

Capítulo 6

Descripción del modelo completo y componentes del captador solar

En este capítulo se hablará sobre el proceso seguido para la obtención del modelo completo en EcosimPro[®] del captador solar basado en campo de espejos de Fresnel que forma parte de la planta solar para producción de frío situada en la azotea del edificio de la Escuela Superior de Ingenieros de Sevilla. También se realizará una descripción en profundidad de cada uno de los puertos y componentes que integran el modelo describiendo su código y funcionalidad.

Todos los componentes, puertos y funciones creados en EcosimPro[®] para generar el modelo del captador solar han sido agrupados en una librería con el nombre de PLANTA_SOLAR creada específicamente para contener todos los elementos necesarios para una planta de este tipo y que pueden ir siendo añadidos en un futuro.

6.1. Modelo completo del captador solar

En primer lugar vamos a describir cómo se ha obtenido el modelo completo del captador solar en EcosimPro[®] partiendo del modelo dado en Matlab y también hablaremos de los problemas a los que se ha tenido que hacer frente a la hora de abordar esta transición.

Abordaremos el diseño del modelo explicando tres puntos esenciales que se han de tener en cuenta para la elaboración del mismo, describiendo el resultado o conclusiones que se han obtenido al considerar cada uno de ellos:

- Modularidad del sistema
- Diseño de la comunicación de los módulos
- Condiciones de contorno

6.1.1. Modularidad del sistema

Como ya se ha comentado en el capítulo 3, **el lenguaje de programación de EcosimPro[®] es un lenguaje orientado a objeto**. Cada uno de estos objetos encapsula una serie de ecuaciones algebraicas y/o diferenciales que describen su comportamiento intrínseco. Atendiendo a esto último, nos encontramos ante **el primer paso importante** a realizar para la obtención del modelo en EcosimPro[®]: **diseñar y definir la modularidad del sistema, es decir, su división en diversos objetos o componentes**.

El **modelo en Matlab** del captador presenta un conjunto **definido de variables, constantes y funciones que se ejecutan siguiendo un determinado orden de manera**

secuencial, es decir, no se aprecia una clara división funcional en módulos. Se trata pues de un todo que se ejecuta según un determinado orden definido.

La orientación hacia objeto que presenta **EcosimPro**[®] nos obliga precisamente a acometer esa **división del modelo en diferentes componentes que funcionarán de forma independiente pero que a su vez estarán interrelacionados entre sí compartiendo información unos con otros**.

En resumen, el proceso de modularidad que se llevará a cabo consiste en dividir el modelo completo del captador solar desarrollado en Matlab y tratado como un todo, en diferentes objetos siguiendo un determinado criterio de manera que el funcionamiento conjunto y coordinado de todos ellos modele de la misma manera el funcionamiento del captador. Esquemáticamente, este proceso puede explicarse de la siguiente manera:

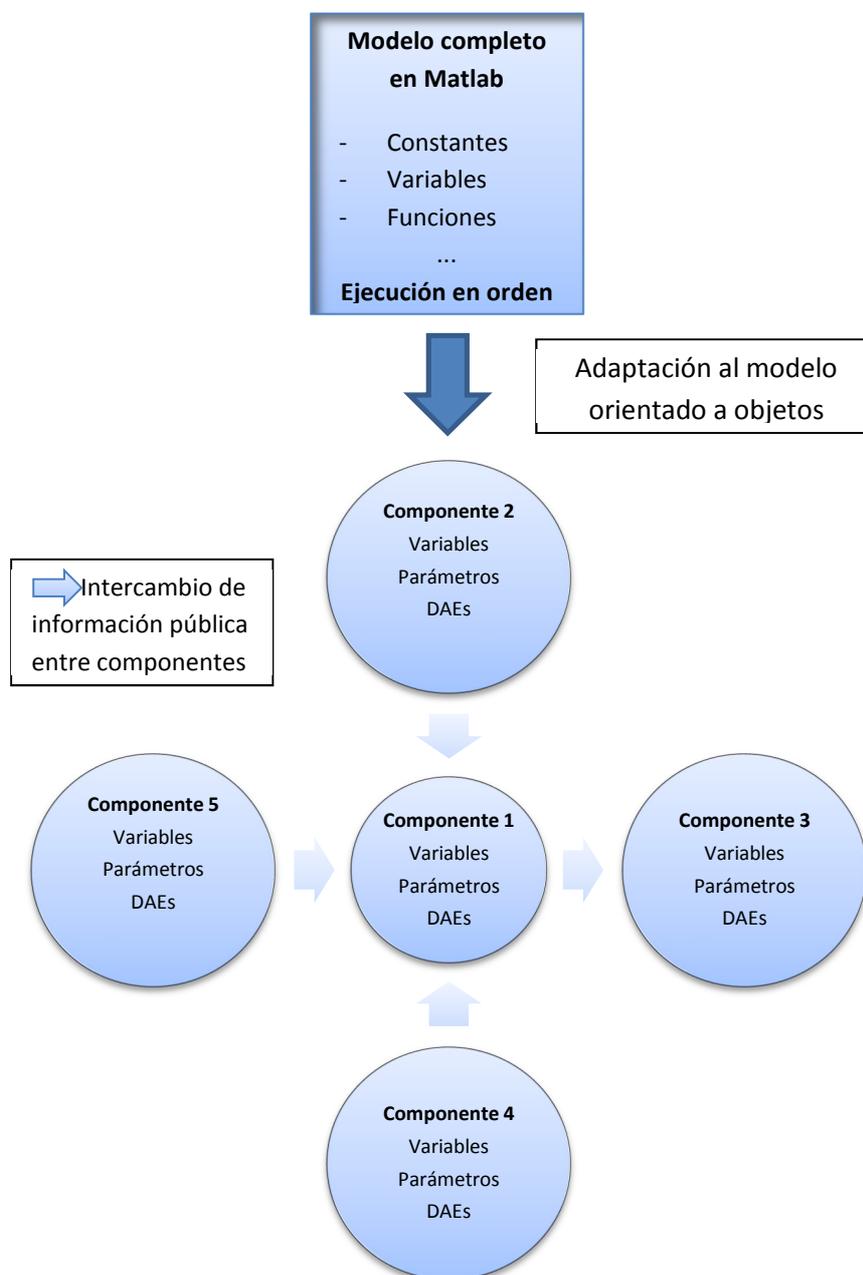


Ilustración 26. Proceso de adaptación al modelo orientado a objetos

Como se observa, cada componente contiene un conjunto de variables, parámetros y DAEs (ecuaciones algebraico-diferenciales) que definen su comportamiento intrínseco. Todos los elementos anteriores forman parte de la información privada de cada componente y por tanto, no se puede acceder a ella ni ser modificada desde cualquier otro componente exterior.

Por otro lado, la flechas que van de un componente a otro reflejan el intercambio de información inter-componentes necesario para el funcionamiento del modelo. Esa información pertenece al área pública de cada componente y por tanto puede ser accedida desde el exterior. Normalmente el contenido de la información pública está formado por valores de variables de entrada e información de parámetros calculados por un componente pero que son necesarios para que otro componente pueda desarrollar su funcionalidad con normalidad.

En nuestro caso, **el diseño de la modularidad del sistema** y su división en diferentes componentes se ha realizado **atendiendo al criterio físico del captador y de su entorno y a las variables que se manejan en el mismo**. De esta forma, las variables que maneja el sistema están relacionadas con el componente físico que las produce, es decir, cada componente genera las variables que físicamente están relacionadas con él.

Teniendo en cuenta el sentido físico del sistema, se ha dividido el modelo inicial en Matlab en 6 componentes que son los siguientes:

- Configuración_TUB
- Espejos
- Sol
- TUB_CAPTADOR
- Salida_Flujo
- Entrada_Flujo

Todos estos componentes calculan y producen una serie de variables que están relacionadas con su sentido físico. Por ejemplo, el componente *Sol* producirá y manejará las variables *Irradiancia* y *Temperatura_ambiente*, el componente *Espejos* las variables *Porcentaje_Espejos* y *Factor_de_Sombra* de los espejos, etc. Es decir, **cada componente representa el elemento físico con el que está relacionado**.

La principal **ventaja** que aporta esta división en diferentes componentes del sistema completo es la **posibilidad de reutilización** de los mismos en otros sistemas diferentes pertenecientes a otros campos, pudiendo adaptarlos a una nueva funcionalidad. Es decir, a partir de aquí cada componente funciona como un elemento independiente y puede ser cogido de la librería creada, adaptado y reutilizado para ser empleado en otro sistema diferente perteneciente a otro campo distinto.

Otra ventaja importante del diseño en módulos se produce a la hora de la **detección y depuración de errores en el sistema**, ya que de esta forma, se pueden ir probando

cada componente de manera independiente y ver si se produce o no un fallo en su funcionamiento antes de ensamblarlos y hacer pruebas en el sistema completo.

Una vez definidos estos componentes con toda su información pública y privada, aprovecharemos el principio de **encapsulación** comentado en el **capítulo 3** para definir un nuevo componente más complejo llamado *Campo_Solar*, que englobará a los 6 componentes anteriores y que reflejará el comportamiento completo del captador solar.

6.1.2. Comunicación de los módulos

Una vez diseñada la modularidad del sistema, nuestro **siguiente objetivo** es **ensamblar los componentes** citados anteriormente para tener un simulador del captador solar completo para la realización de experimentos sobre él y aprovechar así de este modo las ventajas que presenta EcosimPro[®] como lenguaje de simulación interactivo en tiempo real.

A la hora realizar el ensamblaje de los componentes del modelo nos encontramos con el siguiente aspecto de diseño a tener en cuenta: la **comunicación de unos objetos con otros**. Este problema se resuelve en EcosimPro[®] mediante la implementación de *Puertos*.

Los puertos pueden ser, como cabría esperar, de entrada o de salida. Cada puerto lleva asociado un número de variables que transporta y un comportamiento intrínseco de las mismas que se ha de programar. De aquí se desprende que si estas variables se las ha de comunicar a otro objeto, este segundo objeto ha de tener también un puerto cuyas variables sean las mismas que las que se le quieren transmitir, es decir, tiene que tener un puerto del mismo tipo.

Según lo comentado en el párrafo anterior, se empieza a entrever un **aspecto fundamental que el programador ha de tener en cuenta en el momento de diseñar los puertos: la compatibilidad y robustez en la definición de los puertos**. Esta definición va a resultar clave en un futuro ante la posibilidad de ir añadiendo diferentes módulos a un modelo, ya que de la correcta elección de las variables a transmitir y de su comportamiento intrínseco se determinará la consistencia y estabilidad del modelo final realizado. Esta característica hace que el programador no sólo se tenga que centrar en la implementación del componente que esté llevando a cabo en ese mismo instante, sino que ha de tener en mente que dicho componente forma parte de un modelo mucho más complejo. Por tanto es importante poner especial énfasis en disponer las ecuaciones y los puertos de tal manera que al integrarlo todo en un sistema más complejo compuesto por varios objetos no se desestabilice el comportamiento global del modelo.

Teniendo en cuenta la reflexión realizada en el párrafo anterior, se han creado un total de **4 tipos diferentes de puertos** que resuelven las tareas de transferencia de datos entre los componentes del modelo:

- Cond_espejos
- Fluido
- Parámetros_TUB
- Solar

Cada uno de estos puertos realizará una comunicación de variables públicas desde unos componentes a otros quedando de esta forma ensamblado el modelo final del captador solar.

Al igual que se ha realizado con los componentes, los puertos creados están íntimamente relacionados con las variables físicas reales que transportan, es decir, existe una concordancia entre el modelo y el elemento real que representa. Así por ejemplo, el puerto *Solar* transporta las variables *Temperatura ambiente e Irradiancia* relacionadas directamente con el componente *Sol* y la función que éste desempeña en el modelo.

Los puertos vienen representados gráficamente en EcosimPro[®] mediante pequeños círculos que se diferencian en el color del relleno y que permite distinguir a unos puertos de otros. Estos círculos **son puntos de conectividad a partir de los cuales se realizan todas las conexiones del modelo.**

A continuación se presenta una tabla en la que se especifican cada uno de los puertos empleados en el modelo, las variables que transportan y su correspondiente representación gráfica:

Puertos	Variables transportadas	Elemento gráfico
Cond_espejos	Factor de sombra Porcentaje espejos enfocados	
Fluido	Temperatura del fluido Caudal	
Parámetros_TUB	Diversos parámetros relacionados con la tubería del captador como: eficiencia media, diámetro y áreas, características termodinámicas del metal, etc.	
Solar	Temperatura ambiente Irradiancia	

Tabla 6. Puertos del modelo

6.1.3. Condiciones de Contorno

Otro aspecto al que hay que prestar especial atención en la programación, es la **elección de las variables de contorno del sistema, las *Boundary's***. Vamos a explicar qué sentido tiene la elección de estas variables durante la programación y el porqué de su importancia mediante la exposición de ejemplo.

Imaginemos que queremos realizar la simulación en EcosimPro[®] de un modelo que representa un depósito de agua con una bomba y una resistencia para calentar el agua. La variable de salida de nuestro sistema será un caudal de agua que se extrae del depósito a una temperatura dada. Por otro lado, lo normal es que dispongamos como variables de actuación, es decir, como variables de entrada globales a nuestro sistema de la regulación de la bomba (incremento o disminución de caudal) y del apagado o encendido de la resistencia.

Al programar este sistema en EcosimPro[®], como la programación es modular, vamos creando cada uno de los componentes con sus respectivos puertos para transportar los valores de las variables de un componente a otro, sus variables intrínsecas y su comportamiento descrito a base de ecuaciones que ligan todas estas variables. Pues bien, **al realizar la programación se ha de tener especial cuidado al escribir las ecuaciones, pues dependiendo de cómo se haga esto, al integrar los distintos componentes, puede que obtengamos como variables de actuación unas que no sean las deseadas** cuando realmente queríamos que las *Boundary* fuesen la regulación de la bomba y la señal de encendido o apagado de la resistencia.

Pongamos un ejemplo más simple pero más concreto que escenifica lo que acabamos de explicar. Supongamos que definimos un puerto en EcosimPro[®] llamado *puerto* que transporta una variable de tipo REAL llamada *num*. Su código es el siguiente:

```
PORT puerto SINGLE IN
      EQUAL OUT REAL num
END PORT
```

También tenemos un componente llamado *objeto1* cuya programación en EL sería:

```
COMPONENT objeto1
      IN puerto OB_IN
      OUT puerto OB_OUT
CONTINUOUS
      OB_OUT.num = sin (TIME) + OB_IN.num
END COMPONENT
```

Este componente calcula el valor de la variable del puerto de salida *OB_OUT.num* como suma del seno del tiempo y el valor de *num* que le llega por el puerto de entrada *OB_IN*. Si experimentásemos este componente de forma aislada, la variable *OB_IN.num* no está fijada, y al realizar una partición para simular un experimento, EcosimPro[®] nos

indicaría que para resolver el problema matemático necesita fijar una *Boundary*, y nos sugeriría como tal la variable *OB_IN.num*.

Hay que **resaltar la importancia de cómo escribir las ecuaciones ya que en muchos casos dependiendo de esto, EcosimPro[®] nos sugerirá unas *Boundary's* u otras**. Por ejemplo, imaginemos que en *objeto1* queremos asignar a *OB_OUT.num* el valor que le llega por *OB_IN.num*. En el bloque *CONTINUOUS* escribiríamos:

$$OB_OUT.num = OB_IN.num$$

Si escribimos la ecuación de esta manera, al realizar la partición EcosimPro[®] nos sugerirá como *Boundary* *OB_OUT.num*, cuando realmente lo que nos interesa es que la variable que actúe como *Boundary* sea *OB_IN.num*. Si invertimos la escritura de la ecuación como sigue, el problema quedaría resuelto.

$$OB_IN.num = OB_OUT.num$$

Cabe resaltar que EcosimPro[®], a la hora de realizar una partición nos sugiere un conjunto de variables para que éstas funcionen como *Boundary's*. Sin embargo, el programador siempre puede elegir de entre la lista de posibles candidatas y no tiene porqué coincidir con la propuesta por EcosimPro[®].

Se podría pensar que el hecho de que el programa nos la sugiera o no, no tiene relevancia alguna, pero esto no es así. El acierto de que **EcosimPro[®] nos sugiera como *Boundary* las variables que nosotros inicialmente diseñamos para que lo fueran** es de vital importancia por tres razones principalmente:

- 1- A la hora de conectar más componentes ocurre una “*cancelación de Boundary's*”, y precisamente elimina aquellas que el programa sugiere. Esto es de una importancia esencial a la hora de programar modelos complejos.
- 2- Si EcosimPro[®] ha sugerido una variable como *Boundary*, de forma interna, al resolver los sistemas de ecuaciones, va a hacerle un tratamiento diferente si es *Boundary*.
- 3- Nos va a resultar mucho más fácil realizar particiones si todas las variables sugeridas como *Boundary* son las que nosotros diseñamos, en vez de tener que ir mirando toda la lista de “posibles candidatas”, sobre todo si tenemos un modelo con muchas variables.

Para un mayor comprensión, vamos a ejemplificar la cancelación de *Boundary's*. Para ello, vamos a unir dos objetos y en función de la entrada que le demos al *Objeto2*, queremos ver como varía la salida del *Objeto1*, es decir, nuestra intención es que la *Boundary* sea la entrada del *Objeto2*.

Realizamos otra instancia de un componente llamado “Objeto2”, cuyas ecuaciones son:

```
COMPONENT objeto2
PORTS
    IN puerto OB_IN
    OUT puerto OB_OUT
CONTINUOUS
    OB_IN.num = OB_OUT.num
    --La boundary de objeto2 seria OB_IN.num
END COMPONENT
```

EcosimPro[®], tal como hemos comentado anteriormente, nos sugeriría como *Boundary OB_IN.num*, tal y como nosotros queremos.

Lo siguiente es unir estos dos componentes conectando la salida del objeto2 “*objeto2.OB_OUT.num*” a la entrada del objeto1 “*objeto1.OB_IN.num*”. Efectivamente, al realizar una partición sobre este componente compuesto, nos va a sugerir como Boundary la variable “*objeto2.OB_IN.num*”, es decir, como nosotros habíamos diseñado. EcosimPro[®] ha cancelado de forma automática la Boundary inicial del Objeto1, “*objeto1.OB_IN.num*”. En este sentido, EcosimPro[®] esta muy bien programado. Todo descansa entonces en el cuidado que ponga el programador a la hora de tratar las ecuaciones en los componentes, para que después al unir muchos componentes se cancelen muchas Boundary, y solo queden al final las que realmente deberían serlo.

Esto se ha de tener muy en cuenta, puesto que una buena programación, facilita mucho la puesta en marcha de un componente complejo y dota a este componente de una futura ampliación mucho más fructífera.

Una vez realizadas todas las consideraciones anteriores, en lo que concierne a nuestro modelo concreto sobre el captador solar, **las condiciones de contorno en el mismo vendrán impuestas por las variables que determinan e influyen en su comportamiento**. De este modo, tenemos que las **boundary’s que actúan sobre el sistema** son las siguientes:

- Caudal de entrada
- Temperatura del fluido de entrada
- Irradiancia
- Temperatura del ambiente
- Porcentaje de espejos enfocados
- Factor de sombra del campo de espejos

Todas estas variables determinan el funcionamiento del captador y por tanto la **variable de salida** que nos interesa conocer y que es **la temperatura del fluido cuando sale del sistema**.

Si experimentásemos el componente *TUB_CAPTADOR* de forma aislada, al realizar la partición del mismo empleando el asistente que proporciona EcosimPro[®], llegaríamos a la ventana “BOUNDARIES WIZARD”, donde el programa nos anuncia que en el modelo matemático del componente se han detectado más variables que ecuaciones y que por tanto, para continuar es necesario seleccionar un conjunto de boundaries de entre la lista de variables propuesta. La imagen del asistente es la siguiente:

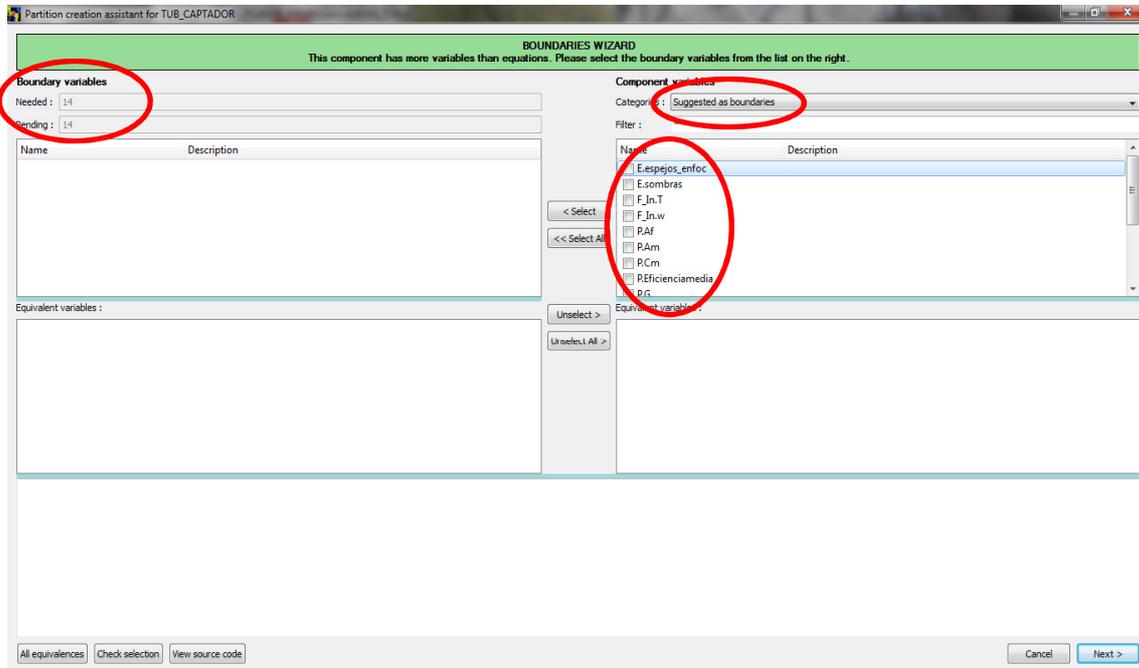


Ilustración 27. Asistente para la selección de boundaries

Si observamos la ventana, en la parte superior izquierda del asistente, el programa nos indica el número necesario de boundaries que es preciso definir para validar la partición del componente *TUB_CAPTADOR*. En nuestro caso, se requieren un total de 14 variables de tipo boundaries. Por otro lado, si nos fijamos en la zona de la derecha, EcosimPro[®] nos sugiere aquellas variables que, teniendo en cuenta el diseño realizado en la programación del componente, podrían o deberían funcionar como boundaries.

Si todo ha sido realizado correctamente, en el cuadro de variables boundaries propuestas, deberían figurar aquellas que nosotros hemos diseñado previamente como boundaries en nuestro modelo, es decir, caudal de entrada, temperatura del fluido de entrada, irradiancia, temperatura ambiente, porcentaje de espejos enfocados y factor de sombra del campo de espejos. Pues bien, efectivamente estas variables aparecen como variables sugeridas por EcosimPro[®] como boundaries, y por **tanto concluimos que las ecuaciones que modelan el componente *TUB_CAPTADOR* han sido escritas y ordenadas de la forma correcta dando validez a su programación.**

El resto de variables hasta llegar a las 14 propuestas por EcosimPro[®], son parámetros del componente *TUB_CAPTADOR* que han de ser fijados en el modelo tras la realización de una serie de cálculos previos. Estos parámetros podrían haber sido introducidos de forma directa en el componente de forma que no aparecieran como

boundaries sino como parámetros del mismo, pero se ha optado por crear un nuevo componente *Configuración_TUB*, que realice esos cálculos previos y luego pase los parámetros calculados al componente TUB_CAPTADOR, apareciendo de esta manera en este último como boundaries y dejando únicamente como parámetros directos del mismo los concernientes a la discretización de la tubería del captador, es decir, el número de elementos en los que se divide y la longitud de los mismos.

Las razones para la elección de esta última opción en el cálculo de los parámetros de la tubería del captador son dos:

- 1- Se descarga al componente principal TUB_CAPTADOR de nuevas ecuaciones y cálculos previos que concentrarían en el mismo demasiada carga de código. De esta forma en el componente TUB_CAPTADOR únicamente se incluyen las DAEs del modelo de parámetros distribuidos y otras funciones que calculan parámetros dinámicos del sistema consiguiendo de esta manera una programación más distribuida y sencilla de comprender.
- 2- Se consigue un modelado más genérico del sistema mediante dos componentes diferentes: uno que simula una tubería genérica con sus ecuaciones de funcionamiento y otro que la particulariza mediante la selección y fijación de diversos parámetros como por ejemplo los diámetros interior y exterior de la tubería, factores de ensuciamientos, reflectividad y transmisividad, características de metal, etc.

Por tanto concluimos que **en nuestro modelo existen una serie de condiciones de contorno que es preciso fijar para poder así simular el comportamiento del captador**. Estas condiciones podrían ser introducidas de forma manual a través de diversas funciones, pero como hemos comentado ya con anterioridad, se emplearán los datos tomados de la planta para recrear unas condiciones completamente reales que nos ayuden a validar el modelo.

6.1.4. Componente final: *Campo_Solar*

Una vez realizada la división del modelo en los diferentes componentes funcionales, la definición de los puertos de conexión y la descripción de las variables de contorno se procede a realizar el ensamblado de los distintos elementos para obtener el modelo que simula el funcionamiento completo del captador.

El modelo final vendrá dado por el componente *Campo_Solar*, el cual será creado empleando el principio de **encapsulación** instanciando los diferentes componentes que integran el sistema y uniéndolos entre sí.

Para la creación de este componente se empleará la interfaz gráfica que proporciona EcosimPro[®], ya que nos permite generar de una manera visual y extremadamente sencilla el nuevo componente. Para ello primeramente generamos los símbolos de los

distintos componentes empleando la herramienta de dibujo correspondiente integrada en el programa. Seguidamente, se crea un nuevo esquema llamado *Campo_Solar* y arrastramos los componentes necesarios desde la ventana de símbolos hasta la ventana del esquema generado. Finalmente se realiza la unión de los componentes empleando los puertos correspondientes definidos previamente y se compila el esquema resultante. En la siguiente figura podemos ver el esquema generado correspondiente al componente *Campo_Solar*:

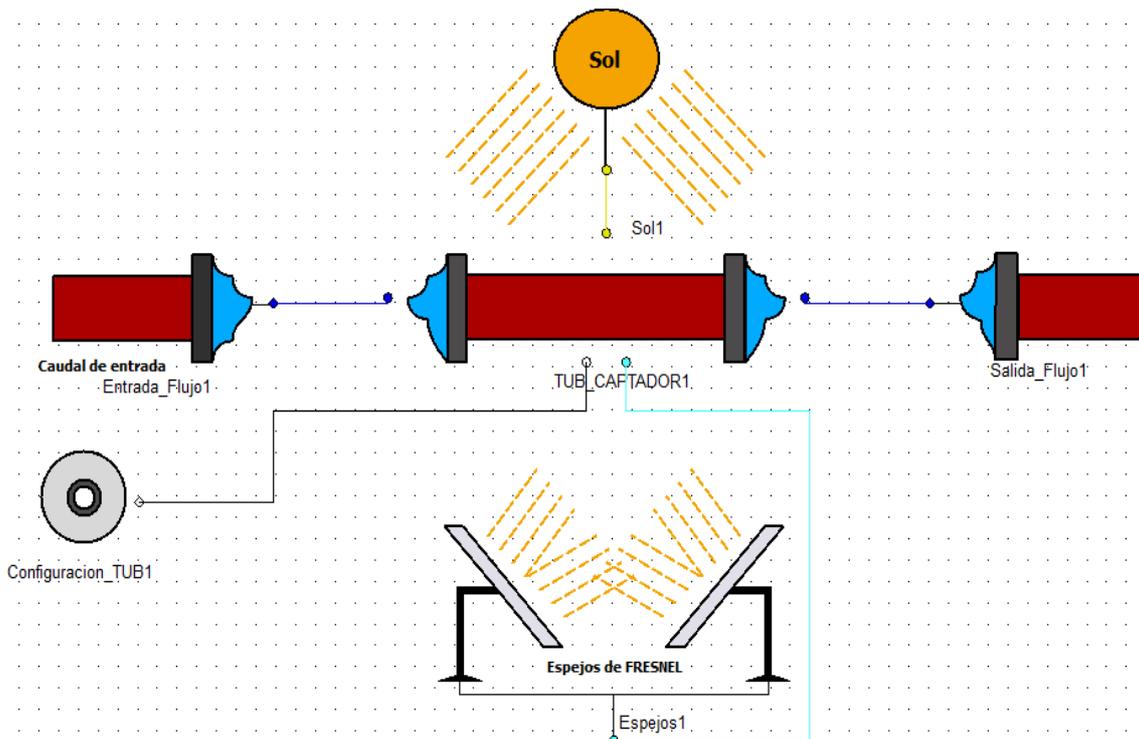


Ilustración 28. Esquema componente Campo_Solar

Como vemos en la imagen, tenemos un componente principal TUB_CAPTADOR que representa físicamente al captador solar de la planta. Podemos decir que se trata de un tramo de tubería con ciertas características especiales. A su alrededor, conectados a él mediante sus respectivos puertos, se encuentran el resto de componentes que se utilizan para fijar las condiciones de funcionamiento, configurar el captador y mostrar los resultados.

Atendiendo a la modularización realizada en el apartado 6.1.1 y que se observa en la ilustración 28, podemos dividir los componentes de nuestro modelo en tres grupos según la funcionalidad que desempeñen:

- **Componentes que fijan las condiciones de contorno y funcionamiento del sistema:** en este grupo se encuentran los componentes *Sol*, *Espejos* y *Entrada_Flujo*. Estos componentes leen los datos reales tomados de la planta por el sistema de adquisición de datos de la misma, realizan las transformaciones necesarias sobre ellos y calculan finalmente las boundary's que transmitirán al componente principal que simula el captador solar a través de los puertos correspondientes.

- **Componente captador:** este grupo está formado únicamente por el componente *TUB_CAPTADOR*. Este componente simula el comportamiento real del tramo de tubería que atraviesa el campo de espejos de Fresnel en el que se reflejan los rayos solares para sobrecalentar el fluido y que tiene una longitud total de 64 metros. Es el elemento más importante del sistema y en él se concentran todas las ecuaciones algebraico-diferenciales descritas en el capítulo 5 que describen su comportamiento. Podemos decir que es el elemento central del modelo y que representa físicamente al captador de la planta. Recibirá a través de los puertos correspondientes las condiciones de contorno necesarias para determinar su funcionamiento y que vendrán impuestas por los componentes pertenecientes al primer grupo.
- **Componentes de configuración y obtención de resultados:** en este grupo se hallan los componentes *Salida_Fluido* y *Configuración_TUB*. El primero de los dos recoge por un lado la salida del componente principal (temperatura del fluido a la salida del captador según el modelo) y por otro, la salida real del captador dada por el sistema de adquisición de datos de la planta, para después realizar una comparación entre ambas. También realiza un cálculo del error cometido por el modelo empleando el criterio de la integral del error ITAE, útil para el posterior diseño de posibles controladores para el captador. El segundo componente *Configuración_TUB* realiza una serie de cálculos previos para determinar una serie de parámetros de configuración de la tubería del captador que posteriormente son pasados a este componente a través de un puerto específicamente diseñado para ello.

Cada componente de los citados anteriormente posee los puertos que necesita para conectarse con el exterior y transportar sus datos públicos. Los puertos y las conexiones que realizan se describen a continuación:

- **Puerto Fluido:** podemos decir que es el puerto que conecta los distintos tramos de tubería de la planta. Transporta variables relacionadas físicamente con el fluido que circula por el circuito de la planta: *caudal* y *Temperatura del fluido*. Conecta al componente *TUB_CAPTADOR* con los componente de entrada y salida al mismo: *Entrada_Flujo* y *Salida_Flujo*.
- **Puerto Solar:** puerto creado específicamente para la conexión del componente *Sol* con el componente principal *TUB_CAPTADOR*, para transportar las variables *Irradiancia* y *Temperatura ambiente* relacionadas con el efecto que produce el sol sobre la tubería del captador.
- **Puerto Cond_espejos:** puerto específico creado para conectar el campo de espejos de Fresnel (componente *Espejos*) con el componente central *TUB_CAPTADOR*. Transporta las variables relacionadas con el campo de espejos de la planta que afectan al calentamiento de la tubería del captador: *Porcentaje de espejos enfocados* y *Factor de sombra* del campo de espejos de Fresnel.

- **Puerto Parámetros_TUB:** puerto que transporta los parámetros relacionados con la configuración de la tubería del captador desde el componente que los genera *Configuración_TUB*, hasta el componente central TUB_CAPTADOR que representa a la tubería propiamente dicha. Los parámetros que transporta son: *eficiencia media, diámetro propio de la tubería, áreas transversales del fluido y metal, apertura del colector, características del metal y tiempo de integración.*

Al realizar la unión de todos los elementos comentados anteriormente mediante los puertos descritos, se producirá la **cancelación de boundaries** que buscábamos comentada en el apartado 6.1.3, **dando como resultado un modelo matemático completamente cerrado y fijado a través de los componentes que generan las condiciones de contorno y funcionamiento reales necesarias para recrear el funcionamiento del captador.** Por ejemplo, como ya hemos visto al instanciar el componente TUB_CAPTADOR, el programa detectaba como condiciones de contorno la *temperatura y el caudal de entrada* al mismo. Pues bien, al realizar la unión de este componente con el componente *Entrada_Flujo* a través de los puertos tipo *Fluido* de entrada y de salida que posee cada uno como se observa en la siguiente figura, se produce la cancelación de las variables boundaries iniciales *TUB_CAPTADOR.F_In.T* y *TUB_CAPTADOR.F_In.w* que representan la temperatura y caudal de entrada respectivamente.

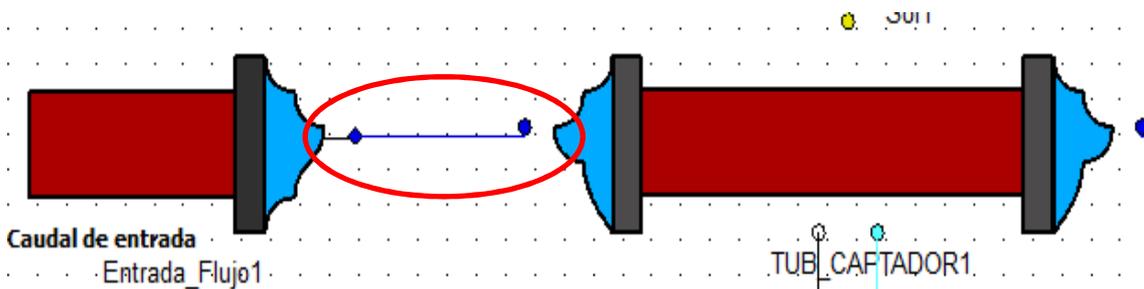


Ilustración 29. Conexión de puertos y cancelación de Boundaries

El proceso de cancelación de boundaries se repite en cada una de las conexiones de puertos existentes en el modelo hasta producirse una cancelación total de las mismas. Resaltamos de nuevo la importancia en el diseño del modelo a través de sus puertos y componentes para que EcosimPro[®] nos proponga como boundaries aquellas que nosotros inicialmente hemos diseñado para tal efecto. En nuestro caso, durante el proceso de programación se ha tenido que modificar en varias ocasiones el orden de las ecuaciones en diferentes componentes para que al final el resultado fuera el esperado, pero se insiste en que, aunque sea una tarea ardua, es necesario hacerla para evitar futuros problemas con el modelo en nuevas ampliaciones.

Otro aspecto es la asignación de valores a las boundaries. Una vez que el programa nos ha dado como boundaries las variables adecuadas, podemos optar por fijarlas dando lugar a una partición cerrada o dejarlas libres. En nuestro caso se ha optado por fijarlas empleando datos reales obtenidos de la planta. Si se opta por la otra opción, entonces a

la hora de experimentar el modelo es necesario dar los valores a la boundaries antes de simular el experimento mediante constantes, funciones, etc.

En la siguiente figura se muestra la ventana del asistente para la creación de la partición final del componente *Campo_Solar*, donde se observa como no existen variables boundaries en el modelo debido a la cancelación de las mismas mediante la unión de los distintos componentes.

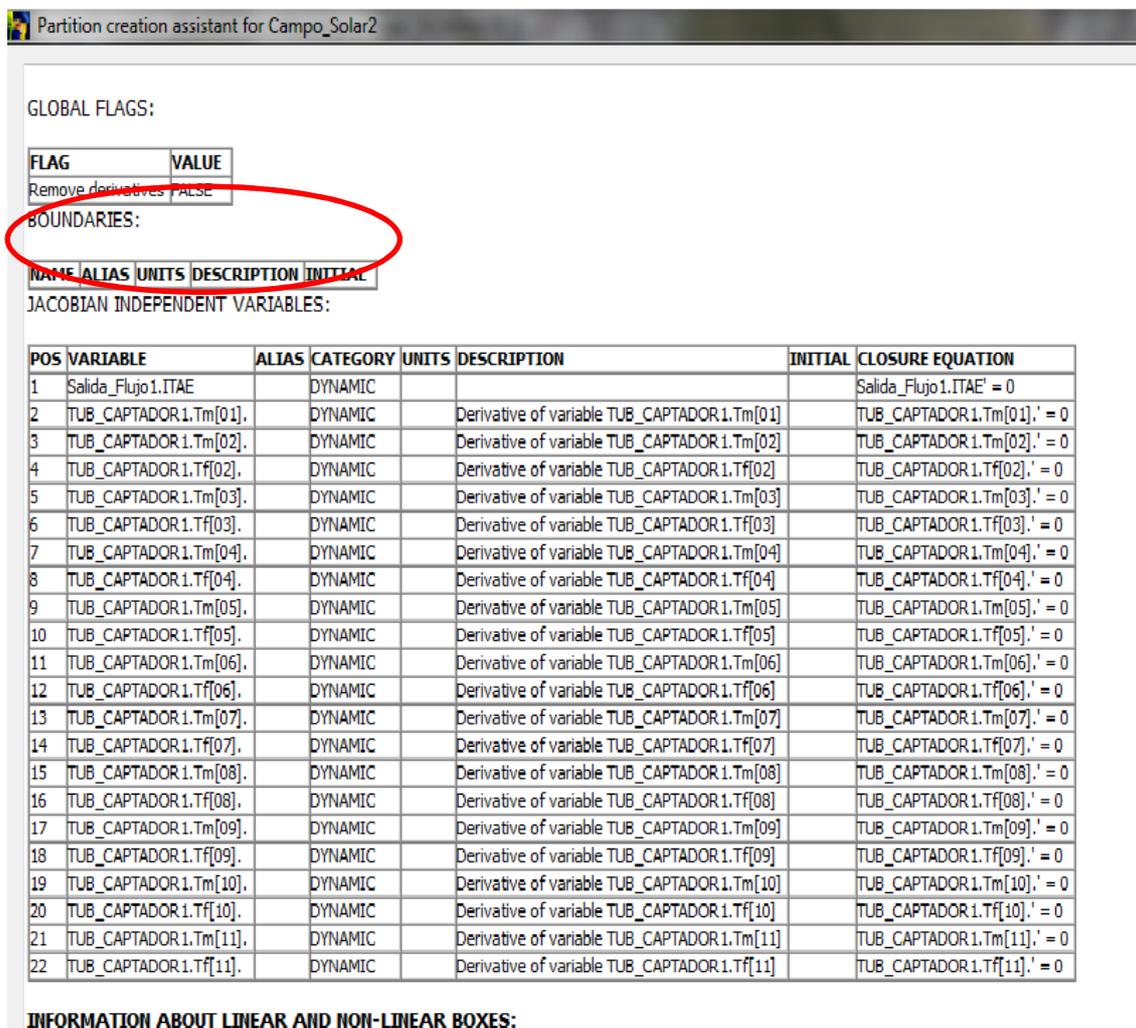


Ilustración 30. Resumen partición componente *Campo_Solar*

6.2. Programación de componentes y puertos

En este apartado se expondrán cada uno de los componentes y puertos que han sido programados en lenguaje EL de EcosimPro® para la obtención del modelo completo del captador. Se introducirá brevemente la función de cada uno de ellos en el modelo para después describir su código y posteriormente realizar una explicación detallada sobre el mismo destacando sus características más notorias de cara a su comportamiento y tratamiento de parámetros para alcanzar los objetivos deseados en su posterior simulación.

6.2.1. Puertos del modelo

El modelo del captador contará con una serie de puertos que definen la conexión de los componentes con el exterior (normalmente otro componente). Como ya se ha comentado en el capítulo 3, cada uno tiene asociado una clase (ej. Eléctrico) y una dirección IN (entrada) o OUT (salida).

Los puertos encapsulan las variables que representan el intercambio entre componentes. De esta forma en EcosimPro[®] se evita tener que conectar variable a variable como se realiza en otros simuladores como Simulink.

Antes de pasar a describir los distintos puertos del modelo, se van a explicar algunos aspectos importantes a tener en cuenta en la definición de los puertos de los que se ha hecho uso en este caso y que aparecerán posteriormente.

EcosimPro[®] introduce **inteligencia a la conexión de puertos empleando diferentes operadores que permiten generar automáticamente ecuaciones de conexión entre los puertos conectados**. Esto nos evita el tener que crear componentes intermedios más simples como divisores y uniones, lo que supone una notable ventaja y considerable ahorro de elementos. También se pueden introducir **restricciones en las conexiones de los puertos** tanto en el número de ellos conectados como en su sentido. Nosotros únicamente veremos aquí los tres que han sido empleados para la definición de los puertos de nuestro modelo, siendo muchos más los existentes y que permiten crear puertos de gran complejidad e inteligencia.

Los operadores empleados en la definición de los puertos del modelo son los siguientes:

- **SINGLE**: fuerza a que las conexiones de un puerto sean únicas. Con SINGLE IN se prohíbe la conexión múltiple con puertos de entrada. Análogamente, se usaría SINGLE OUT para el caso de limitación de conexión múltiple con puertos de salida. Un claro ejemplo de aplicación es en la simulación de control, donde una señal de control es aplicable a distintas salidas, pero no tiene sentido usar varias señales de control a una sola entrada.
- **EQUAL**: este operador fuerza a mantener el valor de una variable para todos los puertos de una conexión. Un ejemplo es el caso de la tensión en un puerto de tipo eléctrico.
- **SUM**: el operador SUM se asocia a variables de flujo e indica que la suma de los flujos entrantes debe ser igual a la de los salientes. Un ejemplo de nuevo para el caso eléctrico es el de la intensidad.

EcosimPro[®] generará de forma automática las ecuaciones de conexión entre los puertos conectados teniendo en cuenta las restricciones introducidas, lo cual resulta una ventaja significativa a la hora de facilitar el trabajo de futuros programadores en la introducción de nuevos componentes, pues sin saber con profundidad el funcionamiento

completo del modelo y sus puertos, se pueden realizar nuevas conexiones y EcosimPro® controlará que se cumplan las restricciones impuestas anteriormente sin que el usuario tenga que hacerse cargo de esta tarea.

Puerto Cond espejos

La función que realiza este puerto es la de transmitir entre componentes las variables que están relacionadas físicamente con las **condiciones que presentan los espejos en un instante concreto del día**. Concretamente transmite información sobre el porcentaje de espejos orientados hacia el captador y el factor de sombra que influye sobre el campo de espejos de Fresnel en un momento determinada del día. Para ello emplea las variables *espejos_enfoc* y *sombras*, las cuales reflejan respectivamente los dos datos expresados anteriormente sobre las condiciones de los espejos a lo largo del todo el día de prueba en cuestión. Ambas variables se miden en tanto por uno.

El código en EcosimPro® es el siguiente:

```
PORT cond_espejos SINGLE IN          -- Condiciones campo de espejos
    EQUAL OUT REAL sombras           -- Factor de sombra (0-1)
    EQUAL OUT REAL espejos_enfoc     -- % Espejos enfocados (0-1)
END PORT
```

Este puerto transportará entre los componentes *TUB_CAPTADOR* y *Espejos* las variables *espejos_enfoc* y *sombras* y se representa gráficamente en el esquema de conexiones por un círculo azul celeste.

Como se observa en el código, se ha empleado una **restricción en las conexiones para este puerto** en el caso de que éste se defina como entrada *IN*. Mediante *SINGLE* se fuerza a que las conexiones del puerto sean únicas y por tanto se está impidiendo las conexiones múltiples en la entrada del puerto tipo *cond_espejos*. Si nos paramos a pensar en el sentido de esta restricción, ésta resulta evidente ya que las condiciones de los espejos son las que son, no pueden existir diferentes condiciones para un día y un instante determinado en la planta y por tanto sólo tiene sentido que esas condiciones sean las mismas en la entrada y transmitidas por un solo puerto. Esta restricción impedirá futuros problemas en las conexiones de los puertos a la hora de ampliar el modelo con nuevos componentes de la planta que compartan información y variables con los ya existentes.

A parte de la restricción anterior en cuanto a las conexiones de este puerto, se introducen nuevas restricciones inteligentes para las variables que encapsula. Mediante el operador *EQUAL* forzamos a mantener el valor de una variable para todos los puertos de una conexión. En nuestro caso, se añade también el operador *OUT* para imponer que la acción de *EQUAL* únicamente se produzca cuando este tipo de puertos sean definidos como salida. En resumen, cuando el puerto *cond_espejos* sea definido como

salida, todas las conexiones que partan de dicho puerto deberán presentar los mismos valores en las variables *sombras* y *espejos_enfoc*.

Puerto Fluido

Este puerto se encarga de transmitir toda la información relacionada con el estado del fluido que circula por la planta. Para ello encapsula dos variables w y T que contienen información sobre el caudal y la temperatura del fluido respectivamente en cada instante de tiempo de funcionamiento de la planta. Concretamente transportará dicha información entre los componentes *Entrada_Flujo*, *TUB_CAPTADOR* y *Salida_Flujo*, el cual es el sentido natural del fluido en el captador. Se representa gráficamente por un círculo azul oscuro. La unidades de caudal y temperatura son m^3/h y $^{\circ}C$ respectivamente.

El código que presenta es el siguiente:

PORT fluido	-- Estado del fluido
SUM REAL w	-- Caudal másico (m/h)
EQUAL OUT REAL T	-- Temperatura del fluido ($^{\circ}C$)
END PORT	

Como se observa en el código, para este puerto ya **no se impone la restricción sobre múltiples conexiones en las entradas o salidas del mismo mediante el operador *SINGLE***. Lo anterior resulta bastante lógico si nos hacemos a la idea de que este puerto y sus conexiones son como si representaran el sistema de tuberías de la planta, es decir, una conexión entre dos componentes que compartan este puerto se puede ver en realidad como una tubería que transporta el fluido entre ambos. Por tanto no tiene sentido plantear conexiones simples en este puerto ya que impediríamos que existieran puntos de unión o difusión de flujo a lo largo del circuito de circulación del fluido por la planta. En la planta real en la que se basa este modelo existen válvulas que nos permiten hacer recircular el fluido que pasa por ellas en una dirección u otra o en ambas a la vez, por tanto es imprescindible no incluir el operador *SINGLE* en este tipo de puertos.

En cuanto a las variables w y T , la explicación sobre las restricciones que se aplican sobre ellas se entiende si pensamos en el sentido físico de las variables a las que representan. En el caso de w , ésta simboliza el caudal másico de fluido que atraviesa las tuberías del circuito y por tanto lo natural es que se le aplique el operador *SUM*, lo cual implica que al entrar o salir varios caudales másicos éstos se sumen imponiendo **igualdad en la suma de los caudales entrantes con la de los salientes**. Por otro lado, T representa la temperatura del fluido y empleando el operador *EQUAL OUT* estamos imponemos la restricción de que la **temperatura del fluido sea la misma para todos los caudales que salen de un punto de difusión**.

Las ideas expresadas en el párrafo anterior pueden resumirse en la siguiente ilustración en la que también se observan las ecuaciones de conexión generadas automáticamente por el programa al realizar conexiones múltiples de puertos imponiendo los operadores *SUM* y *EQUAL*.

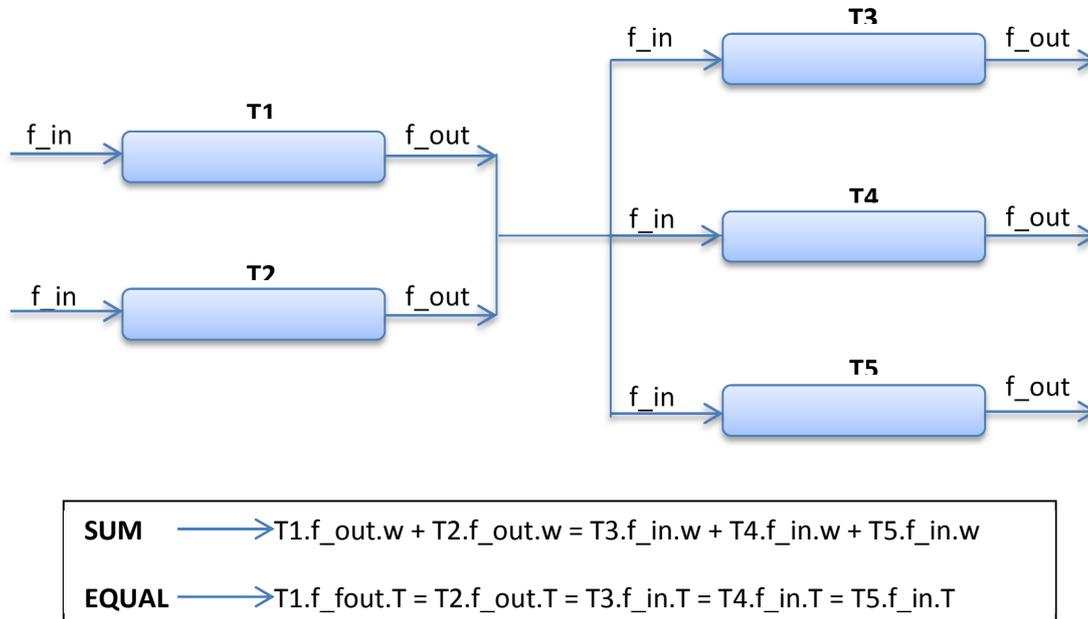


Ilustración 31. Ecuaciones de conexión generadas por los operadores *SUM* y *EQUAL* en una conexión múltiple

Puerto Parámetros TUB

Puerto que transporta información acerca de los parámetros configurables por el usuario que afectan al captador, como son: eficiencia óptica del campo de espejos, diámetro y áreas de la tubería del colector, apertura del mismo, características termodinámicas del metal y tiempo de integración. Todos los datos anteriores serán transportados por este puerto desde el componente *Configuración_TUB* desde donde son introducidos, hasta el componente principal *TUB_CAPTADOR*. Se representa gráficamente por un círculo blanco.

El código que presenta es el siguiente:

```

PORT parametros_TUB SINGLE IN
                                -- Características de la tubería

    EQUAL OUT REAL Eficiencia     -- Eficiencia óptica (0-1)
    EQUAL OUT REAL L              -- Diámetro propio de la tubería (m)
    EQUAL OUT REAL Af             -- Área transversal del fluido (m²)
    EQUAL OUT REAL Am            -- Área transversal del metal (m²)
    EQUAL OUT REAL G              -- Apertura del colector (m)
    EQUAL OUT REAL pm            -- Densidad del metal (Kg/m³)
    EQUAL OUT REAL Cm            -- Calor específico del metal (J/Kg °C)
    EQUAL OUT REAL Tint          -- Tiempo de integración (s)

END PORT
    
```

Nuevamente en este puerto se impone la restricción que evita las conexiones múltiples cuando se configura el mismo como entrada mediante *SINGLE IN*, impidiendo la llegada al puerto de más de una conexión y asegurándonos de esta manera que no haya posibles colisiones de datos de configuración en la entrada. Igualmente, a todas las variables que encapsula este puerto se le aplica *EQUAL* cuando el puerto es configurado de salida, implicando que todas las conexiones que partan de este puerto deberán contener el mismo valor de todos los parámetros de configuración descritos.

Puerto Solar

El puerto Solar transporta la información referente a las condiciones ambientales presentes en la planta a lo largo de todo el día de observación. Concretamente, las variables que nos interesan para el funcionamiento del colector son la irradiancia y la temperatura ambiente, que serán transmitidas mediante las variables de tipo *REAL I* y *Tamb* respectivamente. En el modelo estas variables serán transportadas desde el componente *Sol*, encargado de recoger esta información, hasta el componente principal *TUB_CAPTADOR* donde serán empleadas para realizar los cálculos de temperatura del fluido a la salida del colector. Se representa gráficamente mediante un círculo amarillo. Las unidades de irradiancia y caudal son W/m^2 y $^{\circ}C$ respectivamente.

Su código es como sigue:

```
PORT solar SINGLE IN          -- Condiciones solares
                                EQUAL OUT REAL I          -- Irradiancia (W/m2)
                                EQUAL OUT REAL Tamb       -- Temperatura ambiente (°C)
END PORT
```

Al igual que en puertos anteriores se aplican las condiciones de *SINGLE* para las conexiones y *EQUAL* para las variables cuando el puerto funciona como entrada y salida respectivamente, por tanto se limita a una única conexión de entrada en el puerto y las salidas del mismo han de tener idéntico valor para todas sus variables.

6.2.2. Componentes del modelo

Los componentes que forman el sistema son el elemento principal del mismo y nos proporcionan una ligera visión de cómo se ha realizado el proceso de modularización y cual es la función de cada uno de ellos dentro del modelo global al que pertenecen. Todos y cada uno de ellos representan o **están relacionados con algún elemento físico o parte del sistema real** y con las variables que dicho elemento maneja, comunicándose con el exterior a través de los puertos correspondientes. **Cada componente representa el comportamiento continuo, discreto y secuencial del elemento al que representa.** Iremos viendo los distintos componentes del modelo explicado en cada uno de ellos los bloques que integran su código.

Componente Sol

El componente *Sol* es el primero de los tres componentes que se encargarán de calcular y transmitir las variables de contorno que fijarán las condiciones de funcionamiento de la tubería del captador. En su caso, se encargará de leer las condiciones ambientales en las que se encuentra la planta en cada momento del día, concretamente la irradiancia y la temperatura ambiente, transmitiéndolas posteriormente al componente principal *TUB_CAPTADOR*. Para realizar el transporte de dicha información este componente dispone de un puerto del tipo *Solar* que será configurado como salida (OUT).

El símbolo y código del componente son los siguientes:

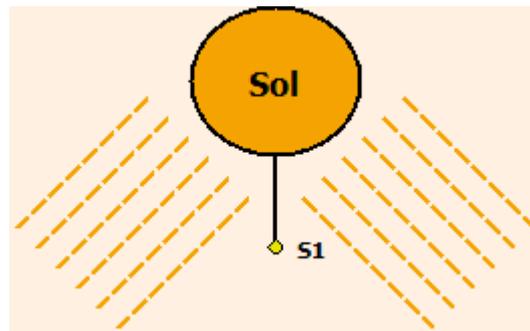


Ilustración 32. Símbolo componente Sol

```

COMPONENT Sol

PORTS
    OUT solar S1                    -- Puerto de salida tipo Solar
DATA
    FILEPATH datafile="@PLANTA_SOLAR@/experiments"
    -- Directorio de datos de la planta
DECLS
    REAL I                          -- Irradiancia
    REAL Tamb                       -- Temperatura ambiente
    -- Tablas unidimensionales para adquisición de datos
    TABLE_1D t_Tamb
    TABLE_1D t_irradiancia
    BOOLEAN correct_I               -- Corrector de irradiancia
INIT
    -- Lectura de datos
    readTableCols1D("datafile/DATA110921.txt",1,6,t_irradiancia)
    readTableCols1D("datafile/DATA110921.txt",1,3,t_Tamb)
DISCRETE
    WHEN (I<10) THEN correct_I= TRUE    -- Irradiancia menor que 10
    END WHEN
CONTINUOUS
    I= ZONE (correct_I== TRUE) 10      -- Corrección de irradiancia
    OTHERS linearInterp1D(t_irradiancia,TIME) -- Interpolación de datos
    Tamb = linearInterp1D(t_Tamb,TIME)
    Tamb=S1.Tamb                      -- Envío por el puerto de salida
    I=S1.I
END COMPONENT
    
```

Los diferentes bloques del componente son los siguientes:

PORTS

Este componente dispone de un único puerto tipo *Solar* llamado *SI* y configurado como salida para transmitir los valores de irradiancia y temperatura de la planta a lo largo de un día.

DATA

Este componente posee como variable tipo DATA, la variable *datafile* de tipo *FILEPATH*. Las variables *FILEPATH* en EcosimPro® son como las cadenas de caracteres y permiten llevar a cabo las mismas operaciones que con una variable tipo *STRING*. La única diferencia entre ambas es que si un tipo es definido como *FILEPATH*, cuando empleamos el editor de objeto se permite al usuario seleccionar y cambiar la ruta del archivo tanto relativa como absoluta.

Nosotros emplearemos la variable ***datafile* para indicar la ruta de archivo relativa donde estarán almacenados los datos reales tomados de la planta** que queremos emplear para realizar la simulación fijando las condiciones de contorno, en este caso: "*@PLANTA_SOLAR@/experiments*". El operador @ se emplea para indicar una ruta de archivo relativa. Posteriormente, cuando nos queramos referir a esa ruta, únicamente emplearemos *datafile* sin tener que escribir la dirección completa. Cuando queramos tomar otros datos que se encuentren en otro directorio distinto, simplemente cambiando el valor de esta variable con la nueva, se tomarán automáticamente los nuevos datos en los que estamos interesados.

La ilustración siguiente muestra la ventana de edición de atributos para este componente, en este caso un único atributo para establecer la ruta de archivo donde se encuentran los datos de la planta.

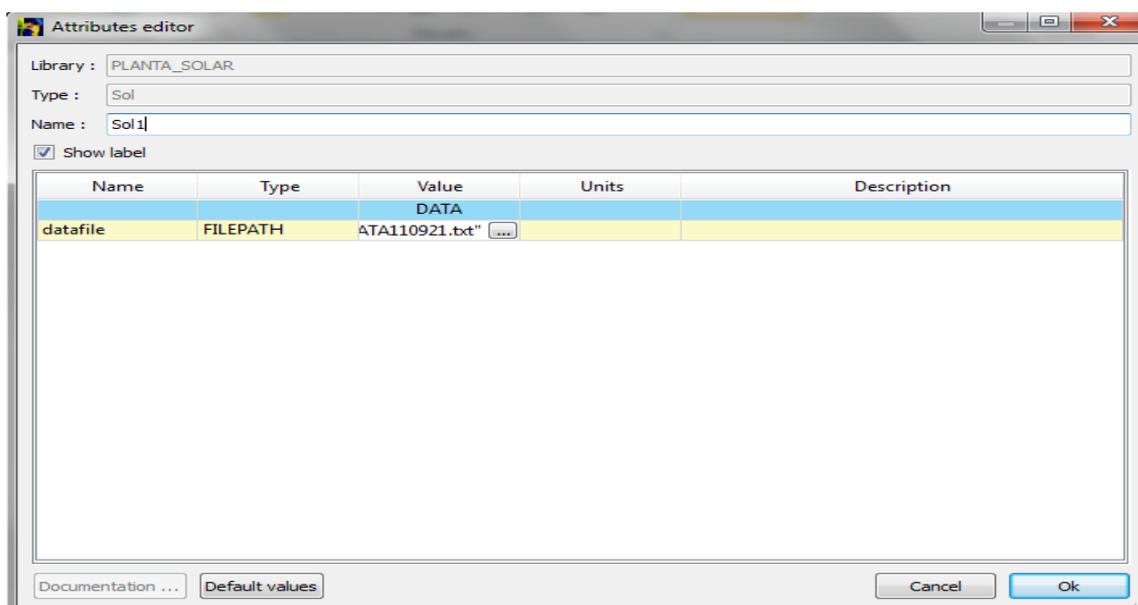


Ilustración 33. Editor atributos componente Sol

Hay que recordar aquí que las variables tipo *DATA* no son modificables al editar el proyecto, pero una vez dentro de la simulación son modificables por el usuario, el cual podrá cambiar las veces que quiera el contenido de las mismas.

DECLS

En el bloque *DECLS* se definen las variables locales o del ámbito privado del componente. Estas variables se emplearán para realizar los cálculos y funciones internas del componente para posteriormente transmitir el resultado al componente TUB_CAPTADOR a través del puerto *Solar*.

En este bloque tenemos dos variables tipo *REAL*, *I* y *Tamb*, que se emplearán para almacenar los valores que toman la irradiancia y la temperatura ambiente a lo largo del día de prueba.

Las dos variables del tipo *TABLE_1D* *t_Tamb* y *t_irradiancia*, son tablas unidimensionales auxiliares que se emplearán para realizar la interpolación de los valores tomados de las tablas de datos reales de la planta. Son tablas de datos para las cuales un valor de entrada devuelve un valor de salida.

Por último se emplea una variable de tipo booleana *BOOLEAN correct_I* para realizar la corrección de la irradiancia cuando ésta alcanza un valor inferior a 10.

INIT

En el bloque de inicio se realiza la **lectura de datos reales de las tablas de texto (extensión .txt) para inicializar las variables *t_Tamb* y *t_irradiancia***. Para ello se emplea la función en EcosimPro® *readTableColsID*, la cual permite leer de un archivo de texto dos columnas y almacenarlas en una dato del tipo *TABLE_ID*. Esta función recibe como parámetros la ruta donde se encuentra el archivo de texto, las dos columnas a leer y la tabla de una dimensión donde se almacenan. En nuestro caso, ya definida la ruta donde se encuentran los datos mediante la variable *datafile*, únicamente hace falta indicar el fichero de datos, y se tomarán las columnas uno y seis y uno y tres para almacenarlas en las tablas *t_Tamb* y *t_irradiancia*. La columna uno del fichero corresponde a la hora, y las columnas seis y tres a la irradiancia y temperatura ambiente respectivamente.

DISCRETE

En este componente se realiza la **inclusión de un evento discreto que en este caso se produce cuando la irradiancia alcanza un valor inferior a 10 W/m²**. Cuando durante la simulación el evento se produzca ($I < 10$), se para la parte continua, se cambia la variable booleana *correct_I* a *TRUE* y se prosigue con la ejecución del bloque *CONTINUOS* donde se realizará el truncamiento de la irradiancia *I* al valor de 10.

CONTINUOUS

Se realiza la **interpolación lineal de los valores contenidos en las tablas** y almacena el resultado en las variables I y T_{amb} para posteriormente transmitir las por el puerto SI .

Para realizar la interpolación de los datos de irradiancia y temperatura ambiente almacenados en las tablas se emplea la función ***linearInterp1D***, la cual recibe como primer argumento la tabla y como segundo argumento la variable con respecto a la cual se realiza la interpolación que en nuestro caso es la variable global $TIME$, la cual es una variable interna de EcosimPro[®] que representa el tiempo de integración instantáneo y es accesible desde cualquier componente, puerto, experimento o función. La función devuelve el valor interpolado que se almacena en la variable correspondiente.

La instrucción ***ZONE*** permite realizar el truncamiento en el valor de la irradiancia si ésta cae por debajo de diez. Esta sentencia permite cambiar entre ecuaciones alternativas durante la simulación, dependiendo del valor de una condición, que en este caso viene dada por la variable booleana $correct_I$. Si en el bloque ***DISCRETE*** se detecta el evento $I < 10$, entonces la variable $correct_I$ pasa a ***TRUE*** y la sentencia ***ZONE*** seleccionará la ecuación $I = 10$. En el momento que el evento desaparece ($correct_I = FALSE$), entonces ***ZONE*** vuelve a seleccionar la ecuación $I = linearInterp1D(t_irradiancia, TIME)$.

Componente Espejos

Segundo componente que se encarga de fijar las condiciones de funcionamiento del colector, en este caso las referentes al campo de espejos de Fresnel. En concreto se encargará de calcular y transmitir al componente principal ***TUB_CAPTADOR*** las variables factor de sombra y porcentaje de espejos enfocados en cualquier instante del día. Para ello dispondrá de un puerto configurado como salida (OUT) del tipo ***cond_espejos***.

La forma de determinar las dos variables que definen las condiciones de los espejos será diferente. Por un lado, la variable $p_espejos$ que contiene el porcentaje del campo de espejos que están enfocados en un momento determinado, se tomará de los datos reales de la planta para un día concreto que se encuentran en los ficheros de texto de datos. Por otro lado, la variable $Factorsombra$ que contiene información del efecto de las sombras sobre el campo de espejos, se evaluará mediante diversas funciones siguiendo el modelo óptico del captador Fresnel explicado en el capítulo 4.

El símbolo y código del componente son los siguientes:

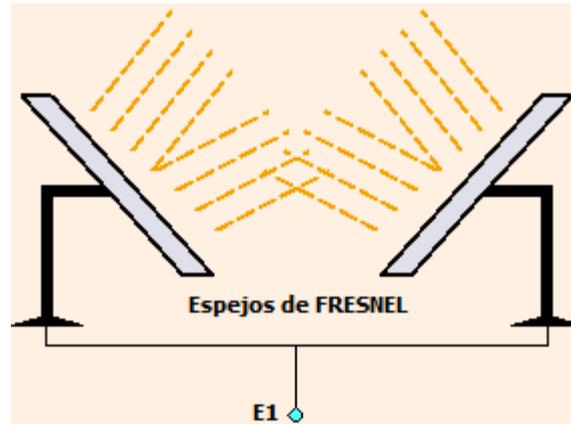


Ilustración 34. Símbolo componente Espejos

```

COMPONENT Espejos

PORTS
    OUT cond_espejos E1      -- Puerto de salida tipo cond_espejos

DATA
    -- Directorio de datos y parámetros configurables
    -- de fecha, hora y orientación de la planta
    FILEPATH datafile="@PLANTA_SOLAR@/experiments"
    INTEGER ano=2011
    INTEGER mes=9
    INTEGER dia=21
    REAL longitud=-6
    REAL latitud=37.41
    REAL hora_comienzo=11.5

DECLS
    -- Variables locales del componente
    -- Tabla auxiliar para adquisición de datos
    -- Porcentaje de espejos enfocados
    TABLE_1D t_p_espejos
    REAL p_espejos
    INTEGER diajuliano      -- Día juliano
    REAL angulodiario      -- Ángulo diario
    REAL Ts                -- Hora solar
    REAL Factorsombra      -- Factor de sombra
    REAL hora              -- Variables temporales
    REAL minuto
    REAL hora_aux

INIT
    -- Inicialización de datos
    -- Lectura fichero columnas porcentajes espejos enfocados
    readTableCols1D("datafile/DATA110921.txt",1,7,t_p_espejos)
    hora=hora_comienzo      -- Inicialización hora simulación
    hora_aux=hora_comienzo
    minuto=0

CONTINUOUS
    -- Interpolación tabla espejos enfocados
    p_espejos = 0.01*linearInterp1D(t_p_espejos,TIME)
    Juliano(ano,mes,dia,diajuliano)      -- Cálculo del día juliano
    angulodiario=angdiario(diajuliano)    -- Cálculo del ángulo diario
    reloj(TIME,hora,minuto,hora_aux)      -- Actualización de hora y minuto
    
```

```
-- Cálculo de la hora solar
Ts=horasolar(hora,minuto,mes,angulodiario,longitud,latitud)
-- Cálculo del factor de sombra
Factorsombra=eficienciageoysombras(angulodiario,Ts,latitud)
-- Envío factor de sombra y espejos enfocados por puerto E1
Factorsombra=E1.sombras
p_espejos=E1.espejosenfoc
END COMPONENT
```

Iremos explicando el funcionamiento de este componente a medida que describamos cada uno de los bloques que lo componen:

PORTS

Este componente dispondrá de un único puerto de comunicación *E1* de tipo *cond_espejos* configurado como salida (OUT) para transmitir al componente principal las variables de factor de sombra (*Factorsombra*) y porcentaje de espejos del campo enfocados hacia el captador (*p_espejos*).

DATA

Este bloque contiene todos los parámetros del componente configurables por el usuario, los cuales han de ser fijados antes de generar la partición y posterior experimento, aunque como ya se ha comentado, es posible su posterior modificación durante la simulación.

La variable *datafile* de tipo *FILEPATH* ya se ha explicado anteriormente y se utiliza para indicar la ruta de archivo donde se encuentran los ficheros de datos reales de la planta de donde se tomarán en el caso de este componente, el porcentaje de espejos enfocados en la planta a lo largo del día.

También se fijan como datos los referentes a la fecha y hora de comienzo de funcionamiento de la planta a la que pertenecen los datos tomados del fichero de texto. Concretamente se establece el día, mes y año en que fueron tomados los datos y la hora de comienzo en la toma de los mismos empleando las variables de tipo *INTEGER día*, *mes*, y *año*, y la variable de tipo *REAL hora_comienzo*. Todos estos datos son imprescindibles para la determinación del factor de sombra que variará en función del día en el que se realice el experimento y de la hora de comienzo del mismo.

Igualmente necesarios son los datos de orientación de la planta para la determinación del factor de sombra, siendo necesario fijar la longitud y latitud de la misma mediante sendas variables de tipo *REAL longitud* y *latitud*.

La ventana del editor de atributos con todos los valores de este componente a fijar, se muestra a continuación:

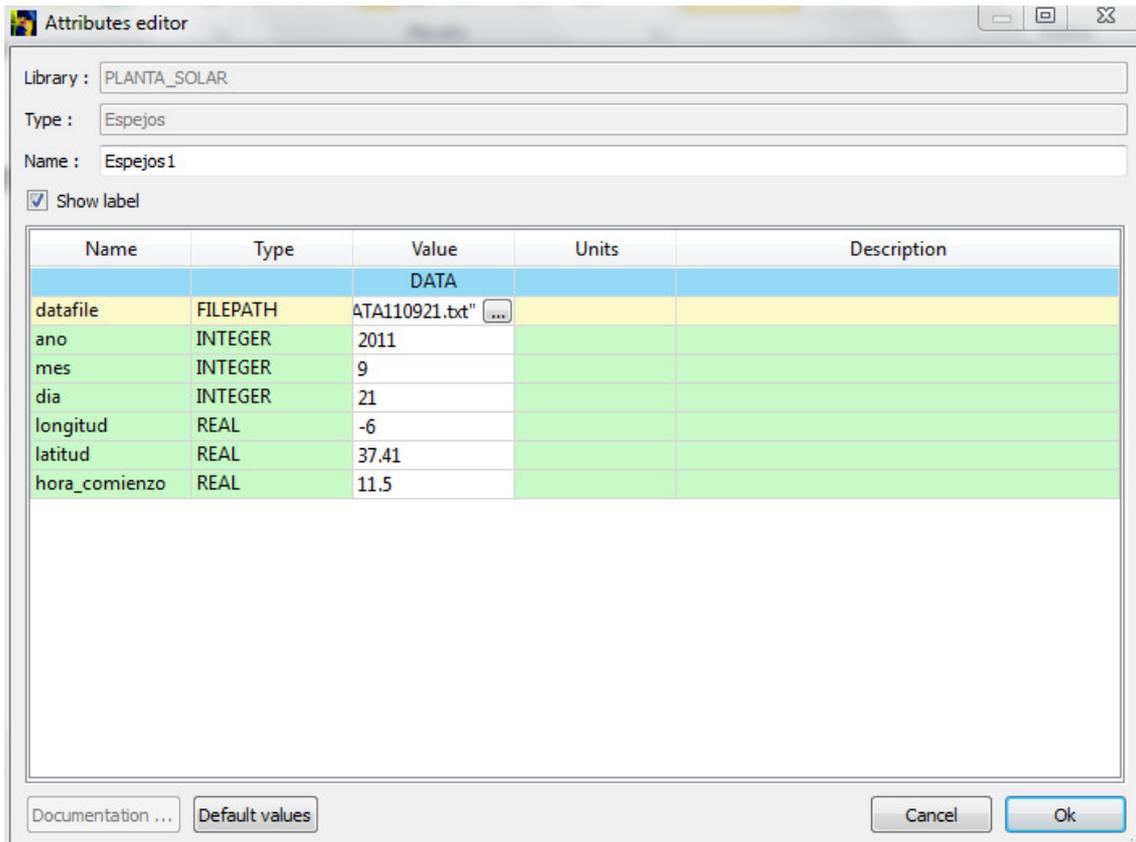


Ilustración 35. Atributos configurables para el componente Espejos

DECLS

Se describen a continuación las variables locales del componente *Espejos*:

En primer lugar tenemos las variables de tipo *TABLE_ID* $t_p_espejos$ y de tipo *REAL* $p_espejos$ que se empleará para realizar la interpolación de los valores de porcentaje de espejos enfocados a lo largo del día tomados de la columna correspondiente en el fichero de texto de datos reales de la planta. Primero se leerán los valores del fichero de texto y se almacenarán en la tabla $t_p_espejos$. Finalmente se realiza la interpolación en el tiempo empleando la variable *TIME* y se almacenará el resultado en $p_espejos$.

Las variable de tipo entero (*INTEGER*), día juliano (*diajuliano*), y las variables de tipo real (*REAL*), ángulo diario (*angulodiario*) y hora solar (T_s), son variables intermedias del modelo óptico que se requieren para calcular el factor de sombra en la planta (*Factorsombra*) a lo largo de un día concreto. Estas variables han sido ya explicadas detenidamente en el modelo óptico expuesto en el capítulo 4.

Por último, las variables de tipo *REAL* $hora$, $hora_aux$ y $minuto$ se emplean para realizar la actualización del instante del día (hora y minuto), en el que nos encontramos a partir de una hora de comienzo. La actualización de la hora es necesaria para el cálculo de la variable hora solar T_s .

INIT

En el bloque de inicialización, realizaremos la lectura de la columna del fichero de datos correspondiente al porcentaje de espejos enfocados a lo largo del día (columna siete) mediante la función *readTableColsID* y almacenaremos el resultado en la variable de tipo *TABLA_ID t_p_espejos*.

También se inicializarán las variables *hora*, *hora_aux* y *minuto* para situar el reloj a la hora de comienzo de la simulación y a partir de ahí ir actualizándolo mediante la variable de tipo global *TIME*.

CONTINUOUS

En este bloque se realiza el cálculo de las dos variables de contorno que este componente debe transmitir al componente principal *TUB_CAPTADOR*. Veremos a continuación cómo se calculan ambas.

El porcentaje de espejos enfocados, *p_espejos*, se obtiene de la misma forma que ya se explicó para obtener la irradiancia y la temperatura ambiente en el componente anterior *Sol*, es decir, empleando la función *linearInterpID* para realizar la interpolación con respecto a la variable *TIME* de los datos contenidos en la tabla *t_p_espejos* y que han sido leídos previamente del fichero de datos en el bloque *INIT*. En este caso, también se realiza una multiplicación de los valores por 0,01 puesto que los datos de porcentaje de espejos tomados de la planta vienen dados en tanto por cien y nosotros necesitamos para nuestro modelo el dato en tanto por uno.

La segunda variable a transmitir, *FactorSombra*, es más difícil de calcular puesto que su valor no viene dado en el fichero de datos reales tomados de la planta, sino que es preciso realizar su cálculo matemáticamente mediante un modelo óptico determinado (capítulo 4), que contiene diferentes funciones las cuales iremos explicado ordenadamente.

La primera de ellas es la función ***Juliano***. Esta función tiene como entradas los datos *año*, *mes* y *día*, y devuelve como salida el valor del día juliano entre 1 y 366. Su código es el siguiente:

```
USE MATH
-- Función que realiza el cálculo del día juliano
-- Entradas: año, mes y día
-- Salidas: día juliano [1-366]
FUNCTION NO_TYPE Juliano (IN INTEGER ano, IN INTEGER mes, IN INTEGER dia, OUT
INTEGER diajuliano)
DECLS
    -- Días para cada mes de un año normal
    INTEGER tabla[12]={31,28,31,30,31,30,31,31,30,31,30,31}
    -- Días para cada mes año bisiesto
    INTEGER tabla_bis[12]={31,29,31,30,31,30,31,31,30,31,30,31}
    INTEGER suma=0
```

```

BODY
  -- Comprobación año bisiesto
  IF (remainder(ano,4)==0 AND (remainder(ano,100)>0 OR
  remainder(ano,400)==0)) THEN
    FOR (i IN 1,mes-1)           -- Año bisiesto
      suma=suma+tabla_bis[i]
    END FOR
  ELSE                             -- Año normal
    FOR (i IN 1,mes-1)
      suma=suma+tabla[i]
    END FOR
  END IF

  diajuliano=suma+dia           -- Cálculo día juliano

END FUNCTION

```

La función genera dos tablas con los días para cada mes en el caso de año normal y año bisiesto. A continuación se comprueba si estamos en año bisiesto o no mediante la comprobación del resto en distintas divisiones. Para ello se emplea la función de la librería *MATH*, *remainder*, la cual devuelve el resto de la división de dos números enteros. Dependiendo del año en que estemos se utiliza una tabla u otra y se suman los días de los meses hasta el mes actual en el que nos encontramos, sin incluir éste último. Finalmente a la suma obtenida se añaden los días que han transcurrido del mes actual obteniendo el día juliano [1-366].

La obtención del día juliano es el paso previo para la obtención del ángulo diario. La función *angdiario* recibe como entrada el día juliano [1-366], y devuelve el valor del ángulo diario en radianes. Para ello simplemente aplica la siguiente ecuación que relaciona el ángulo diario en radianes con el día juliano:

$$Angulodiario = \left(\frac{2\pi}{365}\right) \cdot (diajuliano - 1)$$

Su código es el siguiente:

```

USE MATH
-- Cálculo del ángulo diario en radianes
-- Entradas a la función: día juliano (1-366)
-- Salidas de la función: ángulo diario [radianes]
FUNCTION REAL angdiario(REAL diajuliano)

DECLS
  REAL angulodiario

BODY
  angulodiario=(2*PI/365)*(diajuliano-1)
  RETURN angulodiario

END FUNCTION

```

La función *reloj* realiza la tarea de actualización de la hora y minuto del día a partir del instante de comienzo de la simulación. La variable global interna en EcosimPro® *TIME*, representa el tiempo instantáneo de simulación en segundos que irá aumentando linealmente desde la hora de comienzo de la misma con un incremento que viene determinado por la constante de integración *CINT*. Puesto **que los datos de tiempo de la planta vienen dados en horas como unidad de tiempo, consideraremos que el tiempo de simulación *TIME* también viene dado en la misma base.**

Teniendo en cuenta lo anterior y que **para calcular la hora solar es necesario obtener la hora y minuto del día en el que nos encontramos, debemos ser capaces de calcular estos datos a partir de la hora de inicio de la simulación y del valor de la variable *TIME* o tiempo de simulación transcurrido hasta el momento.** Esto se realiza mediante la función *reloj*.

La función *reloj* recibe como argumento la variables *TIME* que se pasa por valor (prefijo *IN*), y las variables *hora*, *minuto* y *hora_aux* que se pasan por referencia (prefijo *OUT*) y por tanto, sus valores serán actualizados por la función. El código de la función es el siguiente:

```

-- Realiza la actualización de la hora y los minutos en función de la variable TIME
FUNCTION NO_TYPE reloj(IN REAL time, OUT REAL hora,OUT REAL minuto,
OUT REAL hora_aux)

DECLS
  CONST REAL tau=60/3600          -- Relación minuto/hora
BODY
  IF ((time-hora_aux)>=tau) THEN minuto=minuto+1 -- Incremento minuto
    hora_aux=time                    -- Actualización hora auxiliar
  END IF
  IF (minuto>=60) THEN hora=hora+1    -- Incremento hora
    minuto=minuto-60                 -- Inicialización minuto
  END IF
  IF (hora>=24) THEN hora=hora-24
  END IF

END FUNCTION

```

Con cada incremento de tiempo dado por *CINT*, la función comprobará el tiempo que ha transcurrido y actualizará la hora y el minuto en el que se encuentra la simulación. Para ello se crea una variable auxiliar *hora_aux*, la cual almacenará el último valor de la variable *time* en el cual se modificó *minuto*. Si al restar a *time* la variable *hora_aux*, la diferencia es superior o igual a *tau* (valor equivalente a un minuto en base de tiempo hora), entonces se incrementará la variable minuto y se actualizará *hora_aux* al instante actual. Cuando *minuto* sea mayor o igual a sesenta, se realizará el incremento de la variable hora.

Con todos los datos obtenidos por las funciones anteriores, pasamos a calcular ya el valor de la hora solar real. Para ello, emplearemos la función *horasolar*, la cual recibe como argumentos las variables *hora* [1-24], *minuto* [0-60], *mes* [1-12], *angulodiario* (radianes) y la *longitud* (grados), y devuelve el valor de la hora solar actual que se almacena en la variable de tipo *REAL Ts*.

El código que presenta la función es el siguiente:

```

USE MATH
-- Cálculo de la hora solar real
-- Entradas de la función: hora actual oficial [1-24], minuto [0-60], mes [1-12], ángulo diario
[radianes], longitud [grados]
-- Salida de la función: hora solar real [decimal]
FUNCTION REAL horasolar(REAL horaactualoficial, INTEGER minuto, INTEGER mes, REAL
angulodiario, REAL longitud, REAL latitud)

DECLS -- Variables para el cálculo de correcciones
REAL Longitudmeridianocentralhuso=0
REAL diferenciaporposicion
REAL diferenciapormes
REAL Et
REAL diferenciatotal
REAL Ts

BODY
horaactualoficial=horaactualoficial+minuto/60 -- Hora actual oficial en decimal
-- Diferencia por posición en minutos
diferenciaporposicion=(longitud-Longitudmeridianocentralhuso)*4
IF (diferenciaporposicion<0.) THEN
    diferenciaporposicion=-1*diferenciaporposicion
END IF
-- Diferencia por horario de verano
IF (mes<4 OR mes>10) THEN
    diferenciapormes=60 -- Una hora si no estamos entre abril y octubre
ELSE
    diferenciapormes=120 -- Dos horas entre abril y octubre
END IF
-- Ecuación del tiempo en minutos
Et=(0.000075+0.001868*cos(angulodiario)-0.032077*sin(angulodiario)-
0.014615*cos(2*angulodiario)-0.04089*sin(2*angulodiario))*229.18
-- Diferencia total en minutos
diferenciatotal=diferenciapormes+diferenciaporposicion-Et
Ts=horaactualoficial-(diferenciatotal/60)
RETURN Ts

END FUNCTION

```

El procedimiento para el cálculo de la hora solar real se ha dividido en cuatro fases diferentes:

1. Cálculo de la corrección por posición COL: desviación de la hora oficial actual por situación de la planta dada por la posición geográfica de la misma (longitud del lugar), teniendo en cuenta que 1° de longitud equivale a 4 minutos de reloj.
2. Cálculo de la corrección por horario de verano COV: desviación en la hora solar producida por los cambios de hora realizados por el hombre. Entre abril y octubre la corrección será de 2 horas (120 minutos) y fuera de dicho intervalo será de 1 hora.
3. Cálculo de la ecuación del tiempo por la fórmula de Spencer con Γ ángulo diario en radianes:

$$E_t = (0,000075 + 0,001868 \cdot \cos(\Gamma) - 0,032077 \cdot \sin(\Gamma) - 0,014615 \cdot \cos(2\Gamma) - 0,04089 \cdot \sin(2\Gamma)) \cdot 229,18$$

4. Cálculo de la hora solar real:

$$T_s = Hora_{oficial} - COL - COV + E_t$$

Para una mayor explicación sobre las fórmulas y conceptos ver el modelo óptico del captador explicado en el capítulo 4.

Por último, llegamos ya al cálculo del factor de sombra, dato final que nos interesa y que será enviado a través de la variable *sombras* del puerto *El* del componente. La función *eficienciageoysombras* se encargará de realizar esta labor calculando el factor de sombra del lazo para cada terraza. Es decir, se calcula la superficie que no se pierde por las sombras. Para ello tiene en cuenta la orientación e inclinación del campo, las sombras de unos espejos sobre otros, así como la zona que no tiene sombras por efecto del azimut del sol.

Las entradas de la función serán el ángulo diario (radianes) y la hora solar real (decimal), calculadas mediante las funciones anteriores, y la latitud (grados) del lugar en el que se encuentra la planta introducida por el usuario como dato en este componente. La función devuelve un valor entre 0 y 1 correspondiente al coeficiente factor de sombra de la planta.

La implementación de esta función ha resultado de gran complejidad debido a que EcosimPro® no posee la gran variedad de herramientas matemáticas que tiene Matlab, por lo que se ha tenido que recurrir a la realización de funciones intermedias y diversos bucles auxiliares para su ejecución completa en lenguaje EL. También se ha creado una librería auxiliar *MATH_2*, que contiene diversas funciones matemáticas necesarias para resolver esta función.

El código completo de esta función no se presenta en este capítulo debido a la elevada extensión del mismo. Su consulta puede realizarse en el ANEXO I que se adjunta. Su contenido refleja los cálculos matemáticos y geométricos necesarios para la obtención del factor de sombra en la planta según el modelo óptico explicado en el

capítulo 4 de este documento, por lo que se aconseja su lectura previa, antes de abordar el código de esta función para una correcta comprensión del mismo.

Componente Entrada Flujo

Tercer y último componente encargado del establecimiento de las condiciones de funcionamiento del captador. Las variables a determinar por éste, son las relativas al estado de entrada del fluido en el captador, concretamente su caudal y temperatura. Una vez calculadas estas variables, serán enviadas al componente principal *TUB_CAPTADOR* mediante un puerto de tipo *fluido* configurado como salida (*OUT*).

El símbolo y código del componente son los siguientes:

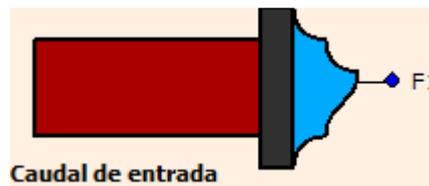


Ilustración 36. Símbolo componente Entrada_Flujo

```

-- Establecimiento de caudal y temperatura fluido de entrada
COMPONENT Entrada_Flujo

PORTS
    OUT fluido F1          -- Puerto de salida tipo fluido
DATA -- Directorio de datos de la planta
    FILEPATH datafile="@PLANTA_SOLAR@/experiments"
DECLS
    TABLE_1D t_caudal    -- Tablas 1D para adquisición de datos
    TABLE_1D t_Tent
    REAL caudal           -- Caudal
    REAL Tent             -- Temperatura entrada fluido
    BOOLEAN correct_caudal -- Corrector de caudal
INIT -- Lectura de datos
    readTableCols1D("datafile/DATA110921.txt",1,2,t_caudal)
    readTableCols1D("datafile/DATA110921.txt",1,4,t_Tent)
DISCRETE
    WHEN (caudal<0)THEN correct_caudal= TRUE -- Caudal negativo
    END WHEN
CONTINUOUS
    caudal= ZONE (correct_caudal== TRUE) 0 -- Corrección de caudal
    OTHERS linearInterp1D(t_caudal,TIME)/3600 -- Interpolación de datos
    Tent = linearInterp1D(t_Tent,TIME)
    Tent=F1.T -- Envío por el puerto de salida
    caudal=F1.w
END COMPONENT
    
```

Las variables a calcular *caudal* y *Tent*, se tomarán del directorio de datos de la planta *datafile* mediante la función *readTableCols1D*, almacenando las columnas en las tablas

auxiliares 1D t_{caudal} y t_{Tent} . Después se realizará la interpolación lineal de las mismas respecto del tiempo mediante la función *linearInterp1D* enviando el resultado a través del puerto de salida tipo *fluido* F1. También se introduce un evento discreto para la corrección del caudal de entrada cuando éste sea negativo.

El código de este componente es exactamente igual al del componente *Sol* ya explicado, por lo que no se entrará a realizar una nueva explicación sobre el mismo.

Componente Configuración TUB

La función de este componente es la de configurar determinados aspectos físicos, geométricos y de eficiencia del captador solar. Para ello, realiza una serie de cálculos previos a partir de unos datos dados para obtener diferentes parámetros necesarios en el cálculo de la temperatura de salida del captador. Los parámetros calculados, serán transmitidos al componente principal *TUB_CAPTADOR* mediante un puerto del tipo *parámetros_TUB*.

La razón principal del diseño de este componente es la descargar al componente principal *TUB_CAPTADOR* de diferentes cálculos convirtiéndolo en un elemento más genérico mediante la elección de distintos valores en este componente. Mostramos el símbolo y código del componente:



Ilustración 37. Símbolo componente Configuración_TUB

```

USE MATH
-- Cálculos iniciales de eficiencias, variables dimensionales y características del metal
COMPONENT Configuracion_TUB

PORTS
    OUT parametros_TUB P          -- Puerto de salida tipo parámetros_TUB
DATA -- Datos configurables
    REAL G=0.5*11                -- Apertura del colector [m]
    -- Definición de reflectividad y factores de eficiencia
    REAL Absortancia=0.94
    REAL Reflectividad=0.9*0.49
    REAL Transmisividad=0.96
    REAL Reflectividadsecundaria=0.77
    REAL factorensuciamiento1=0.8
    REAL factorensuciamiento2=0.8
    -- Características del metal DIN 1.4404
    REAL pm=8027                  -- Densidad [Kg/m3]
    REAL Cm=500                   -- Calor específico [J/Kg°C]
    REAL diametrointeriortubo=0.068 -- Diámetro tuberías [m]
    REAL diametroexteriortubo=0.071
    REAL Tint=0.5                 -- Tiempo de integración [s]
    
```

```

DECLS -- Variables locales
      REAL Am                -- Áreas transversales metal y fluido [m2]
      REAL Af
      REAL L                -- Longitud circunferencia de la tubería [m]
      REAL Eficienciamedia  -- Eficiencia media
CONTINUOUS
      -- Cálculo de variables
      L=PI*diametrointeriortubo
      Af=PI*diametrointeriortubo**2/4
      Am=PI*(diametroexteriortubo**2-diametrointeriortubo**2)/4
      Eficienciamedia=Absortancia*Reflectividad*Transmisividad*Reflectividadse
      cundaria*factorensuciamiento1*factorensuciamiento2
      -- Envío de datos por puerto de salida
      P.L=L
      P.Af=Af
      P.Am=Am
      P.Eficienciamedia=Eficienciamedia
      P.G=G
      P.pm=pm
      P.Cm=Cm
      P.Tint=Tint
END COMPONENT
    
```

Vemos los distintos bloques del código:

PORTS

El componente emplea un único puerto *P* de tipo *parámetros_TUB*, configurado como salida (OUT), para comunicar los datos y variables calculadas al componente *TUB_CAPTADOR*.

DATA

En este bloque se encuentran los datos configurables por el usuario de este componente. Se fijan entre otros, los distintos factores que influyen en la eficiencia del captador como la absortancia, reflectividad y reflectividad secundaria, transmisividad y factores de ensuciamiento de los espejos. Para éste último, se han supuesto factores de ensuciamiento de 0,9 si los espejos están limpios y de 0,8 si están sucios. También se definen aspectos de la tubería como diámetros interior y exterior en metros, y características del metal como su densidad *pm* (kg/m³) y calor específico *Cm* (J/kg°C).

El editor de atributos donde configurar los parámetros de este componente se muestra en la siguiente ilustración:

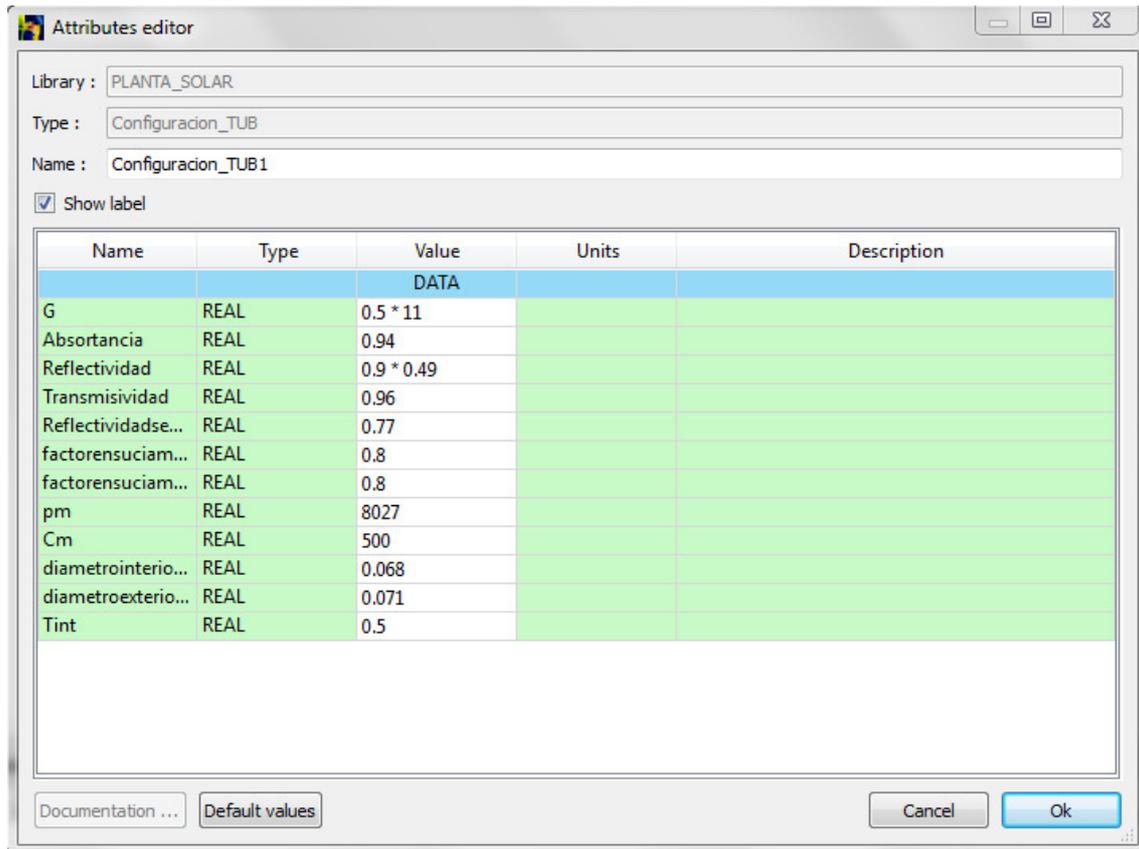


Ilustración 38. Atributos configurables para el componente Configuración_TUB

DECLS

Se definen las variables locales calculadas por el componente: áreas transversales efectivas para el metal y fluido A_m y A_f en metros cuadrados, la longitud de la circunferencia de la tubería L en metros y la eficiencia media del captador.

CONTINUOUS

Se realiza en primer lugar el cálculo de las variables locales definidas en el bloque *DECLS* para después transmitir las junto con el resto de datos introducidos en el editor de atributos por el puerto de salida del componente.

El cálculo de la eficiencia media se realiza multiplicando cada uno de los factores introducidos como atributos que intervienen en la misma. Para la longitud y áreas de la sección de tubería se emplean las fórmulas geométricas correspondientes al perímetro y área de un círculo.

$$L = \pi \cdot d$$

$$A = \pi \cdot \frac{d^2}{4}$$

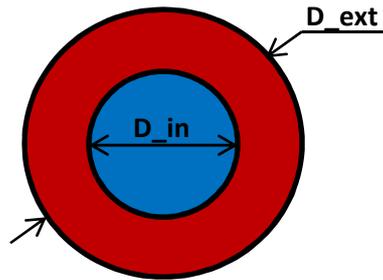


Ilustración 39. Sección tubería

Componente TUB_CAPTADOR

Componente principal del modelo que calcula la temperatura del fluido a la salida del captador a partir de las condiciones de contorno dadas por los componentes *Sol*, *Espejos* y *Entrada_Flujo* a través de los puertos de entrada correspondientes, y de las ecuaciones diferenciales del modelo de parámetros distribuidos explicadas en el capítulo 5. La temperatura de salida del fluido será enviada al componente *Salida_Flujo* a través de un puerto de salida tipo *flujo* para compararla con los datos reales tomados de la planta.

El símbolo y código para este componente se muestran a continuación:

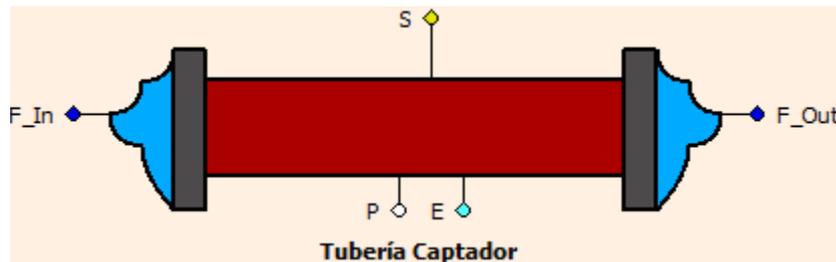


Ilustración 40. Símbolo componente TUB_CAPTADOR

```

-- Cálculo de temperatura del fluido a la salida

COMPONENT TUB_CAPTADOR (INTEGER N)  -- Parámetro número de
                                     -- segmentos de tubería

PORTS
  IN fluido F_In                    -- Puertos de entrada para variables de contorno y
  IN solar S                          -- parámetros de la tubería
  IN cond_espejos E
  IN parametros_TUB P
  OUT fluido F_Out                  -- Puerto de salida tipo fluido

DATA
  REAL incx=1                        -- Longitud del segmento

DECLS
  REAL Tm[N]                         -- Tª metal de cada uno de los N segmentos
  REAL Tf[N]                         -- Tª fluido de cada segmento
  REAL Tfsc[N]                      -- Tª fluido en estado estacionario para cada segmento
    
```

```

REAL HI[N]      -- Pérdidas metal-ambiente para cada segmento
REAL pf[N]      -- Características del fluido para cada segmento
REAL Cf[N]
REAL Ht[N]      -- Transmisión de calor metal-fluido para cada segmento

INIT
  FOR( i IN 1,N ) -- Tª inicial del metal y del fluido
    Tm[i]= 66.05
    Tf[i]=59.05
  END FOR
CONTINUOUS
  Tf[1]=F_In.T  -- Tª fluido en el primer segmento igual a temperatura de entrada
  -- Cálculos para el primer segmento de tubería
  HI[1]=perdidas_metal(Tm[1],S.Tamb)
  caract_fluido(Tf[1],F_In.w,pf[1],Cf[1],Ht[1])
  Tfsc[1]=Tf[1]+((P.L*Ht[1]*P.Tint)/(P.Af*pf[1]*Cf[1]))*(Tm[1]-Tf[1])
  Tm[1]*(P.pm*P.Cm*P.Am)*(CINT/P.Tint)=S.I*P.G*P.Eficienciaimedia*E.s
  ombras*E.espejos_enfoc-HI[1]*P.G*(Tm[1]-S.Tamb)-P.L*Ht[1]*(Tm[1]- Tf[1])
  -- Cálculos segmentos restantes
  EXPAND_BLOCK (i IN 2,N)
  -- Coeficiente pérdidas metal-ambiente para cada segmento
  HI[i]=perdidas_metal(Tm[i],S.Tamb)
  -- Características del fluido y coeficiente de transmisión metal-fluido en cada
  segmento
  caract_fluido(Tf[i],F_In.w,pf[i],Cf[i],Ht[i])
  -- Temperatura fluido en estado estacionario
  Tfsc[i]=Tf[i]+((P.L*Ht[i]*P.Tint)/(P.Af*pf[i]*Cf[i]))*(Tm[i]-Tf[i])
  -- Ecuaciones diferenciales obtenidas del modelo de parámetros distribuidos
  Tm[i]*(P.pm*P.Cm*P.Am)*(CINT/P.Tint)=S.I*P.G*P.Eficienciaimedia*E.s
  ombras*E.espejos_enfoc-HI[i]*P.G*(Tm[i]-S.Tamb)-P.L*Ht[i]*(Tm[i]-Tf[i])
  Tf[i]*(CINT/P.Tint)=((P.L*Ht[i])/(P.Af*pf[i]*Cf[i]))*(Tm[i]-Tf[i])-
  (F_In.w/(P.Af*incx))*(Tfsc[i]-Tfsc[i-1])
  END EXPAND_BLOCK
  F_Out.w=F_In.w      -- Envío de caudal
  F_Out.T=Tf[N]      -- Envío temperatura en último segmento

END COMPONENT

```

Como vemos en la definición de este componente, podemos elegir el nivel de discretización del mismo mediante la introducción de número de segmentos que posee a través del parámetro N . De esta forma podemos obtener una respuesta más o menos aproximada a la salida real del sistema. Cuanto mayor sea N , mayor será la exactitud del modelo, pero también se incrementarán los tiempos de cálculo necesarios para resolver el sistema.

Vemos los bloques que lo componen:

PORTS

Se definen cuatro puertos de entrada (*IN*) y un puerto de salida. Los puertos de entrada *F_In*, *S*, *E* y *P*, permiten recibir las variables caudal, temperatura del fluido de entrada, temperatura ambiente, irradiancia, porcentaje de espejos, factor de sombra y otros parámetros de la tubería que establecerán los componentes correspondientes explicados anteriormente. El puerto de salida tipo *fluido F_Out*, transmite al componente *Salida_Flujo* el resultado del cálculo de la temperatura del fluido a la salida del captador.

DATA

Como dato se introduce únicamente la longitud en metros de cada segmento en que se divide la tubería: *incx*. El editor de atributos para este componente se muestra en la siguiente figura:

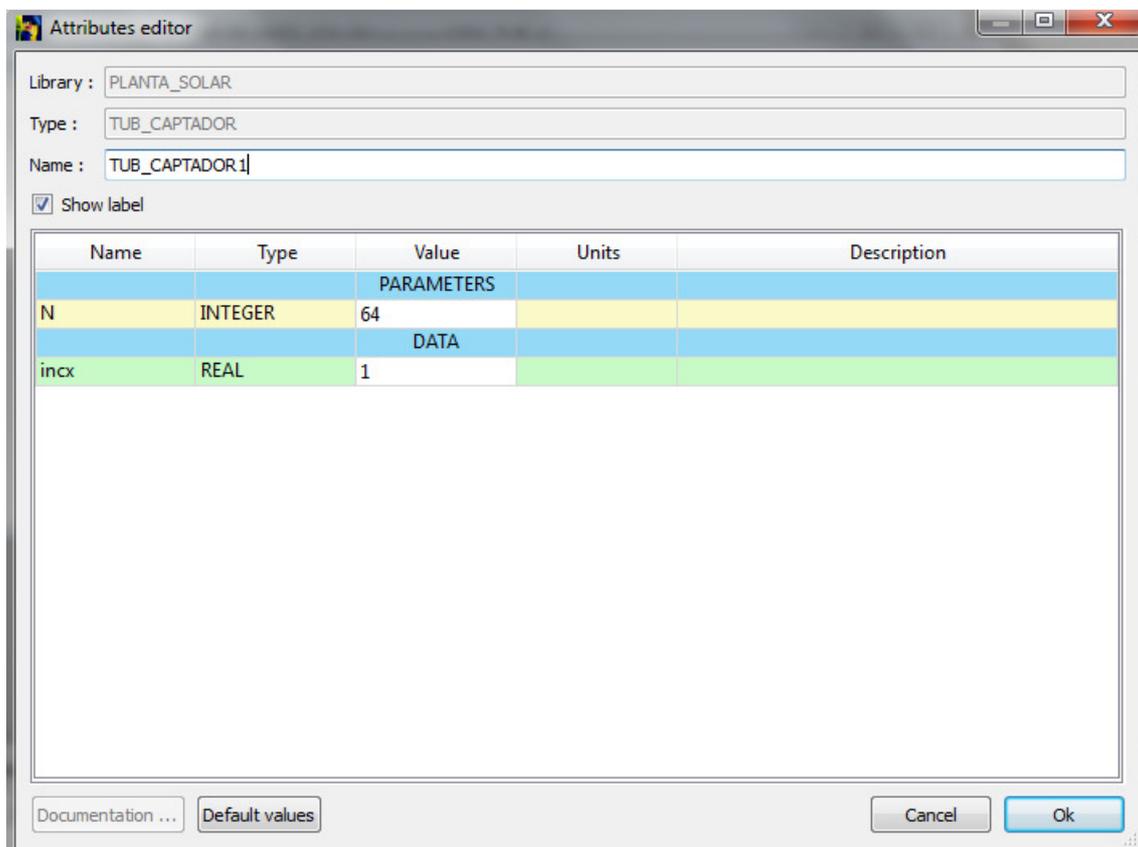


Ilustración 41. Atributos configurables para el componente TUB_CAPTADOR

El usuario puede elegir el número de segmentos de la tubería y la longitud de los mismos determinando la discretización uniforme del sistema.

DECLS

Se definen los vectores de tamaño “N” que contendrán los valores para cada uno de los segmentos de temperatura del metal, del fluido, coeficientes de pérdida y de transmisión y características termodinámicas del fluido en cada instante de tiempo. Como

vemos, no solo la temperatura dependerá de la variable longitudinal, sino también los coeficientes que intervienen en los fenómenos de pérdida y transferencia de calor, Hl y Ht , y las características termodinámicas del fluido Cf y Pf .

INIT

Se inicializan los valores de temperatura de metal y de fluido para todos los segmentos de la tubería. Se toma como temperatura inicial de fluido. la temperatura de entrada del mismo al captador en el inicio de la simulación y se considera la misma temperatura en todos los segmentos. La temperatura inicial del metal se toma también constante en toda la tubería y $7\text{ }^{\circ}\text{C}$ por encima de la temperatura del fluido.

CONTINUOUS

Se realizan primeramente los cálculos para el primer segmento de tubería, estableciendo como ecuación **la igualación de la temperatura del fluido en el primer segmento de tubería con la temperatura del fluido de entrada** dada en el puerto F_In que viene determinada por el componente *Entrada_Fluido*.

El vector de coeficientes de pérdida del metal $Hl[N]$, se calcula mediante la función *perdidas_metal*, que recibe como entradas la temperatura del metal en el segmento y la temperatura ambiente en $^{\circ}\text{C}$ y devuelve el vector actualizado en $\text{W}/\text{m}^2\text{ }^{\circ}\text{C}$. Su código es el siguiente:

```
-- Cálculo del coeficiente de pérdidas metal-ambiente por m2 de superficie para cada trozo de tubo.  
-- Entradas a la función: Temperatura metal [°C] Temperatura ambiente °C  
-- Salidas de la función: Coeficiente de pérdidas metal-ambiente [W/(m2°C)]  
  
FUNCTION REAL perdidas_metal (IN REAL Tm,IN REAL Tamb)  
DECLS  
    REAL HI  
    CONST REAL Sup=64*11*0.5  
BODY  
    -- Calculamos por metro cuadrado de espejo para perdidas de tubo  
    HI=(4.5659247191149893*1e-1/352)*(Tm-Tamb)-(1.0062045206593238*1e2/Sup)  
    RETURN HI  
  
END FUNCTION
```

Los vectores de coeficientes de transmisión metal-fluido $Ht[N]$ ($\text{W}/\text{m}^2\text{ }^{\circ}\text{C}$), densidad $pf[N]$ (Kg/m^3) y poder calorífico $Cf[N]$ ($\text{J}/\text{kg}^{\circ}\text{C}$) del fluido en cada instante de tiempo, se calculan mediante la función *caract_fluido*, que recibe como entradas la temperatura ($^{\circ}\text{C}$) y caudal (kg/s) de fluido de cada segmento en un instante determinado. Su código es el siguiente:

```

-- Cálculo de las características físicas y termodinámicas del fluido, así como el coeficiente
-- de calor por convección metal-fluido de cada trozo en que se ha dividido el lazo de
colectores.
-- Entradas a la función: Temperatura del fluido [°C] caudal [kg/s]
-- Salidas de la función: densidad pf [kg/m3] Calor específico Cf [J/(kg°C)]
--
-- Coeficiente de transmisión de calor por convección metal-fluido
[W/(m²°C)]

FUNCTION NO_TYPE caract_fluido (IN REAL T, IN REAL caudal, OUT REAL pf, OUT REAL Cf,
OUT REAL Ht)
DECLS
    REAL Hv
BODY
    pf=-0.002549807548*T**2-0.202618731192*T+1003.917572929273
    Cf= 0.000000516739*T**4 -0.000156862606*T**3+ 0.027679455287*T**2-
    1.626420964922*T+ 4207.403959277281
    Hv=0.000000001338864e5*T**4-
    0.000000778905228e5*T**3+0.000187251179615e5*T**2-
    0.025731157369768e5*T+4.108383857418933e5
    Ht=Hv*(caudal)**0.8

END FUNCTION

```

Cada una de las ecuaciones expresadas en las dos funciones anteriores para el cálculo de los coeficientes Hl y Ht y características del fluido pf y Cf , han sido ajustadas por mínimos cuadrados empleando datos del fabricante.

El resto de ecuaciones implementadas en este bloque se corresponden con las del modelo de parámetros distribuidos, y ya han sido explicadas en profundidad en el capítulo 5.

Todas las ecuaciones y funciones anteriores son insertadas dentro del bloque EXPAND_BLOCK de manera que queden establecidas para cada uno de los segmentos de la tubería según el índice correspondiente. La temperatura calculada en el último segmento N , será la enviada al componente *Salida_Flujo* por el puerto F_Out .

Componente Salida Flujo

Este componente recoge la temperatura del fluido a la salida de la tubería calculada por TUB_CAPTADOR a través del puerto de entrada tipo *fluido* y realiza también la adquisición de los datos de temperatura de salida reales del fichero de datos de la planta. Una vez tenga la temperatura de salida real y la del modelo, calcula la “Integral del valor Absoluto del Error ponderado en el Tiempo” ITAE.

La obtención del error integral ITAE, nos va a permitir tener una medida del error cometido por el modelo a lo largo de toda la simulación pudiendo extraer conclusiones posteriores.

La “Integral del valor Absoluto del Error ponderado en el Tiempo” ITAE, es un criterio integral empleado para la sintonización de controladores. Se trata de un método dinámico de lazo cerrado que utiliza toda la información, desde $t=0^+$ hasta el nuevo estado estacionario. El objetivo es optimizar una función de costo dependiente del error, en nuestro caso, la integral del tiempo multiplicada por el valor absoluto del error ITAE. Se define como sigue:

$$ITAE = \int t \cdot |error(t)| \cdot dt \quad (6)$$

ITAE es insensible a los errores iniciales, y a veces inevitables, pero penaliza fuertemente los errores que permanecen a lo largo del tiempo. Por tanto, un error a tiempos avanzados tendrá una gran reacción del controlador.

El motivo de la elección de este criterio es que nuestro sistema real, el captador de la planta solar, bajo condiciones normales, va a estar funcionando constantemente durante grandes intervalos de tiempo, por lo que, a pesar de que se pueden producir errores mayores en los primeros instantes de funcionamiento, algo natural y asumible, nos interesa sobre todo que pasados éstos, y una vez se encuentre la planta a un rendimiento óptimo, podamos atajar todos los errores siguientes sin que se produzcan grandes alteraciones en la temperatura de salida del fluido.

No está dentro de los objetivos de este proyecto el diseño de controladores para el funcionamiento de la planta, sino tan sólo su modelado. Por tanto no existen entradas de referencia de temperatura al modelo para calcular el error integral. Sin embargo, a modo informativo, se ha calculado el término ITAE en base al error cometido por el modelo con respecto a la temperatura real de salida, permitiéndonos obtener una idea del error acumulado por el modelo a lo largo de todas las horas de simulación.

Se muestra ahora el símbolo y el código para este componente:

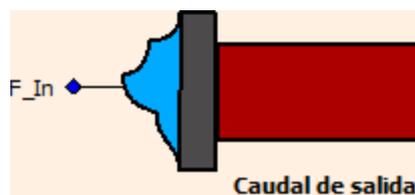


Ilustración 42. Símbolo componente Salida_Flujo

```
-- Obtención temperatura real de salida y cálculo de ITAE
COMPONENT Salida_Flujo

PORTS
  IN fluido F_In  -- Puerto de entrada temperatura del modelo

DATA
  -- Directorio del fichero de datos
  FILEPATH datafile="@PLANTA_SOLAR@/experiments"
  REAL Tini = 11.5      -- Hora de comienzo de la simulación
```

```

DECLS
  TABLE_1D t_Tsreal    -- Tabla para adquisición de datos
  REAL Tsreal          -- Tª real de salida
  REAL ITAE            -- Integral del valor Absoluto del Error ponderado en el Tiempo
INIT
  -- Lectura de datos del fichero
  readTableCols1D("datafile/DATA110921.txt",1,5,t_Tsreal)
CONTINUOUS
  -- Interpolación lineal de la temperatura de salida real
  Tsreal = linearInterp1D(t_Tsreal,TIME)
  ITAE'=(TIME-Tini)*(abs(Tsreal-F_in.T))      -- Cálculo de ITAE
END COMPONENT

```

Vemos los bloques:

PORTS

Se define un único puerto de entrada tipo *fluido F_In* para recoger la temperatura de salida del fluido calculada en el componente *TUB_CAPTADOR*.

DATA

Los datos para este componente son el directorio de ficheros de datos reales tomados de la planta *datafile*, de donde tomaremos la temperatura real de salida del fluido, y la hora de comienzo de la simulación *Tini* en hora decimal. Estos datos son configurables en el editor de atributos del componente.

DECLS

Las variables locales definidas son: *Tsreal* de tipo *REAL* y *t_Tsreal* tipo *TABLE_1D* para la adquisición e interpolación de los datos de temperatura de salida real, y la variable *ITAE* para el cálculo de la Integral del valor Absoluto del Error ponderado en el Tiempo de tipo *REAL* también.

INIT

Inicializamos la tabla *t_Tsreal* mediante la lectura de las columnas 1 y 5 del fichero de datos empleando la función *readTableCols1D*.

CONTINUOUS

Primero, la función *linearInterp1D* realiza la interpolación lineal respecto del tiempo de los valores contenidos en la tabla *t_Tsreal* almacenado el resultado en la variable *Tsreal*. Después se calcula la integral del error *ITAE* a partir de la diferencia entre la temperatura real *Tsreal* y la temperatura del modelo en el puerto de entrada *F_in.T*. Para implementar la ecuación (6), se deriva ésta con respecto del tiempo obteniendo la siguiente ecuación diferencial, la cual se puede escribir directamente en EcosimPro®.

$$\frac{d}{dt}ITAE = (t - t_{ini}) \cdot |Ts_{real} - Ts_{modelo}|$$