
```

Para una mayor comprensión se ilustra en la Figura 60 los Setpoints resultantes del algoritmo genético.

El resultado del sumatorio de consumos para EnergyPlus colocando el termostato dual entre 19 y 23 grados es de 5801.7 KW con una media de 80,58 Kw/h.

La media de consumo por día para el modelo real sin optimizar y optimizado se reflejan en la Figura 61.

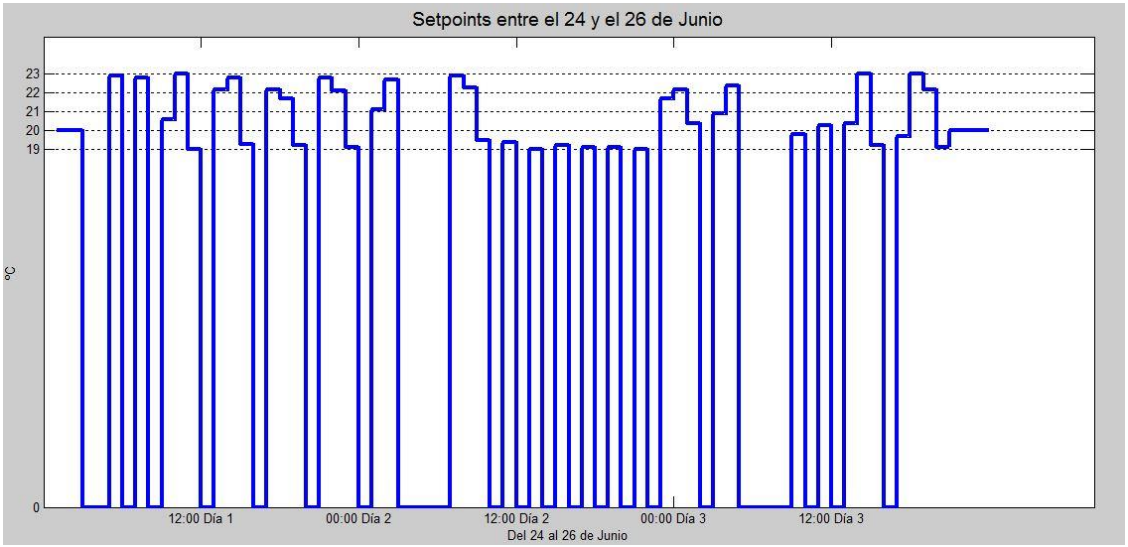

Figura 60. Salida de Setpoints del 24 al 26 de Junio

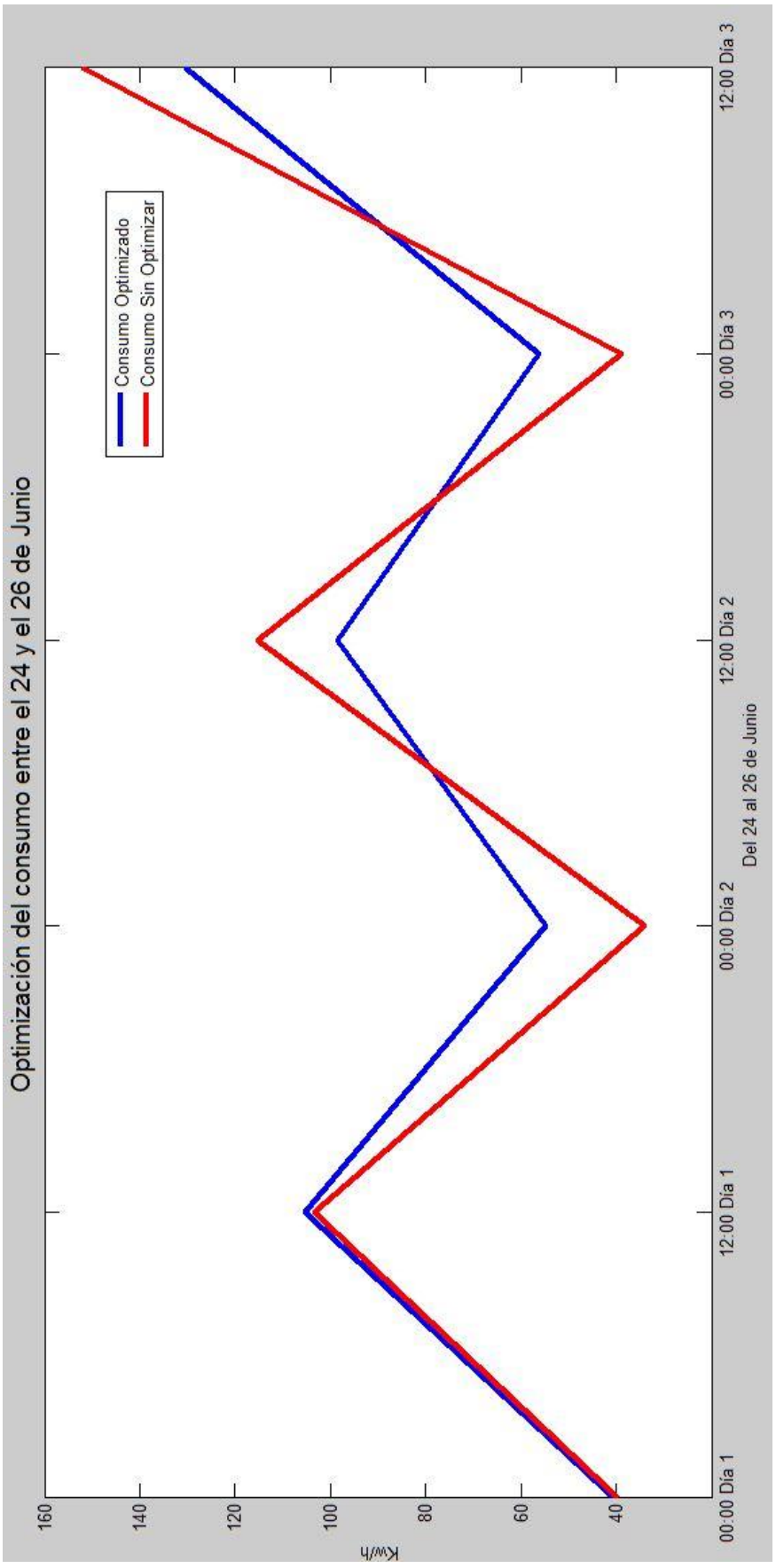

Figura 61. Comparativa de Consumos del 24 al 26 de Junio

Por último, la temperatura resultante del modelo para este intervalo se ilustra en la figura 62.

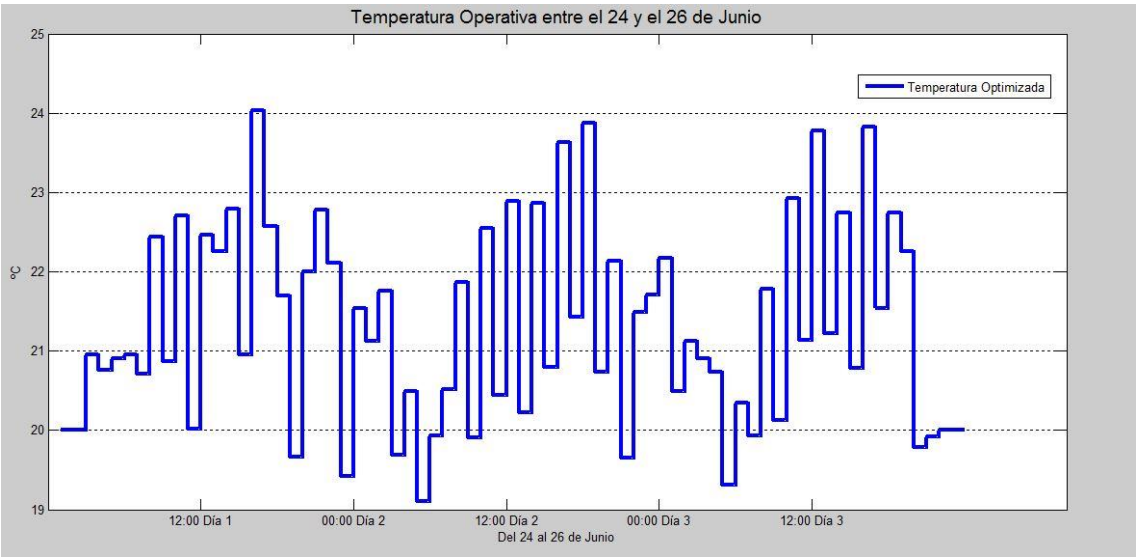

Figura 62. Temperatura Operativa del 24 al 26 de Junio

Del 13 al 15 de Julio

El resultado es el siguiente (Solo se contabiliza desde las 6:00 AM del primer día hasta las 23:00 PM del ultimo día):

```
######    RESULT    #########

   Objective function for xmin: 6968.8

   xmin: [20 20 0 21.9 22.1 22.7 19.7 0 21.6 20.7 19.2 0 21.8 21.4
19 0 22.3 22.7 19 0 22.9 21.8 19.5 0 22 22.5 19.2 0 23 22.2 19.6
0 19.2 0 22.1 21.7 20.1 0 19 0 19 0 21.1 21.3 19.6 0 19.5 0 19.1
0 22.7 21.6 20.2 0 19.8 0 19 0 19.3 0 19.1 0 19.5 0 20.2 0 19.5 0
20 20 20 20]
```

Para una mayor comprensión se ilustra en la Figura 63 los Setpoints resultantes del algoritmo genético.

El resultado del sumatorio de consumos para EnergyPlus colocando el termostato dual entre 19 y 23 grados es de 7329.7 KW con una media de 101.8 Kw/h

La temperatura de salida se representa en la Figura 64.

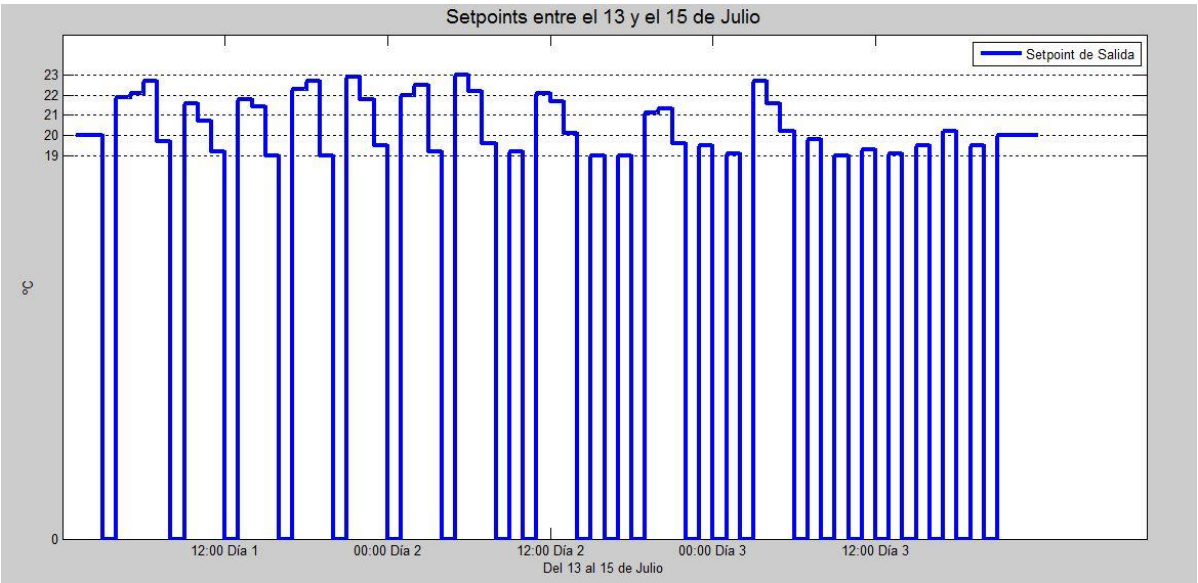

Figura 63. Salida de Setpoints del 13 al 15 de Julio.

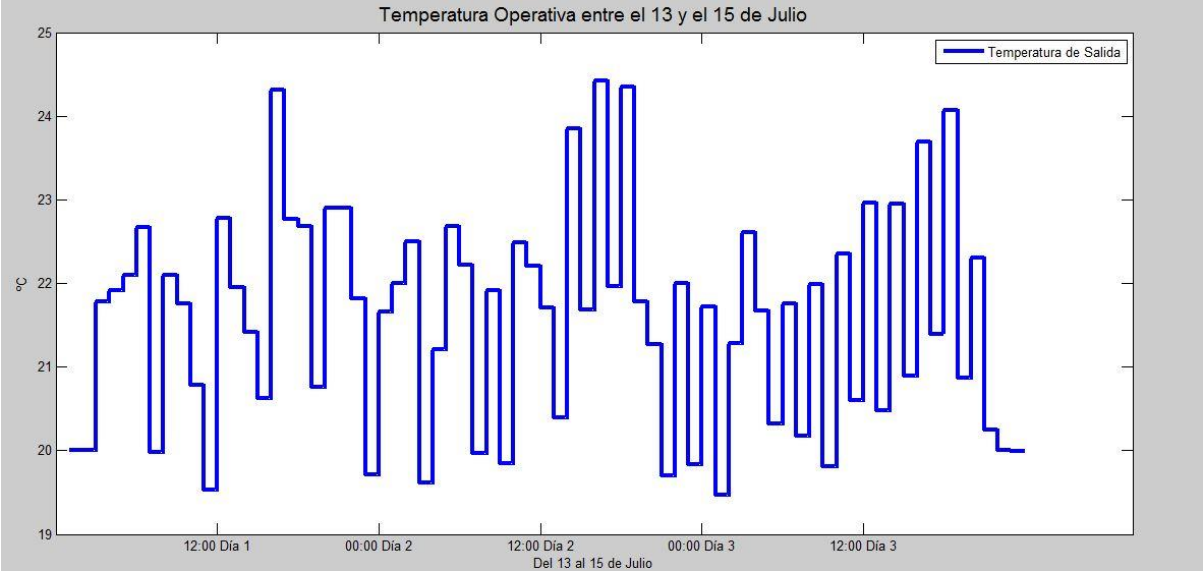

Figura 64. Temperatura Operativa del 13 al 15 de Julio

La media de consumo por día para el modelo real sin optimizar y optimizado se reflejan en la Figura 65.

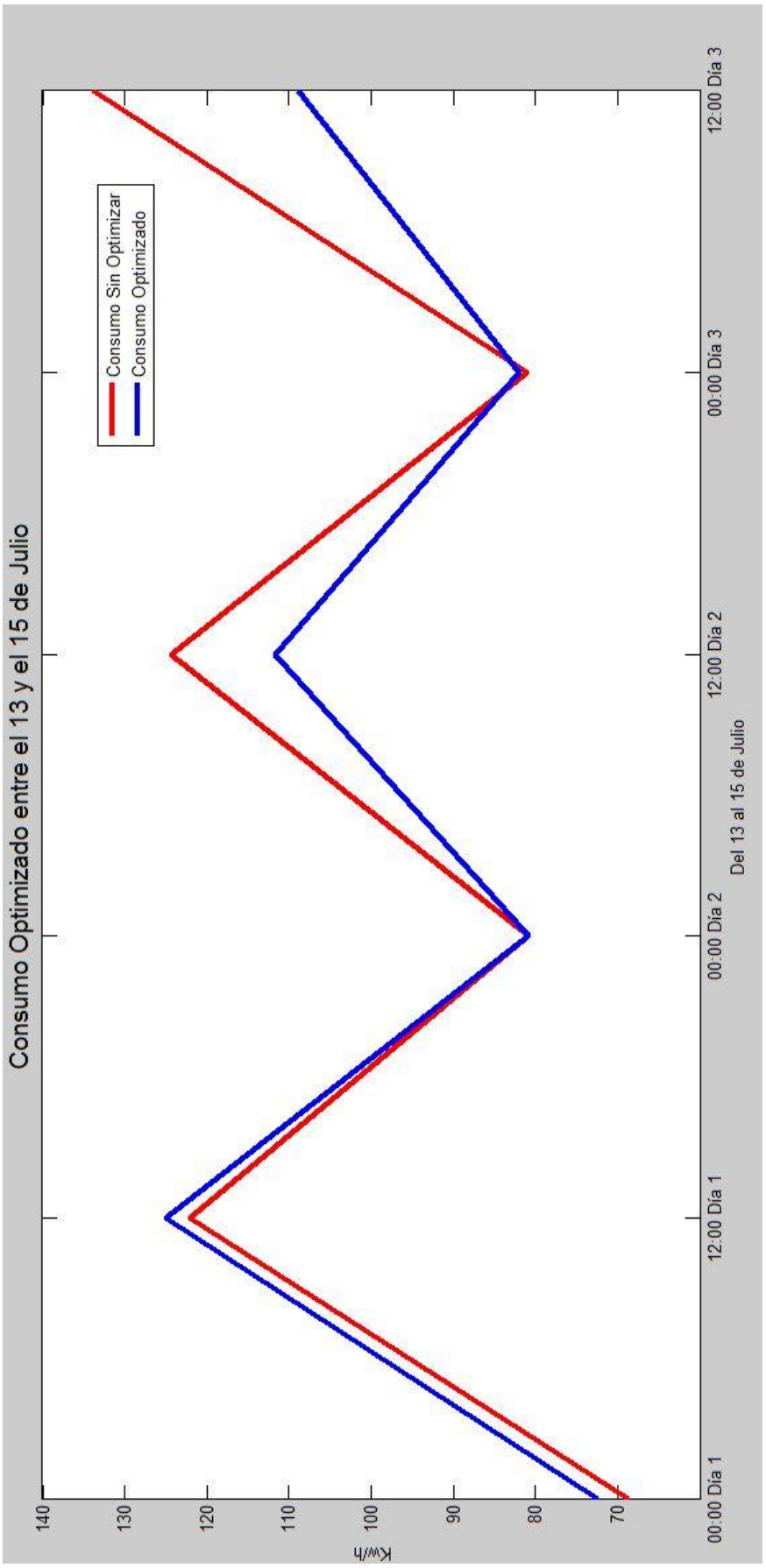

Figura 65. Comparativa de Consumos del 13 al 15 de Julio

6.4.4. Conclusiones

Podemos observar una clara diferencia entre los dos periodos elegidos. Mientras que para los días de Julio el GA es capaz de optimizar en un 5,18 %, en los días elegidos de Junio empeora en torno al 0,5 % la salida de consumo.

Esto se debe principalmente a las temperaturas que hay en dichos intervalos. Mientras que para el mes de Julio las temperaturas son muy elevadas, en Junio a veces caen hasta por debajo de los 19 grados, de manera que se activa el control por banda de la máquina HVAC, resultando un tanto complicado la optimización.

Esto quiere decir que durante un número de horas en los cuales la temperatura de salida está entre 19 y 23 grados, en el modelo real la máquina no funcionaría, por lo que hay que tener en cuenta esto a la hora de calcular las salidas.

Se evidencia la imprecisión propia de emplear el modelo como referencia real a la hora de optimizar. No obstante, el resultado no es del todo malo sobre todo en el periodo del 13 al 15 de Julio en los que incluso se llegó a optimizar la salida.

No obstante, los criterios de optimización quedan un tanto al ‘aire’ ya que las salidas empleadas contienen un error con respecto a la original. De ahí que la temperatura operativa en ocasiones exceda los 23 grados designados (en el código anexado se puede observar un pequeño margen añadido debido a dicho error).

Además, la optimización no sigue un patrón o un criterio definido, por lo que la prueba no resulta para nada concluyente. Es necesario implementar el algoritmo en el modelo real y realizar el control predictivo clásico sobre el mismo.

Se debe por tanto, añadir un factor de corrección así como introducir una recompensa por la cual se premie que el primer setpoint sea 0 con respecto al segundo y tercero (solucionando de esta manera el problema de coger el SP para t y t+1).

Para solucionar el problema de simulación, se empleará otro método para lograr tal fin (el comando lsim) que junto con las necesidades y exigencias planteadas, desencadenan en la segunda parte de esta propuesta, considerada ya como prueba concluyente para determinar el rendimiento definitivo del MPC sobre la optimización de consumo de la máquina HVAC

6.5. Propuesta de control 3. Segunda Parte

Hasta ahora, en las diferentes pruebas realizadas, se ha visto el funcionamiento del algoritmo y se ha aprendido su metodología de trabajo para optimizar el consumo. Para implementar el control predictivo clásico y definitivo, hace falta emplear el modelo real para calcular el estado en el que nos encontramos y aplicar en tiempo real las variables de salida del MPC.

En la presente prueba, se ha implementado dicho control, ya que lo visto hasta ahora no resultaba nada concluyente y se necesitaba implementar correctamente la lógica del MPC, donde un factor de corrección, obtenido de la diferencia entre una simulación previa y la salida del modelo estimado para t-1, nos ajusta la variable de temperatura de salida de manera que se incremente la precisión del MPC.

De igual forma, se podría haber introducido un factor de corrección para el consumo, pero para los resultados no se ha precisado su uso ya que en definitiva, añadir o quitar un valor a una señal de salida no va a afectar a la hora de minimizarla (consumo medio).

Se ha adaptado la simulación y máquina de estados para usarse en simbiosis con el comando 'lsim', además de que su uso ha incrementado de sobremanera la velocidad de ejecución del algoritmo, pudiendo de esta manera realizar pruebas más rápidas e incrementar los valores de generación y población inicial del GA sin que afecte al rendimiento.

6.5.1. Configuración del algoritmo

Los parámetros que emplearemos para lanzar el algoritmo son los siguientes:

- Nº individuos: El número total de individuos de población inicial se ha reducido entre 75-100.
- Nº generaciones: En este caso se ha seleccionado un número máximo de generaciones de 7-10.
- Función de Coste: Se encuentra en el Anexo IX.1, IX.2 y IX.3
- Criterio de optimización: minimizar el sumatorio de consumos para las 10 horas siguientes.

6.5.2. Ejecución del algoritmo

En este caso, emplearemos el fichero de control dado por EnergyPlus para implementar el MPC. Usaremos variables 'persistents' para almacenar los valores actuales de salida del modelo y usarlos en la siguiente iteración para calcular la temperatura de salida del sistema. También, almacenaremos los factores de corrección y estimaremos una media por día para la prueba más larga (de nuevo iremos incrementando el periodo de muestra).

Inicialmente se hicieron pruebas introduciendo como población inicial el resultado dado por el GA en la anterior iteración. No obstante, al tener un sistema con 2 estados, al introducir dicho factor al comienzo forzará al GA a ir por otro camino (estimando que es mejor dejar la máquina encendida), por lo que para las pruebas finales se ha eliminado.

Emplearemos como funciones de simulación las descritas en el apartado 4.4.5. *Lsim como comando de simulación*. Para saber en qué momento estamos, al comienzo de la iteración y dentro de la función de declaración de variables descrita en el Anexo IX.4, se realizará una simulación previa cogiendo los valores de salida obtenidos para el día anterior del inicio del sistema junto con el periodo a muestrear. Esto quiere decir que, si por ejemplo nos encontramos a las 3:00 PM del Día 1 (steptime de 15), simularemos el día anterior y el actual hasta las 2:00 PM y sacamos el valor estimado del modelo, el cual emplearemos junto con el valor anterior para calcular la corrección.

Para determinar la lógica y funcionamiento final del GA, se representa en un flujograma detallado en la Figura 66 cada uno de los elementos que intervienen en el proceso de estimación y optimización.

Como podemos observar, el cambio más significativo es la introducción del parámetro de corrección que ajusta nuestra salida, así como la modificación de la máquina de estados para simplificar su uso y determinar de manera más precisa las salidas del modelo.

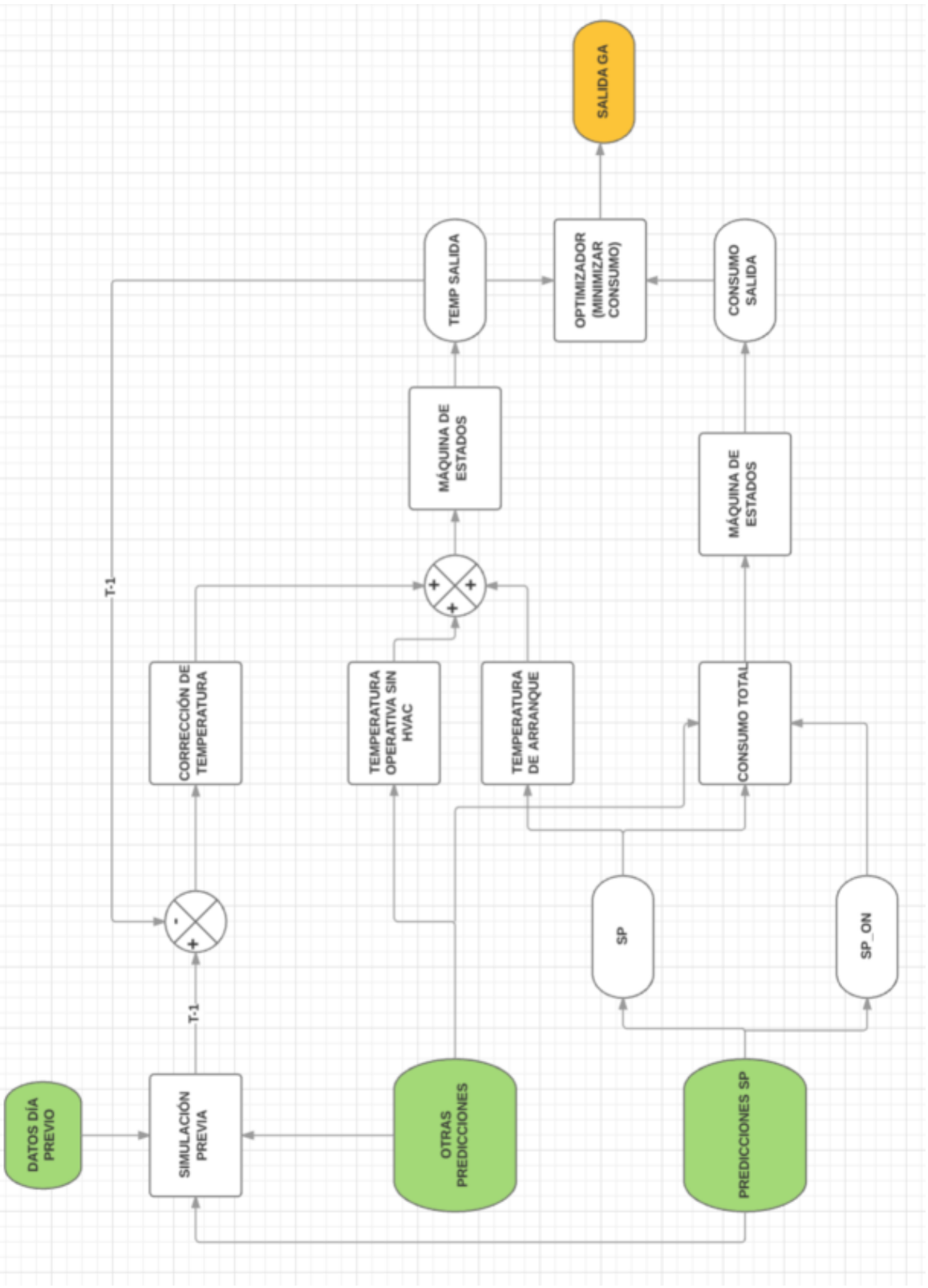

Figura 66. Flujograma principal de la Prueba 5

6.5.3. Resultados

El algoritmo finalmente se ha limitado a una población de 100 individuos iniciales y 10 generaciones máximo.

Se ha realizado pruebas para un mismo intervalo de tiempo, sacando datos inicialmente para 1 día, luego para 3 días y finalmente para la semana entera (en cada resultado se introducen ajustes para el siguiente intervalo)

Simulación para 1 día (13 de Julio)

El resultado es el siguiente (Solo se contabiliza desde las 6:00 AM hasta las 23:00 PM del mismo día

```
######   RESULT   #########

   Objective function for xmin: 2384.8

   xmin: [23 23 23 23 23 23 21.2 22.6 22.9 22.6 19 0 19 0 21.7 23
19 0 22.3 23 22.8 22.5 21.4 22]
```

Durante los primeros SP se ha fijado un valor de 23.

El resultado del sumatorio de consumos para EnergyPlus colocando el termostato dual entre 19 y 23 grados es de 2483.9 KW con una media de 103.4958 Kw/h.

La media de consumo por día para el modelo real sin optimizar y optimizado se reflejan en la Figura 67.

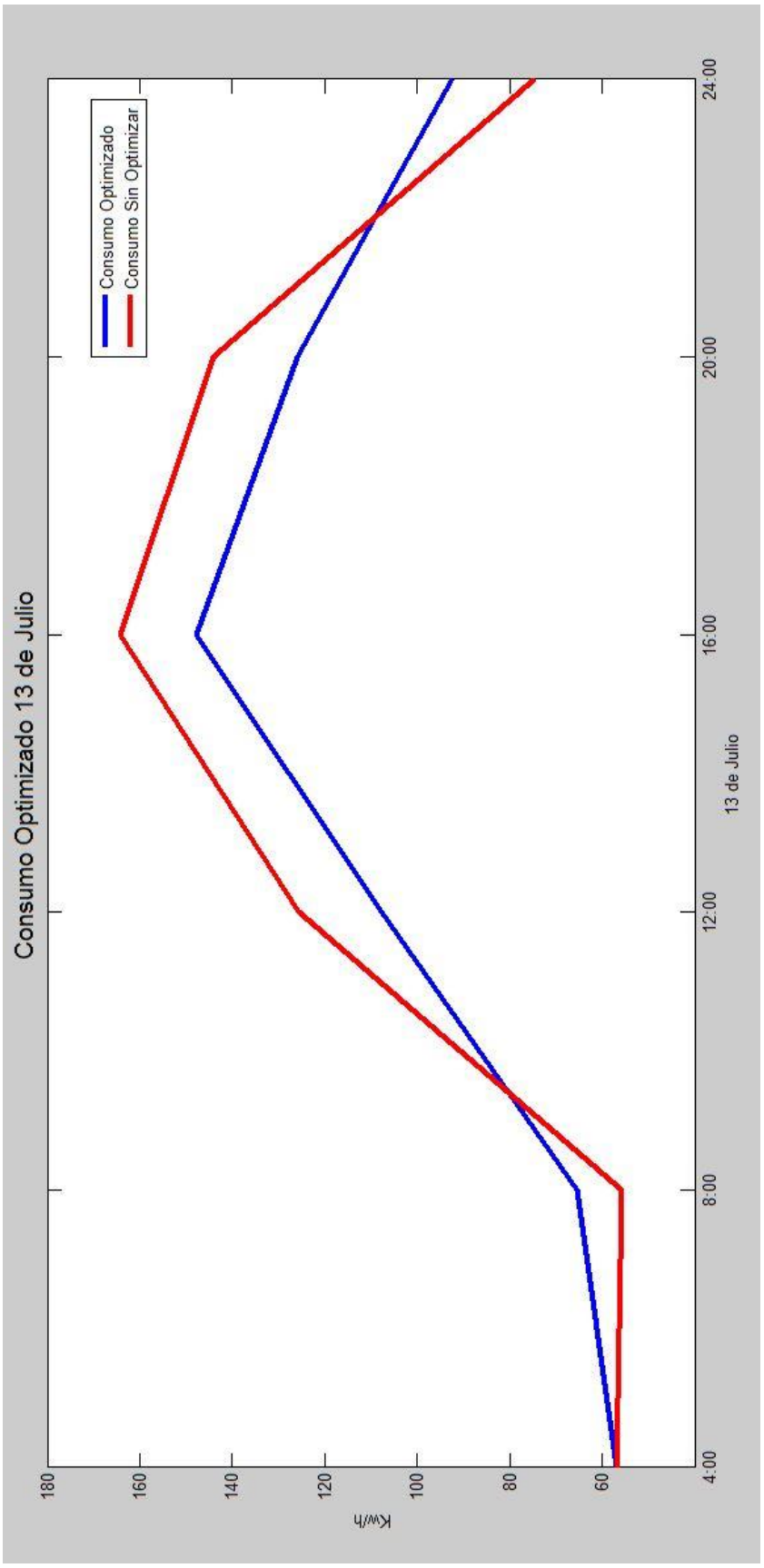

Figura 67. Consumo Optimizado para 1 día (13 de Julio)

La temperatura de salida se muestra en la Figura 68.

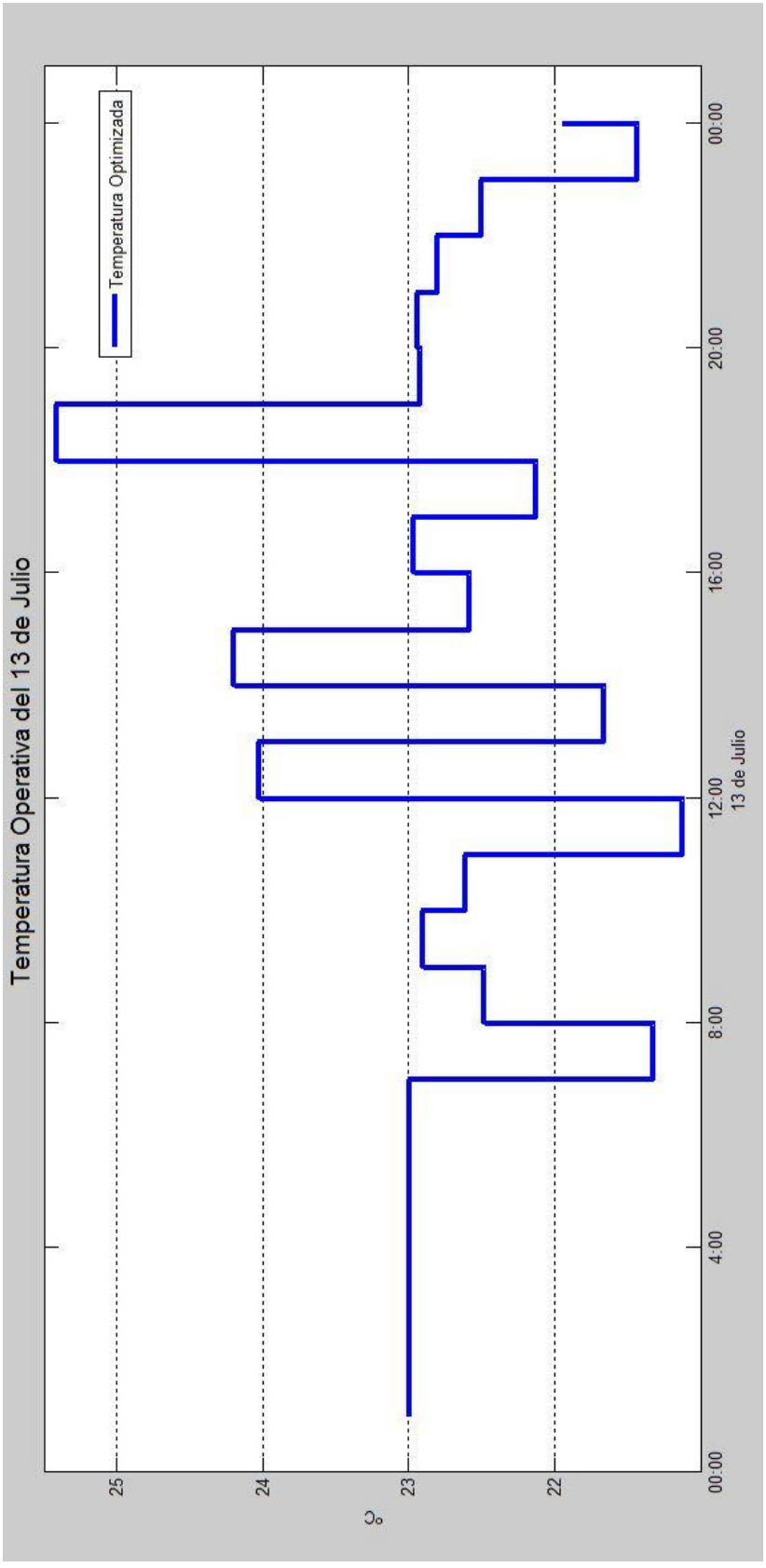

Figura 68. Temperatura Operativa de salida para 1 día (13 de Julio)

Simulación para 3 días (13 al 15 de Julio)

El resultado es el siguiente (Solo se contabiliza desde las 6:00 AM del primer día hasta las 23:00 PM del último día

```
######   RESULT   #########

   Objective function for xmin: 7291.9

   xmin: [23 23 23 23 23 23 22.5 21.6 23 23 19.4 0 22.6 19.1 23
21.1 22.4 22.3 19 0 22.4 21.4 22.4 22.7 22.5 22.1 22.4 22.2 23 23
22.3 22.3 22.1 19 0 22.3 23 19.6 0 21.9 22.6 19 0 22.1 21.7 20.6
22.9 22.6 22.1 23 22.2 22.6 23 21.4 22.7 23 19.4 0 21.2 19 0 22.5
22.7 22.7 19.1 22.7 22.9 22.4 22.8 22.8 23 21.7]
```

Durante los primeros SP se ha fijado un valor de 23.

El resultado del sumatorio de consumos para EnergyPlus colocando el termostato dual entre 19 y 23 grados es de 7439.2 KW con una media de 103.3222 Kw/h.

La media de consumo por día para el modelo real sin optimizar y optimizado se reflejan en la Figura 69.

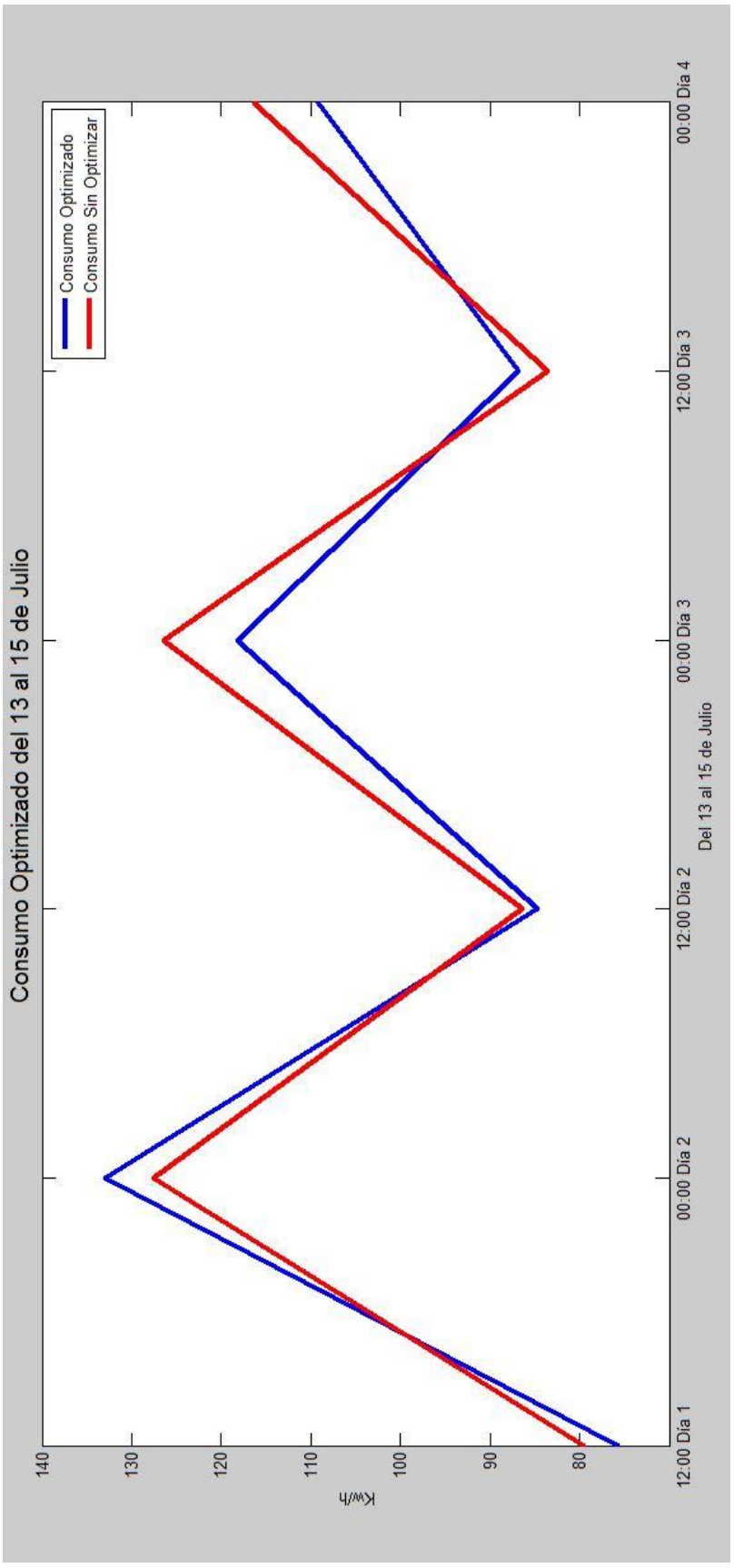

Figura 69. Consumo optimizado para el periodo de 13 a 15 de Julio

La temperatura de salida se muestra en la Figura 70.

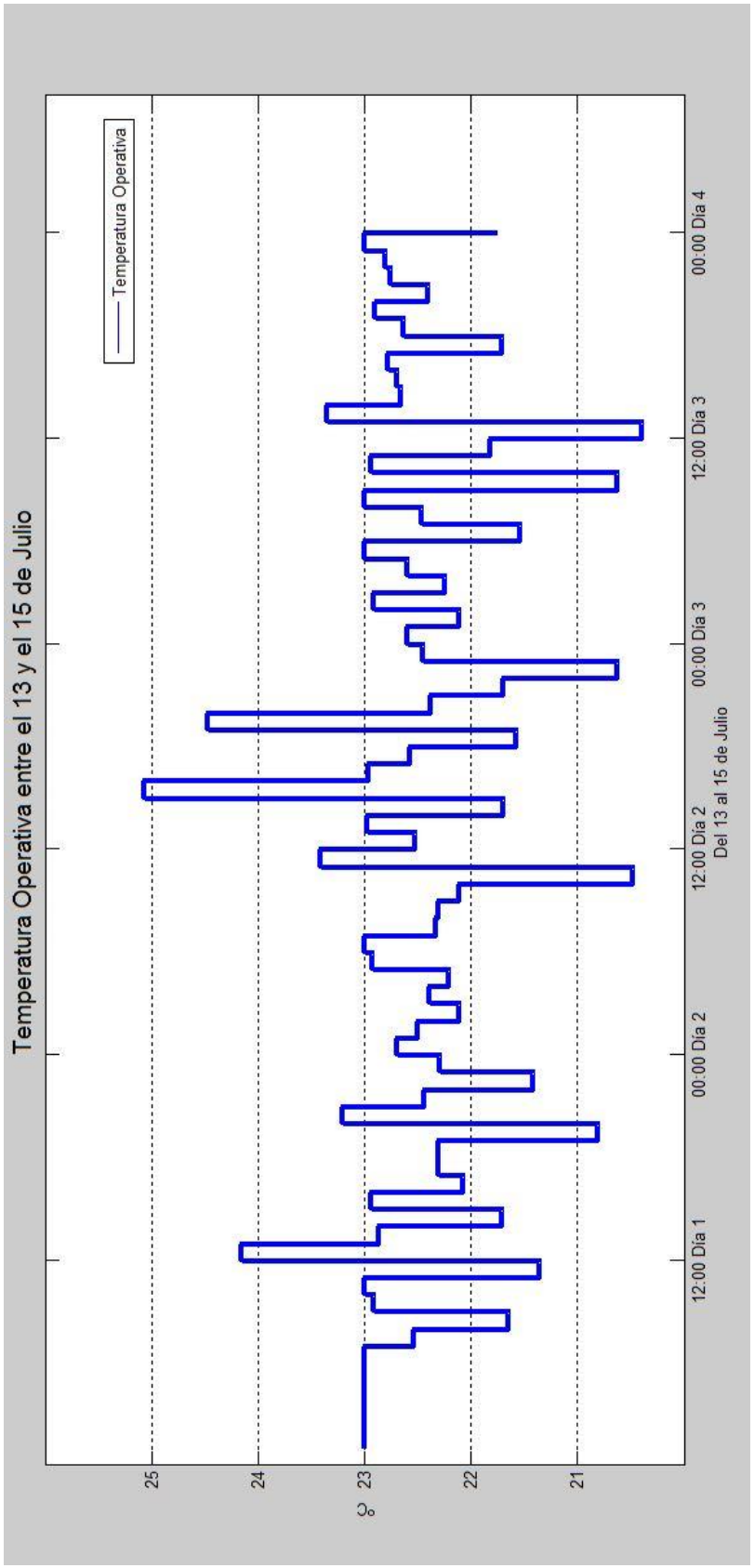

Figura 70. Temperatura Operativa de salida para el periodo de 13 a 15 de Julio

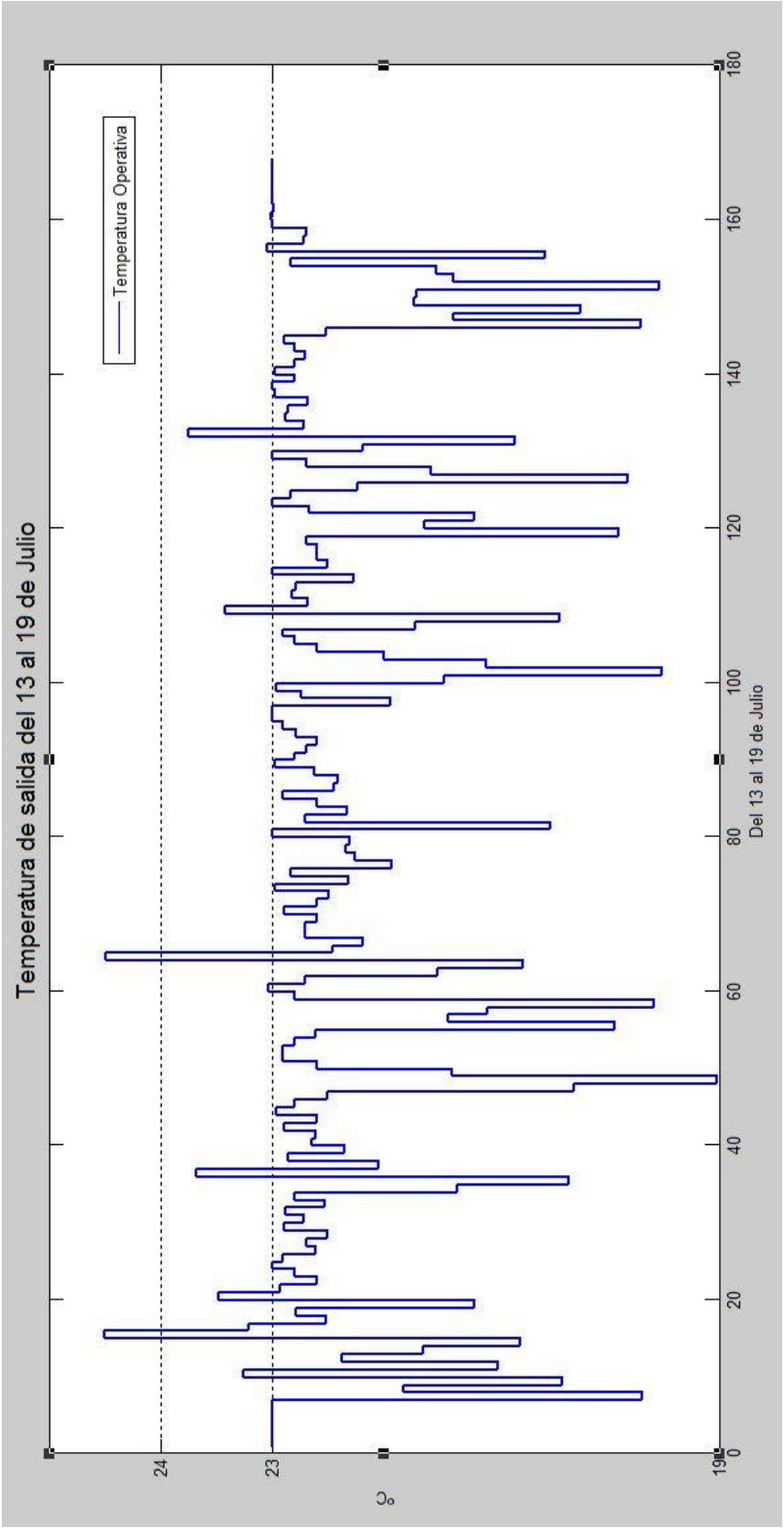

Figura 73. Temperatura de salida para la semana del 13 al 19 de Julio.

6.5.4. Conclusiones

Simulación para 1 día (13 de Julio)

Los primeros resultados obtenidos para la optimización de 1 día muestran una clara mejoría con respecto al consumo, de entorno el 4,16 % del total simulado.

Se muestra además un patrón claro de optimización, en el cual el algoritmo saca un valor bajo de setpoint (en torno a 19 grados) y a continuación introduce el apagado de la máquina, de manera que la inercia térmica parte desde un valor más bajo que un setpoint más alto.

Este patrón se reproduce a lo largo de la tarde, cuando comienza el descenso de temperatura operativa, donde las predicciones aprovechan dicha inercia para determinar que es conveniente introducir un apagado en la máquina.

No obstante, se observa en cuanto a la temperatura operativa de salida, que los límites son sobrepasados con un error bastante grande y no tolerable (llega a haber 25 grados). Esto se debe principalmente a errores en el modelo y a que se dejó el margen establecido en las pruebas 3 y 4 donde se había añadido un +/- 1 grado a los valores límites.

Por tanto, se ve necesario ajustar los límites a un margen más pequeño y aumentar el periodo de muestreo para obtener resultados más concluyentes.

Además, se aumentará la recompensa establecida para que el primer setpoint tenga más peso que el resto a la hora de introducir un apagado de la máquina.

Simulación para 3 días (13 al 15 de Julio)

En esta segunda prueba, se han introducido los cambios anteriormente descritos, obteniendo unos resultados mejores que para la simulación de 1 día, no en cuanto a optimización del consumo, pero si en cuanto a error en la temperatura de salida.

La mejora de consumo es alrededor de un 2%, poca cantidad con respecto a los valores anteriores. No obstante, vemos como se establece un patrón similar que en caso

anterior en el cual resulta más conveniente optimizar el consumo bajando primero el setpoint y metiendo el apagado de la máquina, y de nuevo durante las últimas horas de la tarde.

En esta ocasión, de 7 apagados, solo 3 superan límites de temperatura no tolerables, por lo que de nuevo hace falta ajustar al máximo los límites, entendiendo que debido a la corrección aplicada, la tolerancia en la temperatura debe ser 0, restaurando de esta manera los límites originales de 19 y 23 grados.

Para obtener resultados a largo plazo, ya que se depende completamente del día a simular y su aleatoriedad, se debe aumentar el periodo de simulación a 1 semana y observar los resultados obtenidos.

En cuanto a la corrección aplicada, se ha calculado una media por día en la que se observa que el sistema mejora a cada paso de simulación, de manera que la identificación se vuelve más precisa, con la mejora de rendimiento que eso supone.

Los valores medios de corrección por día se muestran en la Tabla 2.

Lunes	Martes	Miércoles
-0.4648	-0.2767	-0.058

Tabla 3. Corrección media por día

La conclusión a la que se llega es que efectivamente al aumentar el tiempo de simulación, el sistema se vuelve más preciso. Esto supone también que la optimización se vuelve más restrictiva, como se puede observar comparando los dos periodos de muestreo.

Los valores de pico de temperatura se deben a fallos e imprecisiones en la identificación.

Extrapolando a una simulación en mayor rango, se podría decir que para una mes la optimización resultaría en torno a un 2-3 %.

7. CONCLUSIONES

Pese a que en el apartado anterior se introdujo un subapartado de conclusiones para cada una de las propuestas de control planteadas, se muestra a continuación las conclusiones finales en tanto en cuanto los controles aplicados.

A medida que se iban planteando las propuestas, se observa la evolución de cada una con respecto a la anterior introduciendo aspectos de mejora y a la vez complicando el control.

Hay que señalar, que cada propuesta se enfrenta a condiciones diferentes de la máquina, de manera que la estrategia de control planteada puede ser válida contra ese estado.

Se observa que en general, las estrategias planteadas mejoran en cada caso al control propio de la máquina, inclusive el control por banda.

Los errores de temperatura no resultan excesivamente graves teniendo en cuenta que los márgenes de temperatura son orientativos y no algo obligado. Algún ajuste en la máquina de estados o añadir mejoras en el control (como mejorar la corrección) seguramente solucionasen dichas cuestiones.

Evidentemente, cada periodo de muestreo (entiéndase como los días de simulación) difieren bastante entre sí, lo que significa que un control positivo para un determinado periodo no quiere decir que lo sea para otro periodo distinto, y lo mismo ocurre al contrario.

Los resultados se han reflejado en el sistema real, En el cual intervienen otros factores además de la temperatura ambiente y la radiación, de ahí que existan ligeras discrepancias que se salen del rango del este trabajo (normalmente el cómo afectan variables como la humedad relativa, la densidad de personas, etc…).

Como posibles mejoras y planteamientos futuros, sería necesario realizar pruebas que, por falta de tiempo, esclareciesen los resultados finales en distintos periodos y condiciones. No es lo mismo un día de Junio que uno de Agosto.

A continuación se introduce un cuadro a modo de resumen con los diferentes resultados obtenidos.

Tabla 4. Cuadro comparativo de estrategias de control

ESTRATEGIAS DE CONTROL	OPTIMIZADOR	FUNCIONAMIENTO HVAC	CONSUMO ANTES	CONSUMO DESPUÉS	% DE RENDIMIENTO
CONTROL 1	MPC 1	SETPOINT (DUAL) FIJADO EN 20	1290,9 Kw	1007 Kw	28 %
CONTROL 2	MPC 2	SETPOINT FRIO FIJADO EN 21	3383,34 Kw	3162.003 Kw	7 %
CONTROL 3	MPC 3	BANDA DE SETPOINT 19 A 23	7439.2 Kw	7291.9 Kw	2 %

8. AGRADECIMIENTOS

Este trabajo no podría haberse realizado sin los conocimientos adquiridos durante el cursado del Máster en Automática en Informática Industrial.

La realización del presente trabajo final del Máster es fruto de las orientaciones, sugerencias y estímulo del Doctor Javier Sanchís Sáez, quien ha sabido guiarme durante estos meses de manera abierta y receptiva, resolviendo semanalmente las dudas presentadas y preocupándose por los problemas encontrados, de manera que me ha ayudado a mejorar dentro del proyecto, tanto profesional como personalmente.

Deseo agradecer también a la Universidad Politécnica de Valencia, que me ha brindado el material y las instalaciones necesarias para la elaboración del presente trabajo.

Y, por supuesto a mis familiares y amigos que supieron en todo momento el estado del trabajo, tanto las dificultades como los buenos resultados, y que han sabido apoyarme cuando más lo necesitaba, ya sea de manera técnica o emocionalmente.

9. REFERENCIAS

1] Martin Sanchez, J. (1976). Adaptive-Predictive Control System. 821.600.

[2] Koza, John. Genetic Programming: On the programming of computers by means of natural selection. A Bradford Book, 1992.

[3] EnergyPlus [en línea]. [Fecha de consulta: 13 de agosto 2016].

Disponible en: https://energyplus.net/.

[4] Matlab [en línea]. [Fecha de consulta: 18 de agosto 2016].

Disponible en: http://es.mathworks.com/products/matlab/.

[5] Simulink [en línea]. [Fecha de consulta: 18 de agosto 2016].

Disponible en: http://es.mathworks.com/products/simulink/.

[6] HVAC [en línea]. [Fecha de consulta: 25 de agosto 2016].

Disponible en: http://www.riback.com/hvac/.

[7] EPMPC [en línea]. [Fecha de consulta: 13 de marzo 2016].

Disponible en: http://www.ibpsa.org/proceedings/BS2013/p_1168.pdf

[8] MLE+: A Matlab-EnergyPlus Co-simulation Interface [en línea] [Fecha de consulta: 26 de marzo 2016]

Autor: Nghiem, Truong X.

Disponible en: http://www.seas.upenn.edu/~nghiem/mleplus.html

ANEXOS

ANEXO I

```
function [eplus_in_curr,userdata] = controlFile(cmd,eplus_out_prev,
eplus_in_prev, time, stepNumber, userdata)
% ---------------FUNCTION INPUTS---------------

% INPUTS TO ENERGYPLUS
% eplus_in_prev - Data Structure with all previous inputs

% OUTPUTS FROM ENERGYPLUS
% eplus_out_prev - Data Structure with all previous outpus

% OTHER INPUTS
% cmd  - MLE+ Command to distinguish init or normal step
% userdata  - user defined variable which can be changed and evolved
% time - vector with timesteps
% stepNumber - Number of Time Step in the simulation (Starts at 1)

% ---------------FUNCTION OUTPUTS---------------
% eplus_in_curr - vector with the values of the input parameters.
% This should be a 1xn vector where n is number of eplus_in parameters
% userdata - user defined variable which can be changed and evolved

if strcmp(cmd,'init')
    % ---------------WRITE YOUR CODE---------------
    eplus_in_curr.HeatSP = 19;
    eplus_in_curr.ColdSP = 23;
    %eplus_in_curr.SetPointCold = 23;
    %eplus_in_curr.SetPointHeat = 20;
elseif strcmp(cmd,'normal')
    % ---------------WRITE YOUR CODE---------------

    %% Control Predictivo Definitivo Para la Prueba 4.
    TempEnergyPlus=eplus_out_prev.TOperativaEP;
    ConEnergyPlus_Cold=eplus_out_prev.ConsumoCold;
    ConEnergyPlus_Heat=eplus_out_prev.ConsumoHeat;
    ConEnergyPlus=ConEnergyPlus_Heat+ConEnergyPlus_Cold;

    if stepNumber > 6
        if stepNumber < 164
            SP =
EjecutarGAPredictivo(stepNumber,eplus_in_prev.ColdSP,TempEnergyPlus,Co
nEnergyPlus);
        else
            SP=23;
        end
    else
        SP = 23;
    end

    HeatSP = 19;
    ColdSP = SP;

    %% Control Predictivo para setear valores de SP

%     SP1(:,1:25)=19;
```

```
%     SP2(:,1:25)=23;
%
%     HeatSP = SP1(stepNumber);
%     ColdSP = SP2(stepNumber);

    %% Máquina Apagada

%     HeatSP = 0;
%     ColdSP = 40;

eplus_in_curr.HeatSP = HeatSP;
eplus_in_curr.ColdSP = ColdSP;

end
end
```

ANEXO II

DatosSimulacion

```
%Simulación del 9-9 al 9-12 . Fichero de clima de Valencia E+. Fijando
SP
%entre 19 y 23 grados, sin tener nada más en cuenta.

%Cargamos los archivos exportados de la simulación.

%Entradas
SPCold = load('SPFrio');
TempAmbiente = load('Temperaturas');
RadiacionSolar = load('RadiacionSolar');


%Salidas
TemperaturaSalida = load('TemperaturaOperativa');
Consumos = load('Consumo');

%Extraemos los datos que nos interesan.

SPFrio = SPCold.data.result;
TemperaturaExt = TempAmbiente.data.result;
Radiacion = RadiacionSolar.data.result;


TemperaturaOperativa = TemperaturaSalida.data.result;
ConsumoHVAC = Consumos.data.result;

%Tranformamos los Julios a Kw/h

for i=1:size(ConsumoHVAC)
    ConsumoHVAC(i)=ConsumoHVAC(i)*(1/3600000);
end
%Creamos una estructura que almacene los dos valores de salidas y
además
%los 2 valores de entrada (Temp Ambiente, SP (Cooling)).

MatrizValores={SPFrio,TemperaturaExt,Radiacion,TemperaturaOperativa,Co
nsumoHVAC};

Graficar;
```

Graficar

```
t=1:1:size(TemperaturaExt,1);
figure
plot(t,TemperaturaExt)
title('Temperatura Ambiente y Operativa')
xlabel('Horas') % x-axis label
ylabel('°C') % y-axis label
hold on
plot(t,TemperaturaOperativa,'r')
```

```
figure
plot(t,SPFrio)
title('SetPoint de Frío')
xlabel('Horas') % x-axis label
ylabel('°C') % y-axis label
figure
plot(t,ConsumoHVAC)
title('Consumo de la máquina HVAC')
xlabel('Horas') % x-axis label
ylabel('KW/h') % y-axis label
figure
plot(t,Radiacion)
title('Radiación Solar')
xlabel('Horas') % x-axis label
ylabel('W/m2') % y-axis label
```

Identificacion

```
%Añadimos el path de los datos de simulación
cd('C:\Users\mluni\Desktop\TFM\DesignBuilder\Modelado5\Simulacion 3
dias\PruebaTemperaturaAmbiente\Resultado');
Datos1=load('Datos.mat');
Primera=cell2mat(Datos1.MatrizValores);



%Orden Entradas: SPFrio,TemperaturaExterior,Radiacion Solar,Humedad.
%Orden Salidas: TemperaturaInterior,Consumo.
Datos=Primera;
t=1:1:size(Datos,1);

%Calculamos entradas
u1=Datos(:,1);
u2=Datos(:,2);
u3=Datos(:,3);
y1=Datos(:,4);
y2=Datos(:,5);

%LLamamos al toolbox de identificación
ident
```

ANEXO III

```
function gaDat=ga(g)
%
% Basic Genetic Algorithm
%
%    gaDat=ga(gaDat)
%    gaDat : Data structure used by the algorithm.
%
% Data structure:
% Parameters that have to be defined by user
% gaDat.FieldD=[lb; ub]; % lower (lb) and upper (up) bounds of the
search space.
%                        % each dimension of the search space requires
bounds
% gaDat.Objfun='costFunction'; % Name of the 0bjective function to be
minimize
%
% Parameters that could be defined by user, in other case, there is a
default value
% gaDat.MAXGEN={gaDat.NVAR*20+10}; % Number of generation,
gaDat.NVAR*20+10 by default
% gaDat.NIND={gaDat.NVAR*50} ;   % Size of the population,
gaDat.NVAR*50 by default
% gaDat.alfa={0};                % Parameter for linear crossover, 0
by default
% gaDat.Pc={0.9};                % Crossover probability, 0.9 by
default
% gaDat.Pm={0.1};                % Mutation probability, 0.1 by
default
% gaDat.ObjfunPar={[]};          % Additional parameters of the
objective function
%                                %  have to be packed in a structure,
empty by default
% gaDat.indini={[]};             % Initialized members of the initial
population, empty
%                                %  by default
% Grupo de Control Predictivo y Optimización - CPOH
% Universitat Politècnica de València.
% http://cpoh.upv.es
% (c) CPOH  1995 - 2012

if nargin==1
    gaDat=g;
else
    error('It is necessary to pass a data structure: gaDat.FieldD and
gaDat.Objfun')
end
% If the parameter doesn't exist in the data structure it is created
with the default value
if ~isfield(gaDat,'NVAR')
    gaDat.NVAR=size(gaDat.FieldD,2);
end
if ~isfield(gaDat,'MAXGEN')
    gaDat.MAXGEN=gaDat.NVAR*20+10;
end
if ~isfield(gaDat,'NIND')
    gaDat.NIND=gaDat.NVAR*50;
end
```

```
if ~isfield(gaDat,'alfa')
    gaDat.alfa=0;
end
if ~isfield(gaDat,'Pc')
    gaDat.Pc=0.9;
end
if ~isfield(gaDat,'Pm')
    gaDat.Pm=0.1;
end
if ~isfield(gaDat,'ObjfunPar')
    gaDat.ObjfunPar=[];
end
if ~isfield(gaDat,'indini')
    gaDat.indini=[];
end

% Internal parameters
gaDat.Chrom=[];
gaDat.ObjV=[];
gaDat.xmin=[];
gaDat.fxmin=inf;
gaDat.xmingen=[];
gaDat.fxmingen=[];
gaDat.rf=(1:gaDat.NIND)';
gaDat.gen=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generation counter
gen=0;

% Initial population      ---------------------------------------
gaDat.Chrom=crtrp(gaDat.NIND,gaDat.FieldD);   % Real codification
% Individuals of gaDat.indini are randomly added in the initial
population
if not(isempty(gaDat.indini))
    nind0=size(gaDat.indini,1);
    posicion0=ceil(rand(1,nind0)*gaDat.NIND);
    gaDat.Chrom(posicion0,:)=gaDat.indini;
end

while (gaDat.gen<gaDat.MAXGEN),
    gaDat.gen=gen;
    gaDat=gaevolucion(gaDat);
    % Increase generation counter        ------------------
    gaDat.xmingen(gen+1,:)=gaDat.xmin;
    gaDat.fxmingen(gen+1,:)=gaDat.fxmin;
    gen=gen+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End main loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Present final results
garesults(gaDat)


%% Subfunction  ---------------------------------------
```

```
%% ------------------------------------------------------
function chrom=crtrp(Nind,FieldDR)
% A random real value matrix is created coerced by upper and
% lower bounds

Nvar = size(FieldDR,2);
aux = rand(Nind,Nvar);
m=[-1 1]*FieldDR;
ublb=ones(Nind,1)*m;
lb=ones(Nind,1)*FieldDR(1,:);
chrom=ublb.*aux+lb;
% chrom=round(chrom);

%% ------------------------------------------------------
function gaDat=gaevolucion(gaDat)
% One generation -------
Chrom=gaDat.Chrom;
nind=size(Chrom,1);
ObjV=inf(nind,1);
for i=1:nind
    if isempty(gaDat.ObjfunPar)
        ObjV(i)=feval(gaDat.Objfun,Chrom(i,:));
    else
        ObjV(i)=feval(gaDat.Objfun,Chrom(i,:),gaDat.ObjfunPar);
    end
end
gaDat.ObjV=ObjV;

% Best individual of the generation -------------------------
[v,p]=min(gaDat.ObjV);
if v<=gaDat.fxmin
    gaDat.xmin=Chrom(p,:);
    gaDat.fxmin=v;
end
% Next generation
% RANKING ------------------------------------------------
FitnV = ranking(gaDat.ObjV,gaDat.rf);
% SELECTION ----------------------------------------------
% Stochastic Universal Sampling (SUS).
SelCh = select('sus',Chrom,FitnV,1);
% CROSSOVER ----------------------------------------------------
% Uniform crossover.
SelCh = lxov(SelCh,gaDat.Pc,gaDat.alfa);
% MUTATION -----------------------------------------------
Chrom = mutbga(SelCh,gaDat.FieldD,[gaDat.Pm 1]); % Codificación Real.
% Reinsert the best individual  --------------------------
Chrom(round(gaDat.NIND/2),:) = gaDat.xmin;
gaDat.Chrom=Chrom;
% Optional additional task required by user
gaiteration(gaDat)

%% ------------------------------------------------------------
function FitV=ranking(ObjV,RFun)
% Ranking function
if nargin==1
    error('Ranking function needs two parameters');
end

if ~(length(ObjV)==length(RFun))
    error('RFun have to be of the same size than ObjV.');
```

```
end

[~,pos]=sort(ObjV);
FitV(pos)=flipud(RFun);
FitV=FitV';

%% -------------------------------------------------------------
function [SelCh]=select(SEL_F, Chrom, FitnV, GGAP)
% Selection Function
if (nargin==3) %  No overlap -------------------
    if strcmp(SEL_F,'rws')
        % Roulette wheel selection method
        indices=rws(FitnV,length(FitnV));
        SelCh=Chrom(indices,:);
    elseif strcmp(SEL_F,'sus')
        % Stochastic unversal sampling selection
        indices=sus(FitnV,length(FitnV));
        SelCh=Chrom(indices,:);
    else
        error('Incorrect selection method');
    end
elseif (nargin==4) % With overlap -----------------------------
    % Indexes of new individuals
    if strcmp(SEL_F,'rws')
        indices=rws(FitnV,round(length(FitnV)*GGAP));
    elseif strcmp(SEL_F,'sus')
        indices=sus2(FitnV,round(length(FitnV)*GGAP));
    else
        error('Incorrect selection method');
    end

    if (GGAP<1) % there is overlap
        % Members of the population to overlap
        oldpos=(1:length(FitnV))';
        for k=1:length(FitnV)
            pos=round(rand*length(FitnV)+0.5);
            % exchange indexes
            oldpos([pos k])=oldpos([k pos]);
        end
        oldpos=oldpos(1:round(length(FitnV)*GGAP));
        SelCh=Chrom;
        SelCh(oldpos,:)=Chrom(indices,:);
    else % more childs than parents
        SelCh=Chrom(indices,:);
    end
else
    error('Incorrect number of paramenters');
end

% Disorder the population.
[~,indi]=sort(rand(length(FitnV),1));
SelCh=SelCh(indi,:);

%% -------------------------------------------------------------------
function NewChrom =lxov(OldChrom, XOVR, alpha)
% Linear crossover
% Produce a~ new population by linear crossover and XOVR crossover
probability
%   NewChroms =lxov(OldChrom, XOVR, alpha, FieldDR)
%
```

```
% Linear recombination.
% Parameters 'beta1' and 'beta2' are randomly obtained inside [-alpha,
1+alpha]
% interval
%   Child1 = beta1*Parent1+(1-beta1)*Parent2
%   Child2 = beta2*Parent1+(1-beta2)*Parent2

if nargin==1
    XOVR = 0.7;
    alpha = 0;
elseif nargin==2
    alpha = 0;
end

n = size(OldChrom,1);   % Number of individuals and chromosome length
npares = floor(n/2);    % Number of pairs
cruzar = rand(npares,1)<= XOVR;     % Pairs to crossover
NewChrom=OldChrom;

for i=1:npares
    pin = (i-1)*2+1;
    if ~(cruzar(i)==0)
        betas=rand(2,1)*(1+2*alpha)-(0.5+alpha);
        A=[betas(1) 1-betas(1); 1-betas(2) betas(2)];
        NewChrom(pin:pin+1,:)=A*OldChrom(pin:pin+1,:);
    end
end

% Coerce points outside search space
% aux = ones(n,1);
% auxf1=aux*FieldDR(1,:);
% auxf2=aux*FieldDR(2,:);
% NewChrom =
(NewChrom>auxf2).*auxf2+(NewChrom<auxf1).*auxf1+(NewChrom<=auxf2 &
NewChrom>=auxf1).*NewChrom;

%% ---------------------------------------------------------------
function NewChrom=mutbga(OldChrom,FieldDR,MutOpt)
% Mutation function
% Real coded mutation.
% Mutation is produced adding a low random value
% OldChrom: Initial population.
% FieldChrom: Upper and lower bounds.
% MutOpt: mutation options,
%         MutOpt(1)=mutation probability (0 to 1).
%         MutOpt(2)=compression of the mutation value (0 to 1).
%         default MutOpt(1)=1/Nvar y MutOpt(2)=1

if (nargin==3)
    pm=MutOpt(1);
    shr=MutOpt(2);
elseif (nargin==2)
    pm=1/size(FieldDR,2);
    shr=1;
else
    error('Incorrect number of parameters');
end

Nind=size(OldChrom,1);
m1=0.5-(1-pm)*0.5;
```

```
m2=0.5+(1-pm)*0.5;
aux=rand(size(OldChrom));
MutMx=(aux>m2)-(aux<m1);
range=[-1 1]*FieldDR*0.5*shr;
range=ones(Nind,1)*range;
index=find(MutMx);
m=20;
alpha=rand(m,length(index))<(1/m);
xx=2.^(0:-1:(1-m));
aux2=xx*alpha;
delta=zeros(size(MutMx));
delta(index)=aux2;
NewChrom=OldChrom+(MutMx.*range.*delta);

% Coerce points outside bounds
aux = ones(Nind,1);
auxf1=aux*FieldDR(1,:);
auxf2=aux*FieldDR(2,:);
NewChrom =
(NewChrom>auxf2).*auxf2+(NewChrom<auxf1).*auxf1+(NewChrom<=auxf2 &
NewChrom>=auxf1).*NewChrom;

%% ------------------------------------------------------------
function NewChrIx=sus2(FitnV, Nsel)
suma=sum(FitnV);
% Position of the roulette pointers
j=0;
sumfit=0;
paso=suma/Nsel; % distance between pointers
flecha=rand*paso; % offset of the first pointer
NewChrIx(Nsel,1)=0;
for i=1:Nsel
    sumfit=sumfit+FitnV(i);
    while (sumfit>=flecha)
        j=j+1;
        NewChrIx(j)=i;
        flecha=flecha+paso;
    end
end

%% ------------------------------------------------------------
```

```
%Cargamos el fichero de <idprocs>
load('FDTs.mat');
%Parametrizamos s para construir las funciones
s=tf('s');

%Sacamos las Kp
Kp13=SPTemp_FDT.Kp.value;
Kp14=ArranqueTemp_FDT.Kp.value;
Kp12=TempAmbiente_FDT.Kp.value;
Kp11=RadSolar_FDT.Kp.value;
Kp21=RadSolarConsumo_FDT.Kp.value;
Kp22=TempConsumo_FDT.Kp.value;
Kp23=SPConsumo_FDT.Kp.value;
Kp24=ArranqueSP_FDT.Kp.value;
%Sacamos las Tp1
Tp13=SPTemp_FDT.Tp1.value;
Tp14=ArranqueTemp_FDT.Tp1.value;
Tp12=TempAmbiente_FDT.Tp1.value;
Tp11=RadSolar_FDT.Tp1.value;
Tp21=RadSolarConsumo_FDT.Tp1.value;
Tp22=TempConsumo_FDT.Tp1.value;
Tp23=SPConsumo_FDT.Tp1.value;
Tp24=ArranqueSP_FDT.Tp1.value;

%Sacamos las Td
Td13=SPTemp_FDT.Td.value;
Td12=TempAmbiente_FDT.Td.value;
Td11=RadSolar_FDT.Td.value;
Td22=TempConsumo_FDT.Td.value;

%Sacamos las Tz
Tz23=SPConsumo_FDT.Tz.value;
Tz21=RadSolarConsumo_FDT.Tz.value;
Tz24=ArranqueSP_FDT.Tz.value;
Tz14=ArranqueTemp_FDT.Tz.value;

%Construimos las G, añadiendo iodelay o Tz según precise
g11=Kp11/(1+Tp11*s);
g11.iodelay=Td11;

g12=Kp12/(1+Tp12*s);
g12.iodelay=Td12;

g13=Kp13/(1+Tp13*s);
g13.iodelay=Td13;

g14=Kp14*(1+Tz14*s)/(1+Tp14*s);

g21=Kp21*(1+Tz21*s)/(1+Tp21*s);

g22=Kp22/(1+Tp22*s);
g22.iodelay=Td22;

g23=Kp23*(1+Tz23*s)/(1+Tp23*s);
```

```
g24=Kp24*(1+Tz24*s)/(1+Tp24*s);

%Cargamos las entradas de TemperaturaAmbiente y Radiación
(Predicciones)
load('RadiacionPrueba.mat');
Radiacion = data.result;
load('TempPrueba.mat');
Temp = data.result;

t=1:1:size(Temp);

%Creamos el espacio de estados para cada una de las funciones
anteriormente descritas
[a11,b11,c11,d11]=tf2ss(g11.num{1},g11.den{1});
[a12,b12,c12,d12]=tf2ss(g12.num{1},g12.den{1});
[a13,b13,c13,d13]=tf2ss(g13.num{1},g13.den{1});
[a14,b14,c14,d14]=tf2ss(g14.num{1},g14.den{1});
[a21,b21,c21,d21]=tf2ss(g21.num{1},g21.den{1});
[a22,b22,c22,d22]=tf2ss(g22.num{1},g22.den{1});
[a23,b23,c23,d23]=tf2ss(g23.num{1},g23.den{1});
[a24,b24,c24,d24]=tf2ss(g24.num{1},g24.den{1});

%Añadir solo si te utiliza Lsim
% sys11=ss(a11,b11,c11,d11);
% sys12=ss(a12,b12,c12,d12);
% sys13=ss(a13,b13,c13,d13);
% sys14=ss(a14,b14,c14,d14);
% sys21=ss(a21,b21,c21,d21);
% sys22=ss(a22,b22,c22,d22);
% sys23=ss(a23,b23,c23,d23);
% sys24=ss(a24,b24,c24,d24);

%Realizamos la extraccion de la media
Radiacion_mean=detrend(Radiacion,0);
Temp_mean=detrend(Temp,0);

%Construimos los vectores finales que empleará Simulink. Se necesita
un
%vector de tiempos y un valor para cada instante
u1=[t',Radiacion_mean];
u2=[t',Temp_mean];

%Cargamos la Temperatura Operativa y extraemos el estado inicial y
%offset del mismo. Para cada nuevo parámetros estimado, se recalcula
el
%estado inicial para crear el nuevo horizonte de predicción
load('TempSalida.mat');
TempSalida=data.result;
Offset=data.result(1,1);
val=detrend(TempSalida,0);
valinicial=val(1,1);

%Realizamos el mismo proceso para la salida de Consumo, teniendo en
cuenta
%que por defecto viene en Julios, los cuales convertiremos a Kw/h
load('ConsumoSalida.mat');
OffsetCon=data.result(1,1)/3600000;
ConE=data.result/3600000;
valCon=detrend(ConE,0);
valinicialCon=valCon(1,1);
```

Para la Prueba 2

```
clear all
close all
clc

for i=1:24              %Iteramos para 1 día (24 horas)
        if i > 5        %Empezamos a partir de las 5:00 AM
            if i < 21   %Terminamos a las 20:00 PM
                SP=EjecutarGA_Prueba1(i,SPFinal);
            else
                SP=20;
            end
        else
            SP=20;
        end
        SPFinal(i)=SP(1);   %Cogemos el último valor de SP
end

SPFinal;
```

Para la Prueba 3

```
clear all
close all
clc

for i=1:168                     %Iteramos para 1 semana entera
    if rem(i,2) == 1            %Cogemos unicamente los valores cada 2
horas
        if i > 1                %La primera hora dejamos un valor fijo
            if i < 159          %Cerramos a las 15:00 PM del último
día
                SP=EjecutarGAPredictivo(i,SPFinal);
            else
                SP=[20 20];
            end
        else
            SP=[20 20];
        end
        SPFinal(i)=SP(1);
        SPFinal(i+1)=SP(2);     %Cogemos el SP actual y de la hora
siguiente
    end
end

SPFinal;
```

ANEXO VI.1

```
function SP = EjecutarGA_Prueba1y2(steptime,SPAnterior)

%% Basic GA parameters
warning ('off','all');

gaDat.Objfun='consumoFinal_Prueba1';        % Funcion de coste
lb=[19 19 19];                              % Aumentar limites a 20
para Prueba 1
ub=[23 23 23];
Datos={steptime SPAnterior};                % Eliminar para Prueba 1
gaDat.ObjfunPar=Datos;
gaDat.FieldD=[lb; ub];
% Execute GA
gaDat.MAXGEN=3;
gaDat.NIND=25;
gaDat=ga(gaDat);
SP=gaDat.xmin;
SP=round(SP);
SP
gaDat.fxmin

end
```

ANEXO VI.2

```
function [sumCon] = consumoFinal_Prueba1y2(SPFinal,Datos)
%CONSUMO Summary of this function goes here
%Detailed explanation goes here
CargarSimulinkGAVerano;   %Cargamos Datos Verano
steptime=Datos{1,1};      %Para la prueba 2, Para la prueba 1 dejar en
24;
SPAnterior=Datos{1,2};    %Para la prueba 2, Para la prueba 1 eliminar

SP(1:24,:)=20;

SPFinal=round(SPFinal);   %A partir de aquí si es la Prueba 1
SP(5:24,:)=SPFinal
                          %y omitir el resto hasta SP_mean

SP(1:(steptime-1))=SPAnterior;
SP(steptime:(steptime+2),:)=SPFinal;
SP((steptime+3):24,:)=SPFinal(3);

SP_mean=detrend(SP,0);
t=1:1:size(SP);
u3 = [t',SP_mean];
j=1;
options = simset('SrcWorkspace','current');
sim('ModeloSimulink1.slx',[1 24],options);    %Simulamos
penalti = 10^9;
```

```
j=1;
k=1;
l=1;
m=1;
for i=1:size(tout,1)              %Nos quedamos con los enteros
    if mod(Consumo(i,1),1) == 0
        Con(j,:)=Consumo(i,2);
        j=j+1;
    end
    if mod(TempOperativa(i,1),1) == 0
        TOperativa(k,:)=TempOperativa(i,2);
        k=k+1;
    end
    if mod(TempOperativaSP(i,1),1) == 0
        TempSPOperativa(m,:)=TempOperativaSP(i,2);
        m=m+1;
    end
end

for i=1:size(SP,1)
    if SP(i) < (TOperativa(i))
        TempSal(i) = SP(i);
    elseif SP(i) >= (TOperativa(i))
        TempSal(i) = TOperativa(i);
        Con(i) = 0;   %Ponemos el consumo a 0 si SP > TOperativa
    end
end

%Cogemos los datos entre 5:00 AM y 20:00 PM para la Prueba 1 y entre
%5:00 AM y 24:00 AM para la Prueba 1
TempSal=TempSal';
TempSal=TempSal(5:20,:);
Con=Con(steptime:(steptime+4),:);

sumCon = sum(Con);


for i=1:size(TempSal,1)              %Añadimos penalti si precisa
    if       TempSal(i) < 17        %Debido al modelo le metemos un
márgen
        sumCon=sumCon+penalti;
    elseif   TempSal(i) > 24
        sumCon=sumCon+penalti;
    end
end
% disp('-------------------------------------------------')
% disp(['   xmin: ' mat2str(SPFinal) ' -- f(xmin): ',num2str(sumCon)])
end
```

ANEXO VII.1

```
function SP = EjecutarGAPredictivo_Prueba3(step,SPAnterior)

%% Basic GA parameters
warning ('off','all');
gaDat.Objfun='consumoFinalPredictivo_Prueba3';       %% Funcion de
coste
Datos={step SPAnterior};
gaDat.ObjfunPar=Datos;
lb(:,1:3)=0;
ub(:,1:3)=1;
gaDat.FieldD=[lb; ub];
% Execute GA
gaDat.MAXGEN=5;
gaDat.NIND=25;
gaDat=ga(gaDat);
SPFinal=gaDat.xmin;
for i=1:size(SPFinal,2)
    if SPFinal(i) < 0.5
        SPFinal(i) = 0;
    elseif SPFinal(i) > 0.5
        SPFinal(i) = 20;
    end
end

SP=SPFinal(:,1:2);
end
```

ANEXO VII.2

```
function [sumCon] = consumoFinalPredictivo_Prueba3(SPFinal,Datos)
%CONSUMO Summary of this function goes here
%Detailed explanation goes here
%% Cargamos el SP anterior y el nuevo, para simular el estado
steptime=Datos{1,1};
SPAnterior=Datos{1,2};
CargarSimulinkVeranoPredictivo;%Cargamos Datos Verano

%Preparamos 3 variables, SP siempre encendido, SP con los apagados y
un SP
%para calcular el offset de la temperatura operativa (SPCon)
SP(1:168,:)=20;
SPCon(1:168,:)=20;
SP_ON(1:168,:)=20;


for i=1:size(SPFinal,2)                 %Seleccionamos si la máquina
se apaga o enciende
    if SPFinal(i) < 0.5
        SPFinal(i) = 0;
    elseif SPFinal(i) > 0.5
        SPFinal(i) = 20;
    end
```

```
end

if steptime < 167                       %limitador para el ultimo
valor de hora.
    a=2;
    SP_ON((steptime):(steptime+a))=SPFinal;
else
    a=168-steptime;
    SP_ON((steptime):(steptime+a))=SPFinal;
end

if steptime < 158                       %Seleccionamos el tercer SP
para las siguientes 7 horas (así tenemos 10 horas).
    a=3;
    b=9;
    SP_ON((steptime+a):(steptime+b))=SPFinal(3);
else
    if steptime < 166                   %Limitador para que las
ultimas 3 horas no de error.
        a=3;
        b=168;
        SP_ON((steptime+a):(b))=SPFinal(3);
    else
        a=168-steptime;
        b=168;
        SP_ON((steptime+a):(b))=SPFinal(3);
    end
end

for i=1:size(SPFinal,2)
    if SPFinal(i) == 0;
        SP(steptime-1+i) = 0;
        SPCon(steptime-1+i) = 0;
        SP_ON(steptime-1+i) = SP_ON(steptime+i-2);
    elseif SPFinal(i) > 0;
        SP(steptime-1+i) = SPFinal(i);
    end
end

for i=1:size(SPAnterior,2)
    if SPAnterior(i) == 0
        SP(i)=0;
    end
end

contador=1;

for i=1:size(SPCon,1)           %Offset de la temperatura de arranque
    if SPCon(i) == 0
        SPOffset = SP_ON(i-1);
        contador=1;
    else
        contador=contador+1;
    end
    if contador > 158
        SPOffset = SP(1);
        contador=1;
    end
end
```

```
if steptime < 158                %Limitador para evitar errores a las
10 horas
    a=3;
    b=9;
    SP((steptime+a):(steptime+b))=SP(steptime+a-1);
else
    if steptime < 166
        a=3;
        b=168;
        SP((steptime+a):(b))=SP(steptime+a-1);
    else
        a=168-steptime;
        b=168;
        SP((steptime+a):(b))=SP(steptime+a-1);
    end
end

for i=1:size(SP_ON,1)
    if SP_ON(i) == 0
        SP_ON(i)=SP_ON(i-1);
    end
end


SP_ON_mean=detrend(SP_ON,0);
SP_mean=detrend(SP,0);
t=1:1:size(SP_ON);
u3 = [t',SP_ON_mean];
u4 = [t',SP_mean];
options = simset('SrcWorkspace','current');
sim('ModeloSimulinkFinalPredictivo1.mdl',[1 72],options);
%Simulamos
penalti = 10^9;

j=1;
k=1;
l=1;
m=1;
n=1;
for i=1:size(Consumo,1)              %Nos quedamos con los enteros
    if mod(tout(i),1) == 0
        Con(j,:)=Consumo(i);
        j=j+1;
    end
    if mod(tout(i),1) == 0
        TOperativa(k,:)=TempOperativa(i);
        k=k+1;
    end
    if mod(tout(i),1) == 0
        TempSPArranque(l,:)=TempArranque(i);
        l=l+1;
    end
    if mod(tout(i),1) == 0
        TempSPOperativa(m,:)=TempOperativaSP(i);
        m=m+1;
    end
    if mod(tout(i),1) == 0
        Con_arranque(n,:)=ConArranque(i);
        n=n+1;
    end
end
```

```
SP_Arranque=SP;

%Máquina de estados de temperatura conforme a las entradas de SP al
sistema.
%Tomará una salida u otra según convenga
for i=1:size(SP_Arranque,1)
    if SP_Arranque(i,1) == 0
        if TempSPArranque(i) < (TOperativa(i))
        TempSal(i) = TempSPArranque(i);
        elseif TempSPArranque(i) >= (TOperativa(i))
        TempSal(i) = TOperativa(i);
        end
    else
        if SP_ON(i) < (TOperativa(i))
        TempSal(i) = SP_ON(i);
        elseif SP_ON(i) >= (TOperativa(i))
        TempSal(i) = TOperativa(i);
        end
    end
end

%Ajuste de consumo
for i=1:size(SP_Arranque,1)
    if SP_Arranque(i,1) == 0
        Con(i) = 0;
    end
end

for i=1:size(Con,1)
    if TempSal(i) < SP_ON(i)
        Con(i) = 0;
    end
end

%Cogemos el consumo para las 10 horas siguientes
if steptime < 159
    a=10;
    Consumo10horas = Con(steptime:steptime+a);
else
    a=168;
    Consumo10horas = Con(steptime:a);
end

ConsumoMedio = sum(Consumo10horas)/10;

TempSal=TempSal';
sumCon = ConsumoMedio;
for i=(steptime):size(TempSal,1)            %Añadimos penalti si
precisa
    if       TempSal(i) < 18
        sumCon=sumCon+penalti;
    elseif   TempSal(i) > 24.2
        sumCon=sumCon+penalti;
    end
end
%  disp('-----------------------------------------------')
%  disp(['   xmin: ' mat2str(ONOFF) ' -- f(xmin): ',num2str(sumCon)])
end
```

ANEXO VIII.1

```
function SP = EjecutarGAPredictivo(step,SPAnterior)

%% Basic GA parameters
warning ('off','all');
gaDat.Objfun='consumoFinalPredictivo';          %Funcion de coste
Datos={step SPAnterior};                          %Datos adicionales de
entrada
gaDat.ObjfunPar=Datos;
lb(:,1:3)=17;                                        %Seleccionamos los
límites
ub(:,1:3)=25;
gaDat.FieldD=[lb; ub];
% Execute GA
gaDat.MAXGEN=5;                                  %Configuramos el GA
gaDat.NIND=50;
gaDat=ga(gaDat);                                 %Lanzamos el algoritmo
SPFinal=gaDat.xmin;                              %Redondeamos el SP
SPFinal=SPFinal.*10;
SPFinal=round(SPFinal);
SPFinal=SPFinal/10;
for i=1:size(SPFinal,2)                                %Colocamos los 0
correctamente
    if SPFinal(i) < 19
        SPFinal(i) = 0;
    elseif SPFinal(i) > 23
        SPFinal(i) = 0;
    else
        SPFinal(i) = SPFinal(i);
    end
end
SP=SPFinal(:,1:2);
end
```

ANEXO VIII.2

```
function [sumCon] = consumoFinalPredictivo(SPFinal,Datos)
%CONSUMO Summary of this function goes here
%Detailed explanation goes here
%% Cargamos el SP anterior y el nuevo, para simular el estado
steptime=Datos{1,1};
SPAnterior=Datos{1,2};
CargarSimulinkVeranoPredictivo;%Cargamos Datos Verano

SPFinal=SPFinal.*10;                             %redondeamos
SPFinal=round(SPFinal);
SPFinal=SPFinal/10;
SP(1:72,:)=20;                                     %Creamos los vectores
de SP correspondiente
SPCon(1:72,:)=20;
SP_ON(1:72,:)=20;
SP_ON(1:(steptime-1))=SPAnterior;               %Cargamos el SP anterior

for i=1:size(SPFinal,2)                          %Ponemos los 0
    if SPFinal(i) < 19
        SPFinal(i) = 0;
    elseif SPFinal(i) > 23
        SPFinal(i) = 0;
```

```
    else
        SPFinal(i) = SPFinal(i);
    end
end

if steptime < 71                                    %Limitamos para evitar
errores
    a=2;
    SP_ON((steptime):(steptime+a))=SPFinal;
else
    a=72-steptime;
    SP_ON((steptime):(steptime+a))=SPFinal;
end

if steptime < 60
    a=3;
    b=9;
    SP_ON((steptime+a):(steptime+b))=SPFinal(3);
else
    if steptime < 70
        a=3;
        b=72;
        SP_ON((steptime+a):(b))=SPFinal(3);
    else
        a=72-steptime;
        b=72;
        SP_ON((steptime+a):(b))=SPFinal(3);
    end
end



for i=1:size(SPFinal,2)                             %Introducimos los
apagados al sistema
    if SPFinal(i) == 0;
        SP(steptime-1+i) = 0;
        SPCon(steptime-1+i) = 0;
        SP_ON(steptime-1+i) = SP_ON(steptime+i-2);
    elseif SPFinal(i) > 0;
        SP(steptime-1+i) = SPFinal(i);
    end
end

for i=1:size(SPAnterior,2)
    if SPAnterior(i) == 0
        SP(i)=0;
    end
end

contador=1;

for i=1:size(SPCon,1)                               %Configuramos uan
variable para la temperatura de arranque
    if SPCon(i) == 0
        SPOffset = SP_ON(i-1);
        contador=1;
    else
        contador=contador+1;
    end
    if contador > 70
```

```
        SPOffset = SP(1);
        contador=1;
    end
end

if steptime < 60
    a=3;
    b=9;
    SP((steptime+a):(steptime+b))=SP(steptime+a-1);
else
    if steptime < 70
        a=3;
        b=72;
        SP((steptime+a):(b))=SP(steptime+a-1);
    else
        a=72-steptime;
        b=72;
        SP((steptime+a):(b))=SP(steptime+a-1);
    end
end

for i=1:size(SP_ON,1)
    if SP_ON(i) == 0
        SP_ON(i)=SP_ON(i-1);
    end
end

%Declaramos las variables que faltan y arrancamos la simulación
SP_ON_mean=detrend(SP_ON,0);
SP_mean=detrend(SP,0);
t=1:1:size(SP_ON);
u3 = [t',SP_ON_mean];
u4 = [t',SP_mean];
options = simset('SrcWorkspace','current');
sim('ModeloSimulinkFinalPredictivo1.mdl',[1 168],options);    %Simulamos
penalti = 10^9;

j=1;
k=1;
l=1;
m=1;
n=1;
for i=1:size(Consumo,1)                %Nos quedamos con los enteros
    if mod(tout(i),1) == 0
        Con(j,:)=Consumo(i);
        j=j+1;
    end
    if mod(tout(i),1) == 0
        TOperativa(k,:)=TempOperativa(i);
        k=k+1;
    end
    if mod(tout(i),1) == 0
        TempSPArranque(l,:)=TempArranque(i);
        l=l+1;
    end
    if mod(tout(i),1) == 0
        TempSPOperativa(m,:)=TempOperativaSP(i);
        m=m+1;
    end
    if mod(tout(i),1) == 0
        Con_arranque(n,:)=ConArranque(i);
```

```
        n=n+1;
    end
end

SP_Arranque=SP;
%Máquina de estados para extraer la salida de temperatura
for i=1:size(SP_Arranque,1)
    if SP_Arranque(i,1) == 0
        if TempSPArranque(i) < (TOperativa(i))
        TempSal(i) = TempSPArranque(i);
        elseif TempSPArranque(i) >= (TOperativa(i))
        TempSal(i) = TOperativa(i);
        end
    else
        if SP_ON(i) < (TOperativa(i))
        TempSal(i) = SP_ON(i);
        elseif SP_ON(i) >= (TOperativa(i))
        TempSal(i) = TOperativa(i);
        end
    end
end
%Maquina de estados para extraer la salida de consumo
for i=1:size(SP_Arranque,1)
    if SP_Arranque(i,1) == 0
        Con(i) = 0;
    end
end


%Completamos el consumo
for i=1:size(Con,1)
    if TempSal(i) < SP_ON(i)
        Con(i) = 0;
    end
end

if steptime < 164
    a=5;
    Consumo5horas = Con(steptime:steptime+a);
else
    a=168;
    Consumo5horas = Con(steptime:a);
end
%Optimizamos el consumo
ConsumoMedio = sum(Consumo5horas)/10;

TempSal=TempSal';
sumCon = ConsumoMedio;
for i=(steptime):size(TempSal,1)            %Añadimos penalti si precisa
    if       TempSal(i) < 18
        sumCon=sumCon+penalti;
    elseif   TempSal(i) > 24.2
        sumCon=sumCon+penalti;
    end
end
end
```

ANEXO IX.1

```
function  [eplus_in_curr,userdata]  =  controlFile(cmd,eplus_out_prev,
eplus_in_prev, time, stepNumber, userdata)


% ---------------DECLARACION DE VARIABLES PERSISTENTS--------------
    persistent TOperativaAnt
    persistent PrediccionAnterior
    persistent Correcciones
```

```
% ---------------FUNCTION INPUTS---------------

% INPUTS TO ENERGYPLUS
% eplus_in_prev - Data Structure with all previous inputs

% OUTPUTS FROM ENERGYPLUS
% eplus_out_prev - Data Structure with all previous outpus

% OTHER INPUTS
% cmd  - MLE+ Command to distinguish init or normal step
% userdata  - user defined variable which can be changed and evolved
% time - vector with timesteps
% stepNumber - Number of Time Step in the simulation (Starts at 1)

% ---------------FUNCTION OUTPUTS---------------
% eplus_in_curr - vector with the values of the input parameters.
% This should be a 1xn vector where n is number of eplus_in parameters
% userdata - user defined variable which can be changed and evolved

if strcmp(cmd,'init')
    % ---------------WRITE YOUR CODE---------------
    eplus_in_curr.HeatSP = 19;
    eplus_in_curr.ColdSP = 23;
    %eplus_in_curr.SetPointCold = 23;
    %eplus_in_curr.SetPointHeat = 20;
elseif strcmp(cmd,'normal')
    % ---------------WRITE YOUR CODE---------------

    %% Control Predictivo Definitivo Para la Prueba 4.
    load('TOperativaAnt.mat')
    TOperativaEP = data.result;

    if stepNumber > 6
        if stepNumber == 7
            TOperativaAnt  =  TOperativaEP(24+stepNumber-1);    %Sacamos
un primer valor de referencia directamente del modelo real
            PrediccionAnterior = 0;
            Correcciones(1:168,:) = 0;
        end
        if stepNumber < 159                                    %Simulamos
para 168 horas
            [SP,TOperativaAct,PrediccionActual,correccionActual]      =
EjecutarGAPredictivo(stepNumber,eplus_in_prev.ColdSP,TOperativaAnt,Pre
diccionAnterior);
        else
            SP=23;
        end
    else
        SP = 23;
    end

    HeatSP = 19;
    ColdSP = SP;

    if stepNumber > 6
        if stepNumber < 159
            TOperativaAnt                =                TOperativaAct;
%Guardamos la temperatura actual para calcular la corrección en t+1.
            PrediccionAnterior = PrediccionActual;
            Correcciones(stepNumber) = correccionActual;
```

```
        end
    end

    if stepNumber == 167                                    %Guardamos
las correcciones aplicadas
        filename='Correcciones.mat';
        save(filename,'Correcciones');
    end

eplus_in_curr.HeatSP                    =                   HeatSP;
%Introducimos los SP al EnergyPlus por medio de la ExternalInterface
eplus_in_curr.ColdSP = ColdSP;

end
end
```

ANEXO IX.2

```
function        [SP,TOperativaAct,PrediccionActual,correccion]        =
EjecutarGAPredictivo(step,SPAnterior,TOperativaAnt,PrediccionAnterior)
%% Basic GA parameters
warning ('off','all');
gaDat.Objfun='consumoFinalPredictivo';
%Funcion de coste
Datos={step                 SPAnterior                TOperativaAnt};
%Introducimos los datos necesarios para la función
gaDat.ObjfunPar=Datos;
lb(:,1:3)=17;
%Designamos los límites del GA
ub(:,1:3)=25;
gaDat.FieldD=[lb; ub];
% Execute GA
gaDat.MAXGEN=10;
%Configuramos las generaciones y los individuos y población inicial
gaDat.NIND=100;
% if step > 7
% gaDat.indini=PrediccionAnterior;
% end
gaDat=ga(gaDat);
%Ejecutamos el GA
SPFinal=gaDat.xmin;                                         %Sacamos
el mejor resultado
[ConM,TOperativaAct,correccion]                                      =
consumoFinalPredictivo(gaDat.xmin,Datos);     %Extraemos la temperatura
actual y el consumo actual para usarlos en t+1
SPFinal=SPFinal.*10;
%Redondeamos a 1 décima
SPFinal=round(SPFinal);
SPFinal=SPFinal/10;

for                                                 i=1:size(SPFinal,2)
%Transformamos los 0 en 40 para que EnergyPlus pueda procesarlos
    if SPFinal(i) < 19
        SPFinal(i) = 40;
    elseif SPFinal(i) > 23
        SPFinal(i) = 40;
    else
        SPFinal(i) = SPFinal(i);
    end
```

```
end

SP=SPFinal(:,1);
%Devolvemos el SP actual para usarlo como población inicial en la
siguiente iteración
PrediccionActual = gaDat.xmin;
end
```

ANEXO IX.3

```
function [ConsumoMedio,TOperativaAct,correccion_temp] =
consumoFinalPredictivo(SPPred,Datos)
%CONSUMO Summary of this function goes here
%Detailed explanation goes here
%% Cargamos el SP anterior y el nuevo, para simular el estado
steptime=Datos{1,1};                                          %Hora
en la que estoy
B=steptime+10;                                              %Variable
para simular los 10 pasos siguientes
SPAnt=Datos{1,2};                                             %Cargo
el Setpoint Anterior
TOperativaAnt=Datos{1,3};                                     %Cargo
la Temperatura Operativa del modelo (la de las horas anteriores
CargarDatosSimulacion;                                        %Carga
un script para construir las funciones de transferencia y coger las
predicciones

SPPred=SPPred.*10;                                          %Redondeamos
a la décima
SPPred=round(SPPred);
SPPred=SPPred/10;
SP(1:B,:)=21;                                                 %Creamos
los vectores de Setpoint.
SPCon(1:B,:)=23;
SP_ON(1:B,:)=23;

%Como el rango es 17-25, le decimos que entre 17 y 19 sea 0 y entre 23
y 25
%sea también 0, así hay un 50 % de posibilidades de que se apague o
%encienda ( y dentro del encendido el valor será entre 19 y 23).
for i=1:size(SPPred,2)
    if SPPred(i) < 19
        SPPred(i) = 0;
    elseif SPPred(i) > 23
        SPPred(i) = 0;
    else
        SPPred(i) = SPPred(i);
    end
end

%Como en EnergyPlus los apagados de la máquina de frío son SP altos (los
%fijo en 40) se transforman a 0.
for i=1:size(SPAnt,2)
    if SPAnt(i) == 40
        SPAnterior(i) = 0;
    else
        SPAnterior(i) = SPAnt(i);
    end
end
```

```
%Creamos el SP siempre encendido
for i=1:size(SPAnterior,2)
    if SPAnterior(i) == 0
        if SPAnterior(i-1) ~= 0
            SP_ON(i) = SPAnterior(i-1);
        else
            SP_ON(i) = 21;
        end
    else
        SP_ON(i) = SPAnterior(i);
    end
end

SP(1:size(SPAnterior,2))=SPAnterior;

%Añadimos el SP estimado por el GA y las siguientes horas lo igualamos
al
%tercer valor
for i=1:(size(SPPred,2)+8)
    if i <= size(SPPred,2)
        if SPPred(i) == 0;
            SP(i+steptime-1) = 0;
            SPCon(i+steptime-1) = 0;
            if SP_ON(i+steptime-2) ~= 0
            SP_ON(i+steptime-1) = SP_ON(i+steptime-2);
            else
            SP_ON(i+steptime-1) = 21;
            end
        elseif SPPred(i) > 0;
            SP(i+steptime-1)= SPPred(i);
            SP_ON(i+steptime-1)=SPPred(i);
        end
    else
        SP(i+steptime-1)=SP(i+steptime-2);
        SP_ON(i+steptime-1)=SP_ON(i+steptime-2);
    end
end

%Creamos una variable para situar la temperatura de arranque (desde que
SP
%parte)
contador = 1;
for i=1:size(SPCon,1)
    if SPCon(i) == 0
        SPOffset = SP_ON(i-1);
        contador=1;
    else
        contador=contador+1;
    end
    if contador > 9
        SPOffset = SP(1);
        contador=1;
    end
end

u14_Arranque = SP - 21;

%Calculamos la correccion que hace falta aplicarle a la salida de
%temperatura para ajustarla con el valor real.
```

```
correccion_temp = TOperativaRef(steptime-1)-TOperativaAnt;

%Declaramos las entradas que nos faltan
SP_ON_mean=detrend(SP_ON,0);
SP_mean=detrend(SP,0);
t=1:1:size(SP_ON);

%Realizamos la simulacion de la temperatura en base a la temperatura
%ambiente y la radiacion solar.
[y11]=lsim(sys11,Radiacion_mean,t,0);
[y12]=lsim(sys12,Temp_mean,t,0);

TOperativa=(y11+y12)+Offset;
TOperativa_corregido= TOperativa + correccion_temp;                %Le
sumamos la correción


%Simulamos la temperatura de arranque, que solo la tendremos en cuenta
%cuando se realice un apagado de la máquina
[y14]=lsim(sys14,u14_Arranque,t,0);
TempArranque = y14;

%Simulamos el consumo total. En este caso no hace falta aplicar un offset
%ya que da igual si le sumamos o quitamos, a la hora de calcular la
media
%de consumo no afecta.
[y21]=lsim(sys21,Radiacion_mean,t,0);
[y22]=lsim(sys22,Temp_mean,t,0);
[y23]=lsim(sys23,SP_ON_mean,t,0);
[y24]=lsim(sys24,SP_mean,t,0);             %Valor a 0 cuando hay un SP
fijado.

%Le añadimos el offset correspondiente
Con = (y21+y22+y23+y24)+OffsetCon;


%Creamos el penalti para la minimización del consumo y un premio
prize = -50;
penalti = 10^9;

TempSal(1:size(SP,1),:) = 0;
TempOperativa(1:size(SP,1),:) = 0;

%Máquina de estados para sacar la temperatura de salida total del sistema
for i=1:size(SP,1)
    if SP(i) ~=0
        TempOperativa(i) = SP(i);
    else
        TempOperativa(i) = TOperativa_corregido(i);
    end
end

for i=2:size(SP,1)
    if SP(i) == 0
        TempSal(i) = TempOperativa(i-1)+TempArranque(i);
    else
        TempSal(i) = TempOperativa(i);
    end
```

```
end

%Calculamos el consumo, teniendo en cuenta que es 0 cuando SP = 0
for i=1:size(SP,1)
    if SP(i,1) == 0
        Con(i) = 0;
    else
        Con(i) = Con(i);
    end
end

%Si mi SP es superior a la temperatura operativa, la máquina no funciona
ya
%que no tiene sentido que enfrie, por lo que el consumo seria 0.
for i=1:size(Con,1)
    if TempSal(i) < SP_ON(i)
        Con(i) = 0;
    else
        Con(i) = Con(i);
    end
end

Con = Con(steptime:B,:);
TOperativa = TOperativa(steptime:B,:);
TempSal = TempSal(steptime:B,:);

%Sacamos el sumatorio de consumos para las t+10 siguientes horas
ConsumoMedio = sum(Con);

%se mete el penalti si la temperatura de salida sobrepasa los límites
for i=1:size(TempSal,1)
    if       TempSal(i) < 19
        ConsumoMedio=ConsumoMedio+penalti;
    elseif   TempSal(i) > 23
        ConsumoMedio=ConsumoMedio+penalti;
    end
end

%se premia el consumo cuando el primer valor del setpoint es un 0
if SPPred(1) == 0
    ConsumoMedio=ConsumoMedio + prize;
end

%se guardan las variables anteriores para la siguiente hora de simulación
TOperativaAct = TOperativa(1);
end
```

ANEXO IX.4

```
%Cargamos los parámetros guardados
load('FDT.mat');
s=tf('s');

%Construimos las G
g11=Kp11/(1+Tp11*s);
g11.iodelay=Td11;

g12=Kp12/(1+Tp12*s);
```

```
g12.iodelay=Td12;

g13=Kp13/(1+Tp13*s);
g13.iodelay=Td13;

g14=Kp14*(1+Tz14*s)/(1+Tp14*s);

g21=Kp21*(1+Tz21*s)/(1+Tp21*s);

g22=Kp22/(1+Tp22*s);
g22.iodelay=Td22;

g23=Kp23*(1+Tz23*s)/(1+Tp23*s);

g24=Kp24*(1+Tz24*s)/(1+Tp24*s);

%Cargamos las predicciones
load('Radiacion.mat');
Radiacion_total = data.result(1:168,:);
load('TempAmbiente.mat');
Temp_total = data.result(1:168,:);

Temp = Temp_total(1:steptime+10);
Radiacion = Radiacion_total(1:steptime+10);

t1=1:1:size(Temp_total);

%Creamos el espacio de estados
[a11,b11,c11,d11]=tf2ss(g11.num{1},g11.den{1});
[a12,b12,c12,d12]=tf2ss(g12.num{1},g12.den{1});
[a13,b13,c13,d13]=tf2ss(g13.num{1},g13.den{1});
[a14,b14,c14,d14]=tf2ss(g14.num{1},g14.den{1});
[a21,b21,c21,d21]=tf2ss(g21.num{1},g21.den{1});
[a22,b22,c22,d22]=tf2ss(g22.num{1},g22.den{1});
[a23,b23,c23,d23]=tf2ss(g23.num{1},g23.den{1});
[a24,b24,c24,d24]=tf2ss(g24.num{1},g24.den{1});

%Añadir solo si te utiliza Lsim
sys11=ss(a11,b11,c11,d11);
sys12=ss(a12,b12,c12,d12);
sys13=ss(a13,b13,c13,d13);
sys14=ss(a14,b14,c14,d14);
sys21=ss(a21,b21,c21,d21);
sys22=ss(a22,b22,c22,d22);
sys23=ss(a23,b23,c23,d23);
sys24=ss(a24,b24,c24,d24);

%Les dejamos el valor medio en 0
Radiacion_mean=detrend(Radiacion,0);
Temp_mean=detrend(Temp,0);
Radiacion_mean_total=detrend(Radiacion_total,0);
Temp_mean_total=detrend(Temp_total,0);

%Offsets de Temperatura y Pto funcionamiento
load('TOperativaAnt.mat')
TSalida=data.result(1:30);
Offset=TSalida(24);
val=detrend(TSalida,0);
valinicial=val(30);          %Recordar cambiar por steptime+24
```

```
%Simulación previa para saber donde estamos y calcular la correccion
%necesaria
[y11]=lsim(sys11,Radiacion_mean_total,t1,0);
[y12]=lsim(sys12,Temp_mean_total,t1,valinicial/sys12.c);

TOperativaRef=(y11+y12)+Offset-valinicial;

%Offsets de Consumo

load('ConsumoAnt.mat')
ConsumoSalida = data.result(1:30)/3600000;
OffsetCon=ConsumoSalida(24);
valCon=detrend(ConsumoSalida,0);
valinicialCon=valCon(30);   %Recordar cambiar por steptime+24
```