PROYECTO FIN DE CARRERA

SIMULINK VS ECOSIMPRO:

COMPARACIÓN METODOLÓGICA Y COMPUTACIONAL

ESCUELA SUPERIOR DE INGENIERÍA

Autor: Juan Martin Navarro Ruiz
Tutor: Manuel Prado Velasco
Fecha: 4-2-2013

Índice Proyecto.

Índice de figuras.

1. Motivación.

El modelado y simulación de sistemas es una herramienta común a diferentes áreas de la ingeniería debido a sus múltiples aplicaciones:

- Facilitan el prototipado de nuevos diseños y la depuración de los mismos antes de alcanzar la fase de fabricación, con la consiguiente reducción de coste y mejora en la calidad del producto.

- Permiten desarrollar simuladores de sistemas, lo que permite modificar variables que en el sistema real o bien no están accesibles o bien no pueden ser modificadas en el rango requerido, también permite el estudio del comportamiento del sistema aislando determinados efecto y eliminando o introduciendo perturbaciones [1].

- Dentro de la teoría de control automático, tanto para el diseño de reguladores que trabajen directamente sobre el sistema físico, como para el desarrollo de observadores matemáticos que permiten la implementación de controladores adaptativos.

- Mejoran el rendimiento de los sistemas físicos, por ejemplo plantas generadoras, trabajando acoplados con los sistemas de supervisión y adquisición de datos (SCADAs).

- Predicción del comportamiento futuro en distintas disciplinas no ingenieriles: economía, meteorología, etc.

Hay que explicar que su aplicación a distintas áreas de la ingeniería viene definido por el tipo de sistemas que analiza y la confianza en el modelado y simulación se puede representar mediante la siguiente figura. En él se muestran varias de las aplicaciones del modelado y simulación de sistemas. Las áreas más oscuras indicarían los sistemas más difíciles de modelar, como es el caso de los sistemas biomédicos y fisiológicos debido a que son sistemas fuertemente realimentados con estructuras multinivel interaccionando a distintos niveles.

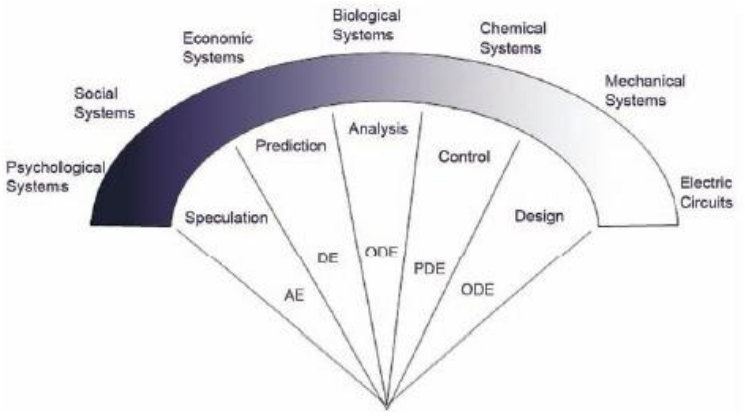


Figura 1.1. Arco iris de Karplus

Décadas de investigación en este área han demostrado la capacidad del modelado y simulación para aportar nuevo conocimiento en el campo, ayudando a entender los mecanismos causales que originan efectos biomédicos observados y facilitando el diagnóstico y tratamiento de pacientes, al incluir los modelos en sistemas de soporte a la decisión médica.

Este avance desde el área de la especulación y el análisis al diseño y control de este tipo de sistemas permanece limitado a los investigadores biomédicos especialistas con conocimientos en modelado y simulación por lo que las herramientas que desarrollan requieren un conocimiento avanzado de las técnicas matemáticas y de las metodologías subyacentes.

Este problema ha adquirido mayor relevancia ante iniciativas que pretenden facilitar la difusión de resultados entre investigadores y profesionales de diferentes países (IUPS physiome project, [2] o Virtual Physiological Human, VPH) y la ágil integración de estos resultados para construir nuevos marcos teóricos que ayuden a validarlos y aplicarlos en los diferentes campos clínicos, desde la asistencias sanitaria a el estudio de nuevos fármacos.

Además, hay que contar con la dificultad de modelado de los sistemas fisiológicos ya que son sistemas multinivel fuertemente acoplados. Este aspecto es clave, ya que requiere por parte de la herramienta informática, la integración de modelos matemáticos referidos a procesos fisiológicos, que pueden o no ser lineales, en un rango que va desde redes intracelulares hasta tejidos, órganos, sistemas de órganos y organismos, independientemente del nivel al cual haya sido formulado [3].

Para ello, se requiere la integración del conocimiento analizado en programas de modelado y simulación. Estos programas de modelado han sido utilizados para distintas experiencias de modelado biomédico y fisiológico, creando una dispersión del conocimiento debido a las distintas caracteristicas de los distintos programas que fuerzan a dividir las metodologías con las que se analizan los sistemas biomédicos y fisiológicos.

En este marco desarrollamos el siguiente proyecto, el cual podemos dividir en dos partes.

Por un lado estudiaremos dos programas de modelado y simulación, EcosimPro y Matlab/Simulink de ámbito general con distintos lenguajes de simulación y sus posibilidades para la integración de modelos biomédicos y fisiológicos.

Este análisis nos proporcionará una visión más clara de cómo debe ser el marco teórico a la hora de desarrollar una metodología de modelado y simulación, cuyas bases deben ser la *reusabilidad*, de forma que integre el conocimiento y la capacidad de *descripción de sistemas multinivel* fuertemente acoplados, como son los sistemas biomédicos y fisiológicos.

Finalmente mostraremos el desarrollo realizado, dentro de un equipo de investigación, para implementar los objetivos metodológicos planteados dentro una librería farmacocinética, LibPK, en EcosimPro en varias versiones: Alfa y Beta, además de las pruebas de concepto realizadas sobre un modelo fisiológico y sobre un modelo farmacocinético.

2. Introducción.

2.1. Revisión de dos herramientas informáticas enfocadas al modelado y simulación de sistemas matemáticos.

Se presentan en este proyecto dos herramientas informáticas enfocadas al modelado y simulación de sistemas matemáticos. Por un lado Simulink/Matlab la cual se encuentra ampliamente extendido tanto en entornos académicos como industriales, y se aplica en multitud de dominios, especialmente vía Simulink, que proporciona una interfaz gráfica la cual permite la implementación de modelos mediante diagramas de bloques.

Por otro lado, EcosimPro es una herramienta mucho más reciente y de origen español. La versión draft 3.0.3 es de Junio de 1999 y la versión 4.8.0 ha salido recientemente. Ecosimpro corre en las distintas plataformas de Windows y usa su propio entorno gráfico para el diseño de modelos.

El modelado de componentes físicos se basa en el Ecosimpro Lenguage (EL), el cual es un lenguaje específico para modelado de sistemas. Tiene un enfoque híbrido: orientado a sistemas continuos, con discontinuidades gobernadas por eventos discretos.

Además de sus metodologías, analizaremos el funcionamiento interno de ambas herramientas y el tratamiento que hacen eventos discretos o lazos algebraicos.

2.1.1 Metodología de Matlab/Simulink.

Matlab es un programa, como ya hemos dicho, de amplia aceptación en ingeniería, utilizado para realizar cálculos técnicos científicos y de propósito general. En el se integran operaciones de cálculo, visualización y programación donde la interacción con el usuario emplea una notación matemática clásica.

Simulink es una aplicación de Matlab que permite construir y simular modelos de sistemas físicos y de control mediante diagramas de bloques. El comportamiento de dichos sistemas se define mediante funciones de transferencia, operaciones matemáticas, elementos de Matlab y señales predefinidas de todo tipo.

Principalmente, se trata de un entorno de trabajo gráfico, en el que se especifican las partes de un sistema y su interconexión en forma de diagramas de bloques. Cada bloque consiste en un conjunto de ecuaciones que se evalúan durante la ejecución del bloque en un diagrama [4].

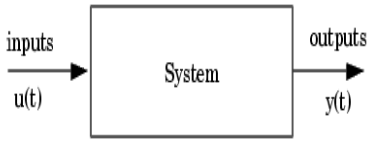


Figura 2.1. Diagrama de un bloque en Simulink.

Simulink trabaja teniendo en cuenta que cada bloque tiene un modelo con ciertas características: Un conjunto de entradas *u*, un conjunto de salidas *y* y un conjunto de estados *x* que se evalúan durante la ejecución del bloque en un diagrama.

El vector de estado puede constar de estados continuos, estados discretos o una combinación de ambos.

Para su evaluación cuentan con tres tipos de funciones ejecutadas por los métodos de bloque.

- Outputs: Calcula las salidas del bloque, dadas:

  Las entradas del paso (step time) actual, y

  Los estados en el paso previo.

- Derivatives: Calcula las derivadas de los bloques de estados continuos en el paso actual, dados:

  Las entradas actuales del bloque, y

  Los valores de los estados en el paso previo

- Update: Calcula el valor de los estados de los bloques discretos en el paso actual, dados:

  Las entradas actuales del bloque, y

  Los valores de los estados discretos en el paso previo.

$$y = f_0(t, x, u) \qquad \text{(Output)}$$

$$\dot{x}_c = f_d(t, x, u) \qquad \text{(Derivative)}$$

$$x_{d_{k+1}} = f_u(t, x, u) \qquad \text{(Update)}$$

where $x = x_c + x_d$

Figura 2.2. Funciones ejecutadas por los métodos de bloque en Simulink.

Este tipo de lenguaje de simulación basado en diagramas de bloques requieren como **primer paso** para el usuario un análisis profundo de las ecuaciones del modelo que pretende implementar y una manipulación del mismo por parte del usuario que debe:

Observar si se dispone de tantas ecuaciones como variables.

Si es así, intentar despejar de cada ecuación una de las variables.

- Las variables que aparecen en una única ecuación se despejan de dicha ecuación.
- Las variables que aparecen derivadas se calculan por la integración numérica (despejamos la derivada de las variables)

Pueden aparecer problemas en la manipulación de las ecuaciones que pueden requerir actuaciones más complejas.

El **segundo paso** consiste en la edición del diagrama de bloques de las ecuaciones manipuladas del modelo. Para ello se nos ponen a disposición multitud de bloques de cálculo con causalidad computacional predefinida (entradas y salidas) de distintos tipos (integradores, sumadores, limitadores...) además de bloques personalizables existen asimismo bloques virtuales, los cuales solo tienen un papel para la organización gráfica (demux) por lo que no juegan ningún papel en la definición de las ecuaciones descritas por el modelo.

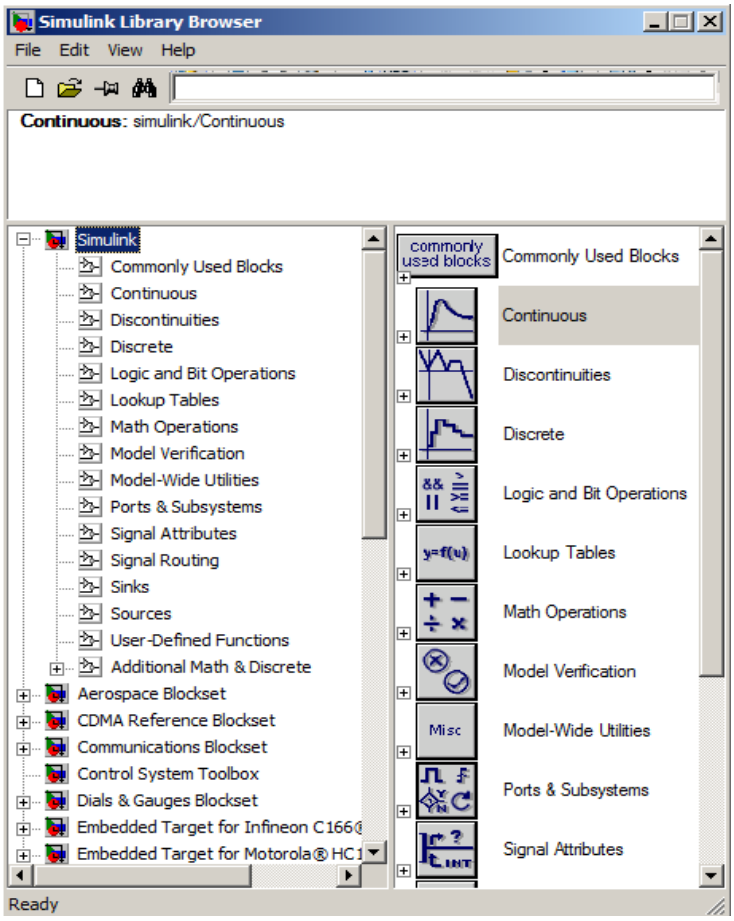


Figura 2.3. Interfaz de Simulink para la creación de modelos.

Después de haber construido el modelo de SIMULINK de un sistema y antes de ejecutar la simulación se debe seleccionar un método de integración y determinar las condiciones de ejecución. Esto se hace en el llamado Solver de Simulink. Hay dos métodos distintos seleccionables: Paso fijo o Paso variable.

1.- Paso Fijo: El solver de paso fijo resuelve el modelo en intervalos regulares de tiempo desde el principio al final de la simulación. El tamaño del intervalo es llamado tamaño de paso. Puede ser elegido por el usuario o calculado por el solver. Generalmente reducir el tamaño de paso incrementa la exactitud del resultado mientras aumenta el tiempo requerido para simular el sistema.

2.- Paso Variable: El solver de paso variable precisamente varía el tamaño de paso, reduciéndolo e incrementando su resolución cuando los estados del modelo cambian rápidamente y aumentando el tamaño de paso para evitar pasos innecesarios cuando los estados del modelo cambian lentamente. Calcular el tamaño de paso añade carga de computación pero puede reducir el número

total de pasos de la simulación. Además es necesario para mantener un determinado nivel de exactitud para modelos que cambian rápidamente o continuos a trozos.

Simulink provee de ambos métodos de integración continuo o discreto.

1.- Solver Continuo: Usa integración numérica para calcular los estados continuos del modelo con una escala de tiempo basada en los estados en los anteriores pasos y sus derivadas. El solver continuo se basa en los bloques individuales para calcular los valores de los estados del modelo en cada paso de integración.

Se han desarrollado numerosas técnicas para resolver la integración numérica de ecuaciones tipo ODEs[1] que representan los estados continuos de un sistema dinámico. Entre los tipos de resolución se mencionan los siguientes:

- **Ode45:** Este método está basado en Dormand-Prince, el cual es un método explicito de un paso de Runge-Kutta, este método es el recomendado por defecto.
- **Ode23:** Este método está basado en Bogacki-Shampine, el cual también es un método explicito de un paso de Runge-Kutta. Puede ser más eficiente que ode45 cuando las tolerancias son amplias
- **Ode113:** Este es un método multipaso de orden variable de Adams-Bashforth-Moulton. Es recomendado cuando la evaluación de la función toma demasiado tiempo y las tolerancias son estrechas.
- **Ode23s:** Este método es de un paso basado en la fórmula de Rosenbrock de segundo orden.

2.- Solver Discreto: Existe básicamente para resolver modelos discretos. Se calcula el siguiente paso de un modelo y nada más. No calcula los estados continuos y se basa en bloques del modelo para actualizar los estados discretos del modelo.

[1] Ecuaciones Diferenciales Ordinarias.

Tratamiento matemático.

Una vez realizado el modelo, aportados los parámetros a los bloques de cálculo y una vez sabemos cómo queremos que sea integrado, se compila el modelo. Esto arranca la simulación en Simulink, la cual, podemos decir que se realiza en tres fases:

- Compilación del modelo
- Fase de enlace
- Bucle de simulación

La fase de compilación:

La fase de compilación se inicia cuando se oprime **Start.** Esto causa que el engine (motor) de Simulink invoque al compilador el cual convierte al modelo en una forma ejecutable.

El proceso denominado compilación consiste en:

-Evaluar las expresiones de los parámetros de los bloques del modelo para determinar sus valores.

-Determinar los atributos de las señales. Nombre, tipo de dato, dimensionalidad

-Chequear que cada bloque pueda aceptar las señales conectadas a sus entradas.

-Realizar una reducción óptima de los bloques.

-Llevar a un mismo nivel la jerarquía del modelo, remplazando los subsistemas virtuales con los bloques que contienen.

-Determinar el orden (*sorted order*) de los bloques.

-Determinar los tiempos de muestreo de todos los bloques.

La fase de enlace:

En esta fase, el Motor de Simulink:

Asigna la memoria necesaria para las áreas de trabajo necesarias para la ejecución del diagrama de bloque: Señales, estados, y parámetros de tiempo de ejecución.

Asigna e inicializa la memoria para las estructuras de datos que almacenan

la información en tiempo de ejecución para cada bloque. Para los bloques de Simulink **SimBlock** es la principal estructura de datos en tiempo de ejecución.

En esta estructura se almacenan:

- Los pointers a los buffers de entrada y salida del bloque.

- Los vectores de estado.

- Los vectores de trabajo.

En esta fase, el motor de Simulink crea las **listas de ejecución de los métodos.**

Estas listas, listan el orden mas eficiente de invocar los métodos de los bloques del modelo para calcular sus salidas

Simulink usa el orden de los bloques generado durante la fase de compilación para construir las listas de ejecución de los métodos.

La fase de bucle de simulacion:

En esta fase, el motor de Simulink calcula sucesivamente, desde el tiempo de inicio al tiempo de fin de la simulación:

- Los estados y las salidas del sistema a intervalos de tiempo.

La fase de bucle de Simulación tiene dos subfases:

- La fase de inicialización. Ocurre una vez, en el principio del lazo.

- La fase de iteración. Se repite una vez en cada **step time**, desde el inicio (**start**) hasta el final (**stop**) de la simulación.

Al inicio de la simulación (**start**), el modelo especifica los estados iniciales y las salidas del sistema a ser simulado.

En cada paso (**step time**), se calculan nuevos valores para, las entradas del sistema, los estados, y las salidas y actualiza el modelo para reflejar los valores calculados.

En cada paso el motor de Simulink:

1.-Calcula las salidas del modelo.

2.-Calcula los estados del modelo (discretos o continuos).

3.-Chequea si existen discontinuidades en los bloques continuos.

4.-Calcula el tiempo para el siguiente paso.

Simulink repite los pasos 1 a 4 hasta alcanzar el tiempo final (stop).

1.-Calculo de las salidas del modelo.

El motor inicia este paso invocando el método Outputs del modelo. El

método Outputs del modelo a su vez invoca al método Outputs del sistema. El método Outputs del sistema a su vez invoca a los métodos Outputs de los bloques en el orden especificado por las listas de ejecución de los métodos de salida generadas en la fase de enlace

El método Outputs del sistema pasa los siguientes argumentos a cada método de Outputs de bloque:

-Un pointer de la estructura de datos del bloque

-Un pointer de su estructura SimBlock.

La estructura de datos SimBlock apunta a la información que el método Outputs necesita para calcular las salidas del bloque, incluyendo la posición de los buffers de entrada y de salida.

2.-Calculo de los estados discretos del modelo

El motor inicia este paso invocando al solver discreto seleccionado .

El solver calcula el tamaño del paso necesario e invoca al método Update del modelo, el método Update del modelo invoca al método Update del sistema, el método Update del sistema invoca a los métodos Update de los bloques.

2.-Calculo de los estados continuos del modelo

El motor inicia este paso invocando al solver discreto seleccionado

Dependiendo del solver, el solver invoca una sola vez el método Derivatives del modelo o entra en un subciclo de pasos de tiempo menores donde el solver repetidamente invoca los métodos Outputs y Derivatives del modelo para calcular salidas y las derivadas del modelo en intervalos sucesivos dentro del paso de tiempo principal.

Los métodos Output y Derivatives del modelo invocan a sus métodos de sistema correspondientes

Los métodos Output y Derivatives del del sistema invocan a sus métodos de bloque correspondientes

En el orden especificado por las listas de ejecución de los métodos de salida generadas en la fase de enlace

3.- Chequea si existen discontinuidades en los bloques continuos(opcional)

Una técnica llamada Zero-Crossing Detection se usa para la detección de discontinuidades en los estados continuos.

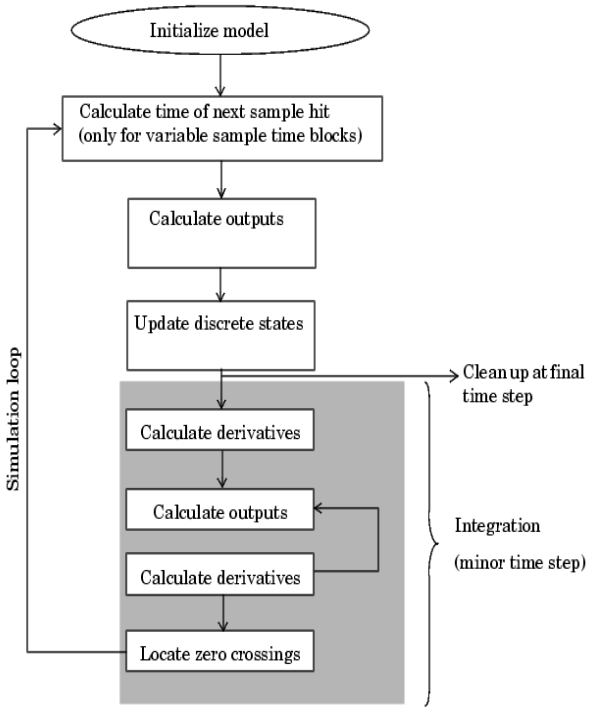


Figura 2.4. Diagrama de la estructura de cálculo de Simulink.

Detección de eventos

Cuando simulamos un sistema dinámico, Simulink comprueba las discontinuidades en el sistema de variables de estado para cada paso de integración, usando una técnica llamada Zero-Crossing Detection. Si Simulink detecta una discontinuidad, determina el tiempo preciso en el cual ocurre la discontinuidad y toma un tiempo adicional antes y después de la discontinuidad.

Las discontinuidades en las variables de estados coincide normalmente con un evento significativo en la evolución del sistema dinámico. Por esto es importante averiguar el punto exacto de discontinuidad ya que sino puede llegar a falsas conclusiones sobre el sistema.

Para evitar conclusiones erróneas es importante que el paso de simulación esté exactamente en el punto de discontinuidad. Para métodos de paso fijo por ejemplo no hay garantía de que el punto de discontinuidad ocurrirá en un paso elegido.

Los métodos de paso variable si ofrecen una solución a este problema, ya que el tamaño de pasos se ajustará en el entorno de la discontinuidad para su detección con precisión. El problema es que para localizar la discontinuidad con exactitud debe tomar muchos pasos pequeños ralentizando la simulación.

Simulink usa la técnica de Zero-Crossing Detection para solucionar el problema. Con esta técnica un bloque puede registrar un grupo de variables

zero-crossing para actualizar la variable. Simulink comprueba entonces si alguna variable a cambiado de signo desde el último paso, ya que un cambio indica que la discontinuidad ocurre en el paso actual.

Si un zero-crossing es detectado. Simulink interpola entre valor anterior y actual de cada variable que cambió de signo para estimar el tiempo exacto de la discontinuidad. Luego pone un paso antes y después de la discontinuidad de forma que se evita simular exactamente la discontinuidad donde el valor del estado será indefinido.

De esta manera puede simular discontinuidad sin usar demasiados pequeños pasos. Muchos bloques de Simulink tienen la Zero-Crossing Detection de forma intrínseca, si se necesita una notificación explícita se usa un bloque específico llamado Hit Crossing Block.

Figura 2.5. Bloque Hit Crossing de Simulink.

La salida del bloque vale la unidad en el instante en el que sucede el evento, el resto del tiempo su valor es cero. Puede distinguir el cruce por el valor asociado al evento en ambas direcciones, en dirección positiva y negativa.

Lazos algebraicos

Una vez realizada la simulación puede darse la situación de que se haya constituido un **lazo algebraico**, los lazos algebraicos o implícitos ocurren cuando dos o más bloques con alimentación directa de sus entradas forman un lazo de realimentación. Cuando esto ocurre, Simulink debe efectuar iteraciones en cada paso para determinar si existe una solución a este problema.

Estos bloques con alimentación directa implican que la salida del bloque depende del valor del puerto de entrada, por ejemplo los bloques Gain, Product y Sum. También hay bloques con alimentación no-directa, por ejemplo el bloque Integrator (su salida es una función del estado), el bloque Constant (no tiene entrada) y el bloque Memory (su salida depende de la entrada en el paso anterior).

Un ejemplo sencillo de realimentación puede ser este lazo algebraico:

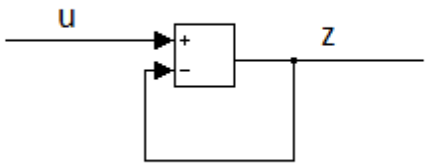


Figura 2.6. Ejemplo de Lazo algebraico

Matemáticamente, el lazo algebraico implica que la salida del bloque Suma es un estado z forzado a ser igual que la entrada u menos z (i.e. z=u-z). La solución de este lazo simple es z=u/2, pero muchos lazos algebraicos no pueden ser resueltos a partir de una simple inspección.

Por ejemplo analizaremos este sistema con múltiples variables de estado z1 y z2.

z2=1-z1; z1=z2-1. z2=1, z1=0.

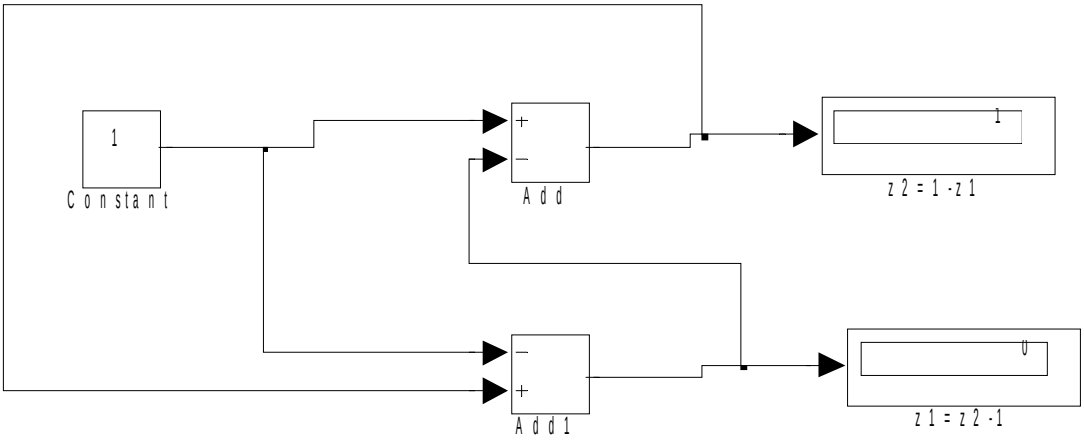


Figura 2.7. Ejemplo de Lazo algebraico implementado en Simulink

El programa nos advierte del lazo algebraico, una forma de evitarlo sería un cambio en la forma de crear el sistema usando un bloque especial que resuelve ecuaciones implícitas.

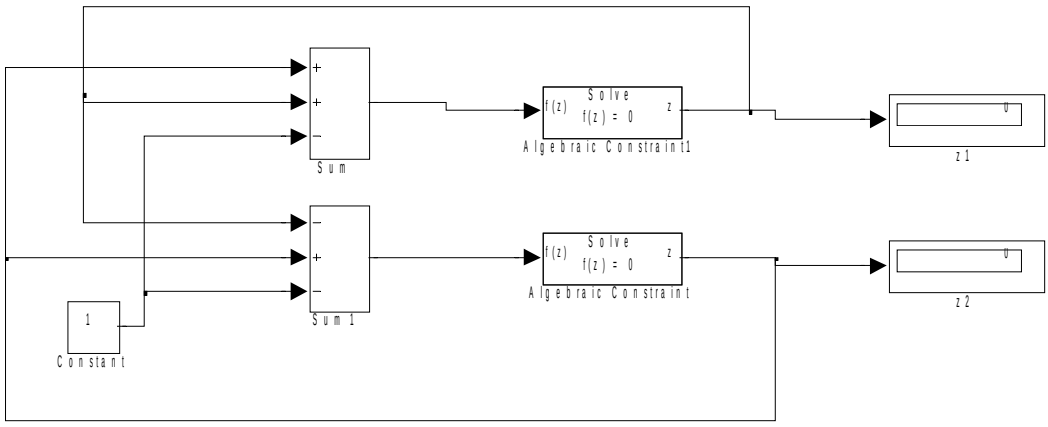


Figura 2.8. Ejemplo de Lazo algebraico implementado en Simulink

El bloque “Algebraic Constraint” es una manera conveniente de modelar las ecuaciones algebraicas y especificar valores iniciales. Ya que este bloque iguala a su señal de entrada F(z) a cero consiguiendo como salida el estado algebraico z.

La salida del bloque tendrá el valor necesario para producir un cero a la entrada. Se puede proporcionar un valor inicial para mejorar la eficiencia del solver al resolver el bucle.

Un lazo algebraico representa una ecuación algebraica escalar o restricción de la forma F(z)=0, donde z es la salida de uno de los bloques del lazo y la función F consta de la realimentación a través de otros bloques del lazo a la entrada del bloque. Las ecuaciones serían:

z2+z1-1=0

z2-z1-1=0

Esta restricción debe surgir como consecuencia de la interconectividad del sistema que se modela, o debe surgir a causa de un modelo DAE[2]

Cuando un modelo contiene un lazo algebraico, Simulink usa un solver de bucle en cada paso de integración (step time). El solver de bucle interpreta iteraciones para determinar la solución al problema (si se puede). Como resultado, modelos con lazos algebraicos funcionan más lentamente que un modelo sin ellos.

Simulink puede eliminar algunos lazos algebraicos que incluyan alguno de los siguientes tipos de bloques:

- Subsistemas atómicos.

- Subsistemas permitidos/preparados.

- Models.

En la figura 2.9 se muestra un diagrama con las opciones que debe seguir un usuario cuando el modelo construido contenga un lazo algebraico.

[2]Ecuaciones Diferenciales Algebraicas

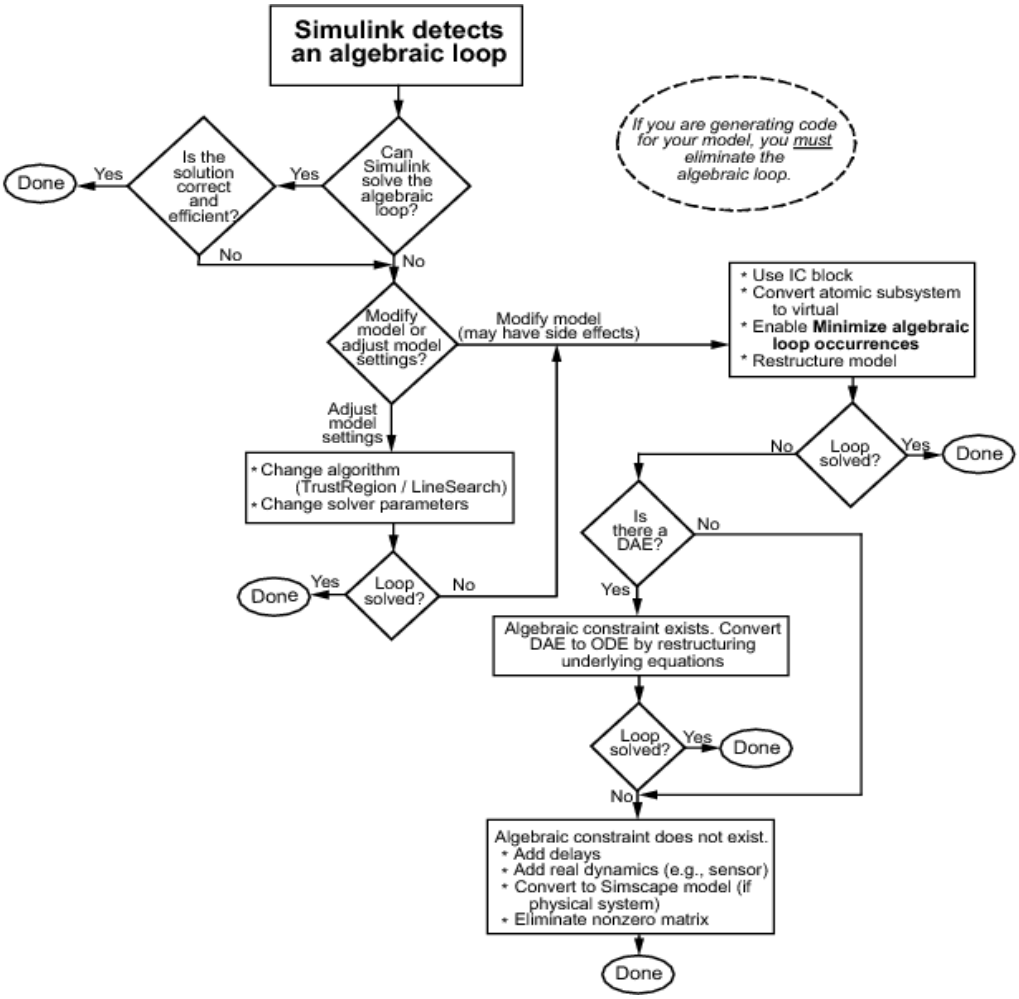


Figura 2.9. Diagrama de flujo con los pasos a seguir por el usuario para eliminar un lazo algebraico en Simulink.

2.1.2 Metodología de EcosimPro.

EcosimPro es una herramienta de modelado y simulación de sistemas continuos y discretos desarrollada por Empresarios Agrupados (EA) durante los últimos 19 años.

Dispone de su propio lengua de simulación llamado EL (EcosimPro Lenguaje) basado en el modelado orientado a objeto el cual integra las últimas tecnologías para el modelado acausal de sistemas representables por ecuaciones algebraico-diferenciales y eventos discretos [5].

Presenta tres modos de vista para el modelador diferenciando el diseño del modelo y por otro la simulación de modelos:

- **Code view**: Permite introducir, leer, modificar, etc. el código de los componentes elaborados.
- **Schematic view**: Interfaz gráfica con librerías de símbolos para la fácil creación de modelos esquemáticos a partir de objetos ya definidos o ya creados por el usuario que se conectan entre sí tal y como están conectados en el sistema físico real.
- **Simulation view**: Se trata de la vista de simulación, y hay que acceder a ella cuando ya se ha creado el modelo mediante alguna de las opciones que el entorno permite (a partir de componentes textuales o a partir de componentes gráficos).

También permite la generación de código C++ (modelos compilados) además de tener interfaces con programas externos: funciones Fortran, Excel, Active X, funciones C, Matlab/Simulink etc.

Tal y como hemos visto EcosimPro tiene un lenguaje orientado a objeto. Por tanto la descripción de modelos se hace a través de la unión de objetos.

El objeto es una descripción teórica de una estructura y un comportamiento de un componente físico. Sus ecuaciones dinámicas (en el caso de que existan) y sus eventos discretos (si los hay) son incluidos en esta descripción. EcosimPro los llama Componentes . Es el bloque más elemental de EcosimPro y es el concepto equivalente de clase en programación.

La descripción del componente consta de distintas secciones:

- La sección DATA. Donde se incluyen los datos que EcosimPro tratará como conocidos,
- La sección DECLS. Donde se definen las variables de entrada y salida del modelo, por lo que o bien son variables dinámicas que el resolvedor de ecuaciones de EcosimPro deberá calcular o bien son

variables de contorno que el usuario deberá especificar en el experimento.

- La sección DISCRETE. En esta sección se incluye aquellas partes del modelo que se ejecutan sólo en determinados momentos de la simulación. Cuando una variable cumple una condición determinada, cuando el tiempo vale un determinado valor (eventos temporales), que a su vez pueden ser periódicos o no.
- La sección CONTINUOUS. En esta sección se incluye las ecuaciones que implican variables que cambian de forma continua a lo largo del tiempo. En este punto es importante recordar que el lenguaje EL es acausal, por lo que no hay que tener en cuenta el orden en el que se haya declarado ni la causalidad que les asigne.
- La sección SEQUENTIAL. Se declara dentro de CONTINUOUS, el código de su interior se ejecuta de manera totalmente secuencial.

La unión entre objetos se hace a través de lo que en EcosimPro llaman Puertos, que encapsula un conjunto de variables que se intercambian juntas, esto facilita el modelado de sistemas complejos pues no requiere trabajar a nivel de variable. Estos conjuntos vienen definidos en general por el tipo de modelo que realicemos, puertos eléctricos, hidráulicos etc.

Así pues un componente se define por:

- Los puertos de entrada y salida.

- La declaración de datos y variables.

- Las ecuaciones que representan el comportamiento.

Los componentes y puertos se almacenan dentro de lo que en EcosimPro se llama Librería, las cuales vienen definidas para distintas áreas de conocimiento: Eléctrica, Hidráulica, Térmica etc.

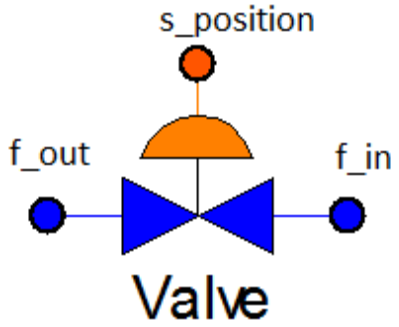


Figura 2.10. Ejemplo de un elemento con sus puertos en EcosimPro.

Para la creación de modelos los MOO[3] han heredado las características propias de los lenguajes de programación orientados a objetos (POO[4]), en los cuales se basan, de forma que han impuesto una nueva visión en la creación de modelos, con una sintaxis que les permite ser más estructurados, más flexibles, estar basados en los principios de parametrización, reutilización, instanciación, encapsulación...[6]

La simulación y el modelado orientados a objetos proporcionan las suficientes características para tratar la complejidad sin afectar al "exterior" (**encapsulación**), habilitan la reutilización de modelos (**herencia** y **agregación**) y permiten crear modelos independientes que sean fáciles de mantener. Además, el desarrollo modular permite modelar un sistema desde abajo a arriba (bottom-up), programando primero los componentes más básicos para, a partir de ellos, ir desarrollando otros más complejos.

Las librerías básicas de componentes se pueden combinar para crear componentes cada vez más complejos utilizando dos métodos:

- Extensión por herencia, a partir de componentes existentes.

- Instanciación y conexión de componentes existentes.

Estas ideas se aplican para crear componentes que representen un sistema completo. Los componentes intermedios también pueden ser simulados. Esto reduce ampliamente los tiempos de mantenimiento y desarrollo.

Uno de los objetivos del modelado es encapsular la complejidad. Consiste en separar los aspectos externos de los objetos, a los cuales pueden acceder otros objetos, de los detalles internos de la implementación del mismo, que quedan ocultos a los demás. Las variables locales, su topología interna, la inicialización, la parte continua y discreta constituyen la parte privada, mientras que los parámetros, puertos y datos constituyen la parte pública, visible, el interfaz de un componente.

La herencia es lo que da a los lenguajes de modelado orientado a bloques su capacidad para compartir interfaces y comportamientos. Los lenguajes de MOO pueden reunir datos y ecuaciones comunes en unos componentes padre, que serán heredadas por los componentes hijos. La herencia es un mecanismo para compartir información entre componentes mediante la especialización.

---

[3] Modelado Orientado a Objeto.

[4] Programación Orientado a Objeto.

Generalmente, los lenguajes de MOO consideran que si los valores de los atributos de un componente entran en contradicción con los heredados de su supercomponente prevalecen en la clase los valores locales. Esta propiedad es una forma de parametrización, ya que supone la modificación de algunas de las propiedades para adaptar el modelo a sus diferentes aplicaciones. Para facilitar este mecanismo en algunos entornos existen las denominadas ecuaciones virtuales, que pueden ser modificadas por el subcomponente sin que se produzcan conflictos.

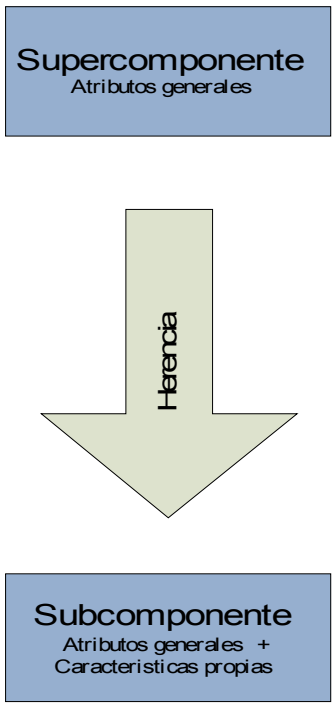


Figura 2.11. Diagrama explicativo del mecanismo de herencia

El polimorfismo es un concepto ligado estrechamente a la reutilización de modelos. Modelos polimórficos son aquellos que tienen interfaces con estructuras equivalentes y que además, tienen los mismos grados de libertad, con lo cual pueden ser usados en el mismo contexto e intercambiados sin necesidad de alterar el resto del sistema. Frecuentemente los modelos polimórficos poseen un mismo supercomponente común: aquel en el que se define la interfaz.

La agregación es una forma fuerte de asociación en la cual el objeto está formado por componentes. Un componente puede heredar de otro u otros y

puede tener múltiples instancias o copias de otros internamente. Este hecho refuerza la característica de la reutilización. Este paradigma se aplica de forma iterativa y sin límites, de manera que la complejidad de un componente final puede estar oculta por esta agregación de instancias internas a otros componentes ya comprobados.

La abstracción consiste en centrarse en los aspectos esenciales inherentes a una entidad e ignorar sus propiedades accidentales. En el desarrollo de un modelo esto significa centrarse en sus características, lo que es y lo que hace, antes de decidir como debería implementarse. La abstracción es un modo de afrontar el problema de la complejidad de los sistemas de grandes proporciones, posibilitando que varios especialistas puedan trabajar sobre distintas partes de un mismo modelo sin tener que conocer todos los detalles del resto del mismo. Una forma de facilitar la abstracción consiste en diferenciar, en cada parte del modelo, entre la interfaz y la descripción interna.

Las librerías son los elementos básicos con los que los lenguajes de MOO organizan la información que manipulan. El propósito básico de una librería es almacenar un conjunto de elementos relacionados entre sí. EcosimPro incluye varias librerías que pueden ser modificadas por el usuario. La potencia de un entorno está en el número y la calidad de sus librerías. Además de mantener la tarea de modelado bien organizada, las librerías proporcionan al entorno de modelado otros servicios como:

-Comprobar elementos obsoletos (componentes, puertos, datos, etc.) que deben recompilarse.

-Encapsular los elementos, permitiendo a diferentes librerías contener elementos con el mismo nombre.

Tratamiento matemático.

A la hora de realizar una simulación, EcosimPro no codifica las ecuaciones tal y como las escribe el usuario, sino que realiza una serie de transformaciones para escribir el sistema de una forma más conveniente.

Una vez generadas las ecuaciones, el siguiente paso es decidir en qué orden se resuelven las variables del modelo. Aunque desde el punto de vista del usuario final EcosimPro es un lenguaje acausal, está claro que para realizar cualquier simulación con un modelo es necesario convertir el conjunto de ecuaciones escritas por el usuario en otro conjunto de ecuaciones escritas de forma secuencial, donde cada línea contenga variables que hayan sido previamente calculadas. EcosimPro implementa estas ecuaciones en un lenguaje causal, en este caso C++. Este proceso se denomina asignación de la causalidad.

En EcosimPro se describe el sistema por medio de ecuaciones, posteriormente el compilador decide cuál es el orden adecuado para cada ecuación y la variable que resuelve.

En cuanto a las variables, existen los siguientes tipos:

- Las variables tipo dato (DATA_VAR) son aquellas que se han definido en la sección DATA de los componentes, han sido inicializadas antes de crear la partición y por tanto su valor no necesita ser calculado en tiempo de simulación debido a que permanecen constantes a lo largo del tiempo.
- Las variables explícitas (EXPLICIT) necesitan obtenerse a partir de las ecuaciones de los componentes, y por tanto las ecuaciones deben ordenarse de tal forma que todas ellas sean calculadas por una y solo una ecuación.
- Las variables de estado (DYNAMIC) son variables que en algún momento aparecen derivadas en el modelo y su valor se obtiene integrando el mismo por un algoritmo de integración.
- Las variables derivadas (DERIVATE) son las complementarias de las variables de estado y expresan la derivada respecto a la variable independiente del sistema (normalmente el tiempo). Estas variables se calcularán de forma explícita.
- Las variables algebraicas (ALGEBRAIC) son variables que no han podido ser calculadas por métodos explícitos y se resuelven por un resolvedor implícito de forma iterativa. Aparecen en todos los lazos algebraicos no lineales del modelo y se calculan de forma implícita.
- Las variables de contorno (BOUNDARY) son variables continuas (tipo REAL) que el usuario ha decidido que serán condiciones de contorno y no son calculadas internamente. La diferencia con los datos es que pueden ser dadas como variables dependientes del tiempo y no sólo como una constante.
- Las variables discretas (DISCRETE) son todas las variables que no sean de tipo REAL (es decir son INTEGER, BOOLEAN, STRING, O FILEPATH) y las etiquetadas como DISCR REAL. Son variables que EcosimPro no tratará de calcularlas nunca del sistema de ecuaciones y son responsabilidades del modelador cambiarlas de forma discreta en el modelo (típicamente en los bloques INIT y DISCRETE).
- Parámetros de construcción. Son parámetros usados en componentes y puertos en el momento de su instanciación. Son usados para dimensionar arrays o conjuntos de ecuaciones y no

pueden ser modificados en ningún momento después. Los datos (DATA_VAR) aunque permanecen constantes durante los cálculos, pueden ser modificados entre dos cálculos, los parámetros de construcción no.

El Jacobiano del modelo lo forman las variables de estado (DYNAMIC) y las variables algebraicas (ALGEBRIC). Los métodos numéricos para integrar el modelo calculan los valores de las variables de estado y algebraicas. Para que la integración numérica pueda llevarse a cabo es necesario dar al integrador el valor inicial de las variables de estado y algebraicas (normalmente en el bloque INIT del experimento). Las ecuaciones de residuos guían al integrador son generadas automáticamente por EcosimPro en base a transformaciones simbólicas del bloque ecuaciones global.

Habitualmente los sistemas de simulación presentan sistemas de ecuaciones que tienen que ser resueltos en poco tiempo. Otras veces, no es posible obtener la solución exacta. La aplicación de métodos numéricos subsana estos dos problemas.

-Para la resolución de sistemas lineales se emplea el método de factorización LU.

-Para la resolución de sistemas no lineales se emplea el método de Newton-Raphson basado en iteraciones que partiendo de un punto anterior calculan el siguiente. El punto de inicio debe darse cerca de la solución para que ésta converja.

Por su parte los métodos de integración tienen como propósito, dado un problema de valor inicial, calcular una aproximación de la solución en una serie de puntos, de tal forma que la función quede suficientemente determinada para el estudio que se pretende.

El método de integración más clásico para resolver ecuaciones diferenciales ordinarias (ODE) es el Runge-Kutta. El método DASSL es más moderno y está orientado a la resolución de sistema de ecuaciones algebraico-diferenciales (DAE). A diferencia del Runge-Kutta, en el que la derivada tenía que estar de forma explicita, en este método no es necesario. El sistema puede estar escrito de forma más general.

Los métodos de integración utilizados en EcosimPro son: Runge-Kutta de orden 4 y DASSL. Por defecto el método de integración es DASSL, pero se puede seleccionar cualquier otro.

Los pasos en el procesado matemático de EcosimPro son:

1. Solución simbólica de ecuaciones lineales de coeficientes constantes.
2. Detección y corrección de problemas de alto índice. Estos problemas surgen cuando aparecen variables de estado (DYNAMIC)

emparejadas o relacionadas entre sí. El resolvedor matemático está diseñado, normalmente, para calcular el valor de las variables de estado integrando su derivada, asumiendo que estas variables son independientes unas de otras.

Existen distintas técnicas para resolver problemas de alto índice que van desde la manipulación simbólica hasta el uso de resolvedores especiales.

EcosimPro usa dos tipos diferentes de algoritmos de manipulación simbólica: solución por eliminación de las restricciones lineales o solución por derivación de ecuaciones.

3. Selección de las variables de contorno. Para ello se clasifican primero las variables en conocidas y no conocidas.
4. Aplicación del máximo algoritmo transversal y clasificación de las ecuaciones. El propósito de este algoritmo consiste en encontrar una variable que pueda ser calculada en cada ecuación. En el campo de las matrices, el problema es conocido como “encuentra el máximo número de no-nulos en la diagonal de la matriz”, mientras que en la teoría de grafos bipartitos se conoce como “perfect matching”.

El objetivo de la clasificación de ecuaciones es permitir a EcosimPro calcular a las variables de forma secuencial.

5. Solución de lazos algebraicos, lineales y no lineales.
6. Resolución numérica de sistemas diferenciales y algebraicos. Finalmente, después de todos los pasos preliminares, el problema se formula como un sistema de ecuaciones algebraico-diferenciales o DAE. La mayoría de las aplicaciones de simulación formulan el problema como un sistema de ecuaciones diferenciales ordinarias o ODE. Por lo tanto una ODE puede ser escrita de forma explícita como una DAE de la siguiente forma.

$$y'(t) - f[t, y(t)] = 0$$

Como ya se ha mencionado, EcosimPro utiliza el método DASSL para esta resolución.

Detección de eventos

Dentro del lenguaje EL, al ser un lenguaje de modelado híbrido, hay diferentes sentencias para el tratamiento de eventos además de una sección específica (DISCRETE) en la definición de cada componente.

Para la detección de eventos EcosimPro utiliza el siguiente algoritmo:

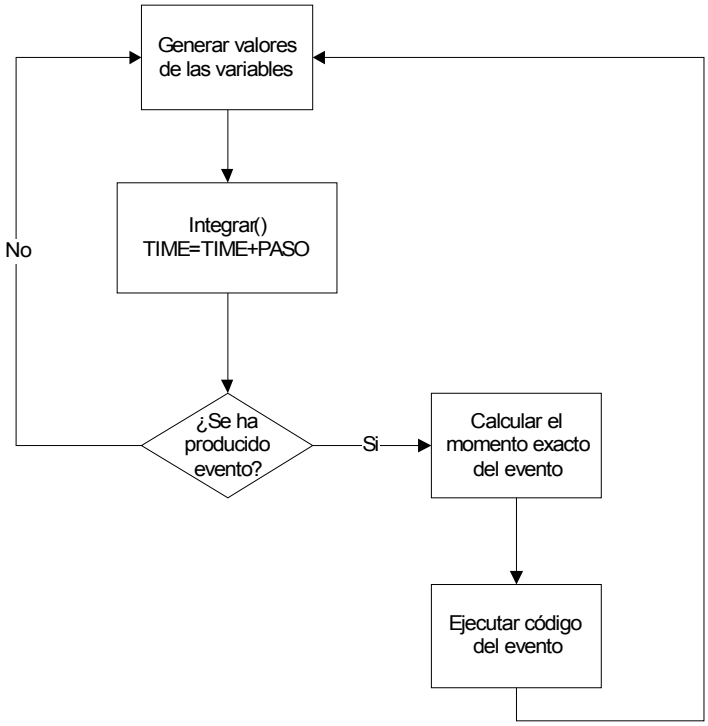


Figura 2.12. Diagrama del algoritmo usado por Ecosimpro para la detección de eventos.

Para detectar el evento, EcosimPro asocia a cada condición una variable de control, si esta cambia de sentido se ha producido un evento.

Una vez lanzado el evento, el siguiente paso es determinar el momento exacto en el cual se produce, para ello utiliza un algoritmo de búsqueda de raíces hasta encontrar la raíz (el valor de TIME) con una precisión determinada.

Lazos algebraicos

Se da cuando EcosimPro detecta variables que no se pueden despejar explícitamente. Pueden ser lineales o no.

-.Lazos Lineales: Para simular este ejemplo se crea el componente *loop*.

```
COMPONENT loop
   DATA
         REAL PI=3.1415926
   DECLS
         REAL x
         REAL u
         REAL v
   CONTINUOUS
         x'=8*cos(2*PI*TIME)
         u+x+v=TIME
         2*u-7*v=6
END COMPONENT
```

Figura 2.13. Componente loop desarrollado en EcosimPro.

Veamos la causalidad asignada por EcosimPro.

SORTED EQUATIONS:

###eqts
[1] x' = 8. * cos(2. * PI * TIME) {E@x'@@}

BOX 1 IS LINEAR

Variables:
u
v

Equations:

[2] u + x + v = TIME {L@u@@1}
[3] u = 3. + 3.5 * v {L@v@@1}

END OF BOX 1

###endEqts

Figura 2.14. Sistema de ecuaciones asignado por Ecosimpro.

La ecuación *"[3]"* del bloque ha sufrido una manipulación algebraica, transformándose en otra equivalente, donde se ha despejado u. El lazo de este

ejemplo es lineal.

Para resolver este lazo plantea un sistema matricial que resuelve numéricamente en cada paso de integración.

$$\begin{pmatrix} 1 & 1 \\ 1 & -3.5 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} TIME - x \\ 3 \end{pmatrix}$$

- Lazos no lineales: Los lazos no lineales son más costosos en tiempo de cómputo que los lineales. Además necesitan procedimientos más complejos. Ilustraremos como trata EcosimPro la resolución de este tipo de lazos mediante un ejemplo. Se crea el componente *loopnl.*

```
COMPONENT loopnl
   DECLS
         REAL x
         REAL y
         REAL z
   CONTINUOUS
         x'+8*x=7
         3*exp(y)+7*z=x
         4*y-8*exp(-z)=1
END COMPONENT
```

Figura 2.15. Componente loopnl. desarrollado en EcosimPro.

Al ser un sistema no lineal EcosimPro proporciona un asistente para que el usuario elija la variable de *tearing* o variable algebraica.

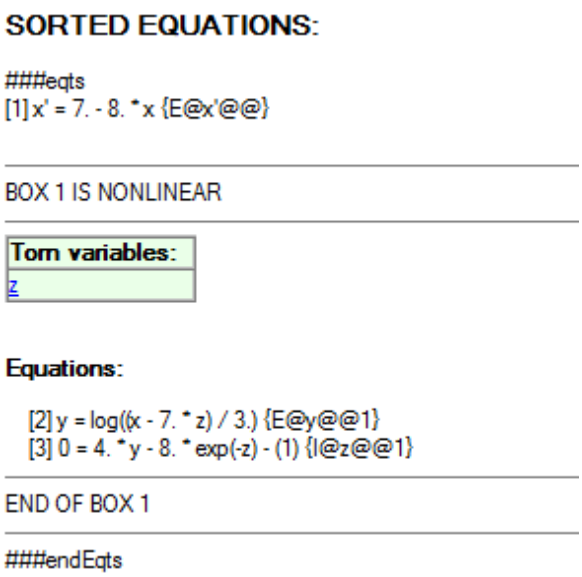


Figura 2.16 Sistemas de ecuaciones asignado por EcosimPro.

Se observa que las ecuaciones "*(2)"* y "*(3)"* forman el lazo no lineal, que la ecuación *"(2)"* ha despejado la variable *"y"* por medio de manipulaciones algebraicas y que la ecuación "*(3)"* esta en forma de residuo: *f*(variables)=0.

Esta última ecuación recibe como entrada "*z"*, la variable elegida como algebraica o de *tearing*, "y" da como salida el valor del residuo de la ecuación "*(3)".* A continuación, por medio del método de Newton halla el valor de "*z"* que anula a *f*().

La ventaja que aporta este sistema con respecto al método de Newton clásico se debe a que al escoger una variable de *tearing* se ha logrado reducir el sistema no lineal en el que para resolverlo habría que aplicar el método de Newton a una función de R^2 en R^2 a una función de R en R.

3. Objetivos.

Bajo estas premisas, se plantea como objetivo del siguiente proyecto el análisis metodológico, numérico y de uso de recursos de las dos herramientas informáticas expuestas para la descripción de un sistema fisiológico. Además se realizan aportaciones al desarrollo de una librería farmacocinética en el marco de un proyecto de investigación.

Ambas herramientas presentan similitudes, como el particionado del sistema físico en bloques que pueden manejarse gráficamente. Sin embargo Simulink/Matlab utiliza una metodología de diagrama de bloques, mientras EcosimPro usa un lenguaje de modelado a objeto, EL.

En base a todo lo expuesto, establecemos los objetivos concretos del proyecto:

- Revisión de ambas herramientas. Análisis teórico y de la metodología subyacente.

- Desarrollo de un modelo farmacocinético (prueba de concepto en el área renal sobre la librería KINETIC) con unas características concretas en ambos lenguajes. Análisis comparativo de aspectos metodológicos y también numéricos.

- Comparación computacional: Consumo de recursos en diferentes condiciones.

- Aportaciones al desarrollo teórico e implementación sobre EcosimPro de una librería de modelado de componentes farmacocinéticos con capacidad de ser extendida al dominio fisiológico, llamada LibPK, en tres fases:
    - LibPK versión Alfa.
        - Aportaciones a pruebas de concepto sobre la versión Alfa de LibPK.
    - LibPK version pre-Beta.
    - LibPK versión Beta.

4. Métodos y Materiales.

4.1. Metodología de la Prueba de concepto de diálisis.

Analizaremos la metodología de las herramientas de modelado en el ámbito del modelado farmacocinético mediante un modelo para la descripción de la evolución en plasma y otros compartimentos de la concentración de toxinas urémicas durante una sesión de hemodiálisis, desarrollado por Prado Velasco y col. en la librería KINETIC de su Tesis Doctoral [7] llamado UreaKin3p.

Este modelo ha sido identificado sobre los datos clínicos de un paciente con insuficiencia renal crónica terminal y sus resultados comparados con éxito con referencia a un modelo previamente publicado y validado mediante comparación con un sistema de monitorización en tiempo real de la urea plasmática en el artículo de Canaud [8]. El procedimiento de identificación del modelo de control fue igualmente validado previamente en el artículo de Prado Velasco y col. [9].

En esta prueba de concepto se desarrolla un modelo compartimental del organismo para la descripción de un proceso de hemodiálisis. En el artículo de Canaud [8] se presenta un contraste con resultados experimentales mediante un modelo bicompartimental. Estos modelos multicompartimentales del organismo han demostrado una elevada capacidad para describir la dinámica de sustancias endógenas y exógenas en farmacocinética. Su aplicación a nefrología también ha sido muy fructífera. Así el modelo de dos compartimentos es capaz de describir con mucha precisión la dinámica de la urea inducida por la hemodiálisis.

A su vez, Prado Velasco, en su Tesis Doctoral [7] presenta un modelo tricompartimental que también ha sido usado en otros trabajos recientes [10] de modelización del organismos para sesiones de hemodiálisis.

Desarrollaremos este mismo modelo tricompartimental del organismo para la prueba de hemodiálisis en Simulink de forma que se puedan analizar las diferencias metodológicas en la formulación del modelo y los resultados del mismo en el apartado 5 de Resultados y discusión del presente proyecto.

4.1.1. Librería KINETIC en EcosimPro. Modelo UreaKin3p.

Se desarrolla en la Tesis Doctoral de Prado Velasco [7] y en otros trabajos suyos [11] la librería KINETIC en EcosimPro como parte de un nuevo sistema de tele-asistencia.

Se trata de una librería farmacocinética desarrollada según los principios de partición basada en componentes físicos (MOO). La partición del sistema en componentes físicos se combina con una división basada en procesos físicos, a un nivel inferior, aprovechando las características de modelado jerárquico de los lenguajes orientados a objetos.

Esta librería cinética compartimental está basada en una partición sugerida por los componentes físicos [12]. La principal diferencia entre esta partición y una partición estrictamente definida por componentes físicos, esta en la flexibilidad para desplazar algún tipo de proceso que tiene lugar en un componente físico, a otro adyacente.

La herencia es aplicada para desarrollar procesos físicos más avanzados, ademas de incluir mecanismos de transferencia de masa adicionales. En ciertos casos en los que la herencia no es suficiente para añadir comportamientos más complejos a un proceso físico, puede aun reutilizarse la mayor parte de lo existente por medio de una herencia con sustitución selectiva de de ciertas ecuaciones. Este último concepto puede implementarse mediante ecuaciones virtuales en EcosimPro.

Estructura y componentes fundamentales de la librería Kinetic.

La librería desarrollada está compuesta por dos tipos de procesos físicos fundamentales. Estos son la conservación de las especies químicas, uniformemente distribuidas en el interior de los compartimentos, y los procesos de transferencia de masa convectiva y difusiva. El primer tipo de proceso quedará encapsulado en un componente de simulación, y la transferencia de masa en otro.

El primer proceso físico es encapsulado en un componente llamado **PoolBase**, y permite la existencia de diferentes tipos celulares, los cuales transfieren a su vez solutos con el compartimento principal, a través de sus membranas celulares. Se ha despreciado el desplazamiento de fluidos inducido ósmosis a través de las membranas de estas células libres (no ligadas a tejidos). Este componente es parametrizable, de manera que se pueden elegir tanto las especies químicas, como los tipos celulares.

El segundo proceso físico fundamental en la librería farmacocinética es la transferencia de masa a través de membranas. La transferencia convectiva queda implícita en las ecuaciones de conservación de la masa del **PoolBase**. Las ecuaciones fenomenológicas asociadas quedarán encapsuladas en un componente de simulación denominado **Membrane**.

Para completar los procesos fundamentales, se añade un modelo unidimensional de dializador, denominado **IdealDialyzer**. El adjetivo ideal viene de la consideración de eficiencia constante. Su aplicación está, por tanto, limitada a sesiones de HD desarrolladas con condiciones operativas constantes.

A partir del componente **PoolBase** son generados otros componentes de simulación con diferentes entradas y salidas de fluido. Estas últimas son empleadas, por ejemplo, para implementar el acceso vascular del paciente. Estos componentes son generados mediante herencia de clase, utilizando las ecuaciones virtuales para considerar los diferentes flujos de especies que pueden venir de las diferentes conexiones a membranas y accesos libres de flujo.

Los componentes son conectados, agregados, por medio de dos tipos de conectores parametrizables. El primero de ellos, denominado **kinfree**, permite la transferencia de fluido mediante convección. A través de este tipo de puerto pueden transferirse células libres. El segundo tipo de conexión, **kinmem**, está asociado a la transferencia de masa tanto mediante difusión , como convección, pero no permite el paso de células libres.

Aplicación al desarrollo de un modelo cinético de tres compartimentos.

El modelo cinético de tres compartimentos de la Urea, modelo UreaKin3p, ha sido preparado con los componentes desarrollados en esta librería. Éste ha sido construido agregando tres componentes tipo compartimento, junto con dos membranas difusivas, y un dializador, parametrizados para manejar sólo una especie química, la Urea.

Los flujos de agua entre compartimentos han sido implementados como condiciones de contorno, aplicando el algoritmo de distribución de fluidos expuesto también en la Tesis Doctoral de Prado Velasco [7].

Estas condiciones de contorno fueron aplicadas durante la fase de experimentación como funciones, que fueron definidas anteriormente, y sus resultados son contrastados en la Tesis Doctoral de Prado Velasco [7].

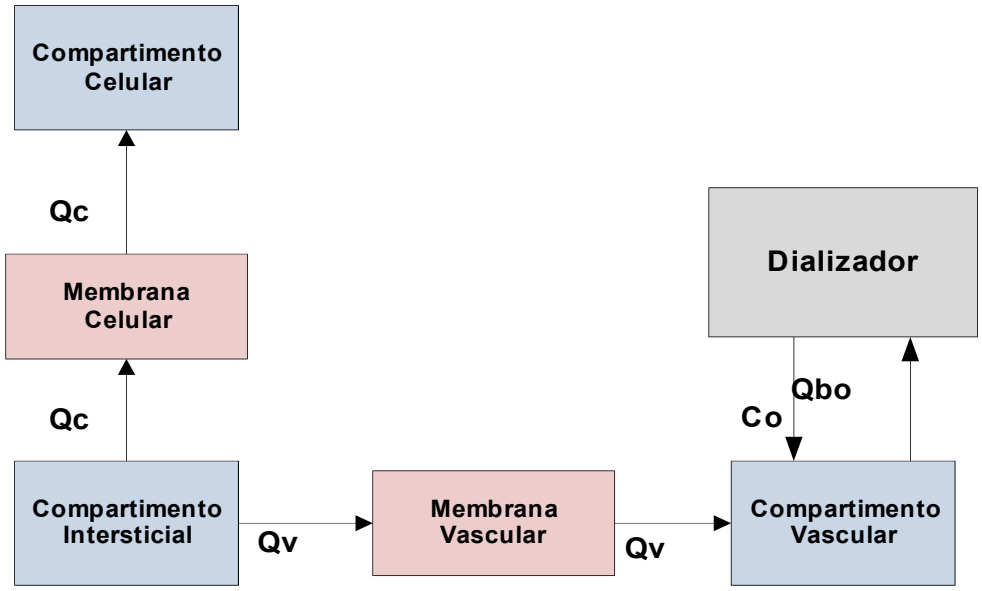


Figura 4.1. Diagrama con el modelo de 3 compartimentos en una sesión de HD.

4.1.2. Desarrollo del modelo UreaKin3p sobre Simulink.

Se procede al desarrollo del modelo UreaKin3p sobre Simulink, para ello hay que dejar de lado el modelado orientado a objeto y centrarse en el modelado por diagramas de bloques.

Primeramente se requiere un profundo estudio de las ecuaciones del modelo, UreaKin3p desarrollado en EcosimPro, y la manipulación de las mismas.

Las ecuaciones de continuidad que describen la dinámica de un soluto, e.g urea, suponiendo que la urea está uniformemente distribuida en cada compartimiento.

$$\frac{dc}{dt} \cdot V + \frac{dV}{dt} \cdot c + W_{total} = G$$

Para el solvente, agua:

$$\frac{dV}{dt} - W_{solvent} = 0$$

Donde:

c - Concentración de especie.
V - Volumen.
W_{total}- Flujo total de soluto.
$W_{solvent}$- Flujo total de solvente.
G - Generación de especie.

El flujo de solvente viene definido por el usuario en el modelo UreaKin3p como condición de frontera.

Por otro lado las ecuaciones que describen los procesos de transferencia de masa convectiva y difusiva, la ley de Fick y de arrastre. Ya que en esta versión de la librería se ha despreciado el desplazamiento de fluidos inducido ósmosis a través de las membranas.

$$W_{dif} = A \cdot Kdiff \cdot (c_1 - c_2)$$

$$W_{total} = c \cdot W_{dif} + W_{solvent}$$

Donde:

c – Concentración de especie en el interior de la membrana.
c_1 y c_2 – Concentración de especie en los límites de la membrana.
W_{dif} – Flujo difusivo.
A- Área de transferencia.
Kdiff- Constante de difusión.

Las ecuaciones del dializador en el modelo UreaKin3p de EcosimPro son aproximaciones que representan la dinámica de un dializador ideal, definido exclusivamente mediante la efectividad y el flujo ultrafiltrado.

Como condición de frontera se da el flujo total entrante al Dializador.

$$W_{bulkI} - W_{bulkO} = w_{uf}$$

$$w_I[i] - w_O[i] = w_{cheli}$$

$$w_{cheli} = effic \cdot W_{bulkO} \cdot c + w_{uf} \cdot c$$

Donde:

c- Concentración de especie en el interior del Dializador.
effic- Eficiencia del Dializador.
w_I - Flujo de especie entrante en el Dializador.
w_O - Flujo de especie entrante en el Dializador.
w_{cheli}- Flujo eliminado de especie.
W_{bulkI}- Flujo total entrante en el Dializador.

W_{bulk0}- Flujo total saliente del Dializador.
w_{uf} - Flujo ultrafiltrado.

El flujo eliminado, w_{cheli}, será de especie del compartimiento vascular.

Por otro lado el flujo ultrafiltrado, w_{uf} , es volumen eliminado de agua del compartimento vascular y se considera dato. Así se averigua el flujo total saliente del Dializador.

Estas ecuaciones son modeladas dentro de subsistemas para luego conectarse entre ellos de forma que las variables que se requieran sean compartidas. Sin embargo, tal y como se observa en un lenguaje causal las variables son de entrada o salida de los subsistemas y no compartidas.

Los subsistemas requeridos son los tres compartimentos: Vascular, Intersticial y Celular. Además de la Membrana y el Dializador.

Los tres compartimentos tienen en su interior la ecuación básica de continuidad de especie química.

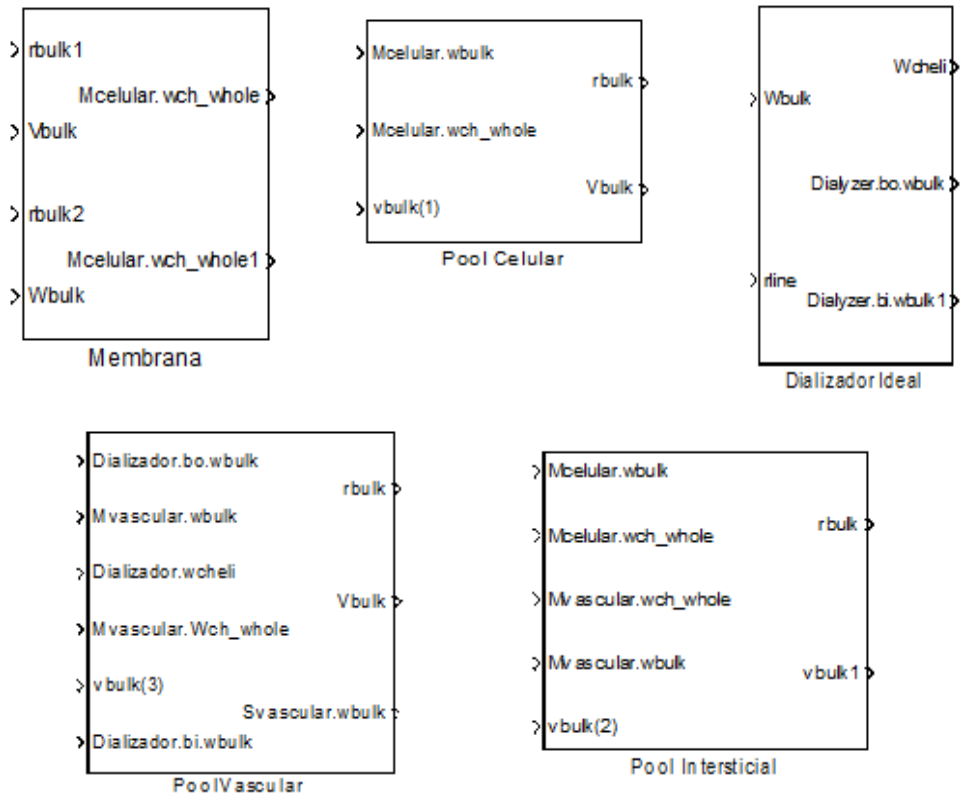


Figura 4.2. Susistemas creados en Simulink para el modelo UreaKin3p.

Analizamos el interior de alguno de estos subsistemas, donde se puede ver como la implementación de las ecuaciones anteriormente expuestas son descritas mediante diagramas de bloques. En el caso de los subsistemas compartimentos o pools tienen otro subsistema en su interior.

Esta tipo de descripción requiere una manipulación de las ecuaciones anteriores de forma que estas sean causales, o lo que es lo mismo hay que estudiar el sistema para encontrar las variables que deben ser resueltas en la ecuación, salidas, y las variables que ya están resueltas actúan como entradas. Esto implica poner la ecuación de forma computacional.

Los valores que son condiciones de frontera, como la generación 'G' o las variables de estado, como la concentración 'r', son representadas como variables tipo escalón de forma que el valor tiene que ser proporcionado por el usuario.

Los datos o constantes son proporcionados mediante bloques constantes o directamente con bloques multiplicación por constantes.

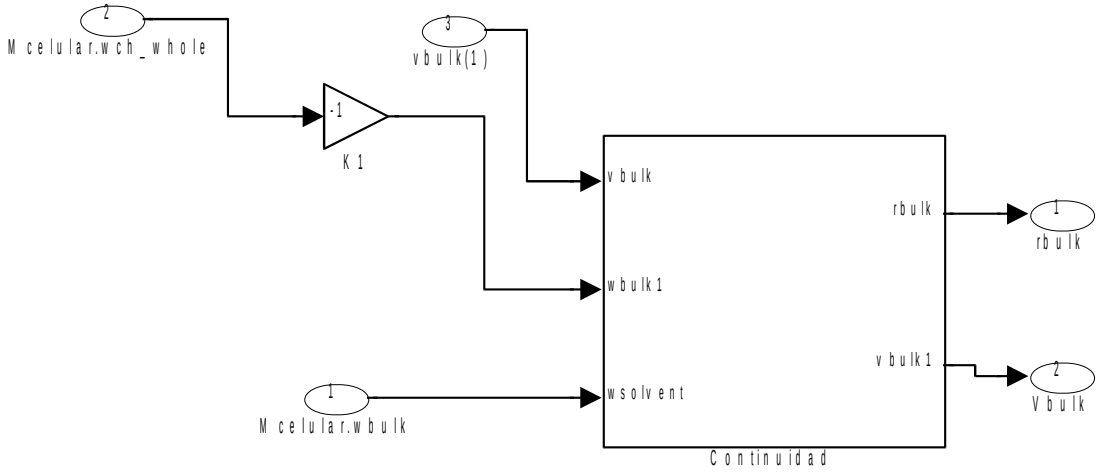


Figura 4.3. Compartimento celular en Simulink.

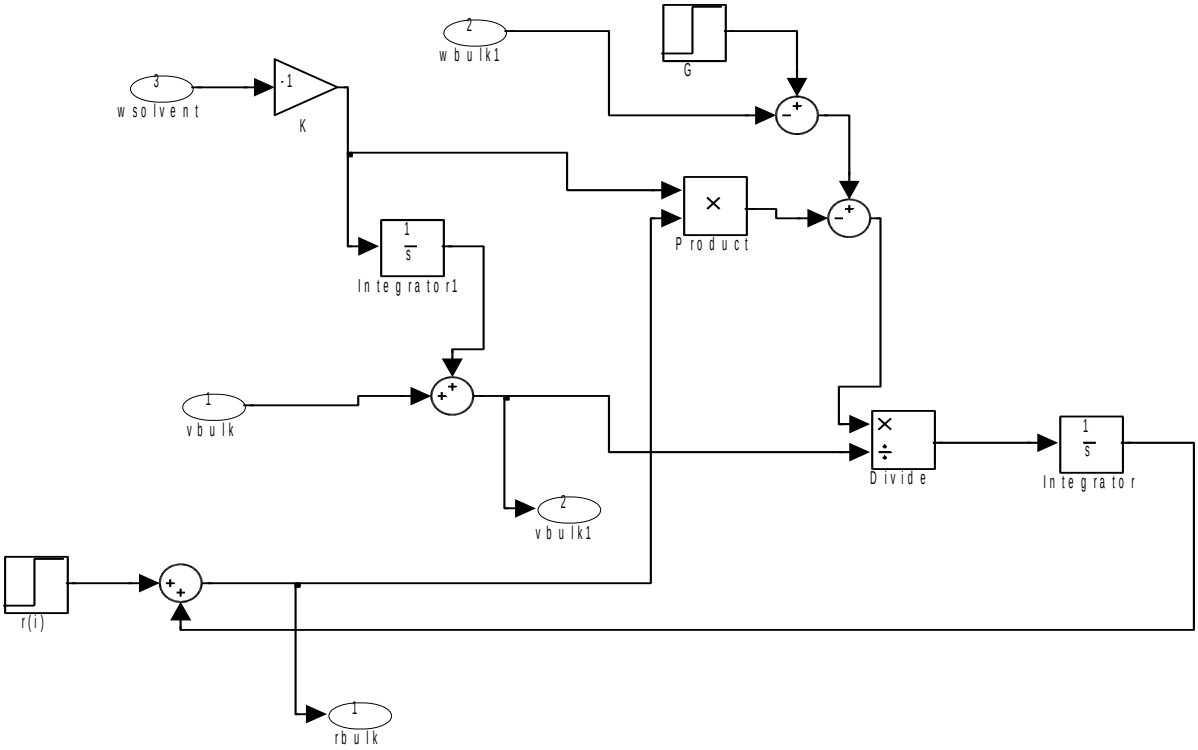


Figura 4.4. Subsistema continuidad en Simulink

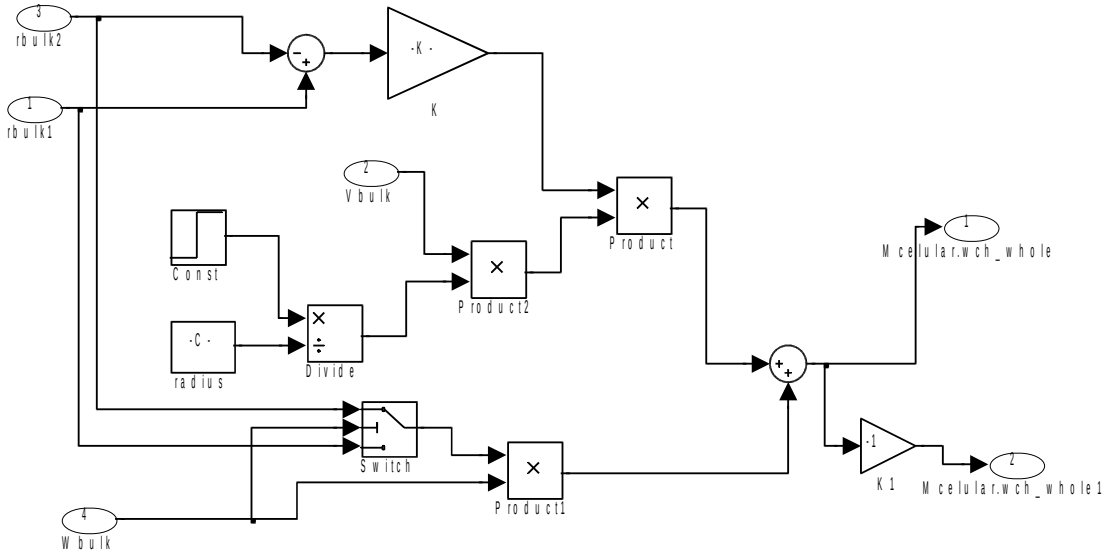


Figura 4.5. Subsistema membrana en Simulink.

Una vez creados los subsistemas necesarios para el modelo UreaKin3p se conectan entre ellos.

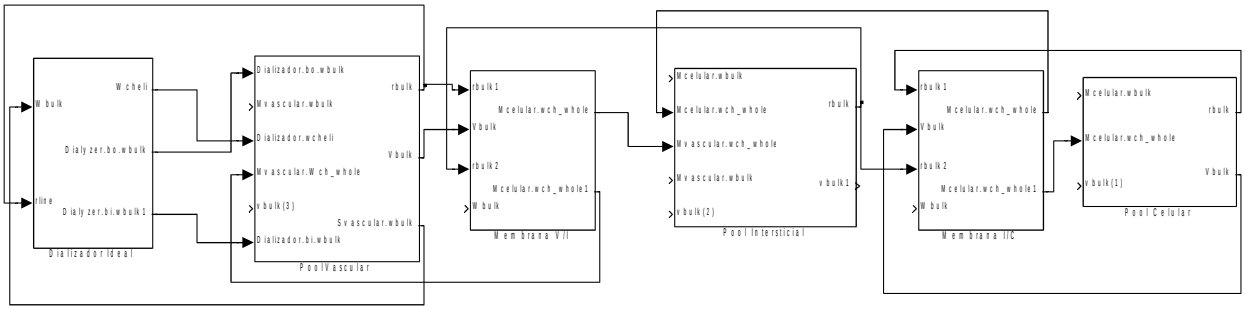


Figura 4.6. Modelo Ureakin3p en Simulink sin condiciones de frontera.

En los modelos que mostramos vemos como algunas de las entradas de los bloques no tienen valor asignado: los volúmenes de los compartimentos y los flujos volumétricos del solvente.

Sus valores vienen definidos, en el primer caso, por el algoritmo de cálculo de fracciones de variación volúmenes en hemodiálisis desarrollado en la Tesis de Prado Velasco [7] bajo el nombre de *volhd.* Este algoritmo recibe un valor de volumen a partir de los datos concretos de un paciente de la función *VvaronChertow* [11] para luego calcular las fracciones volumétricas de cada compartimento. En la fase siguiente, la de rebote, el algoritmo encargado del cálculo de fracciones de variación de volúmenes es llamado *volred*, también desarrollado en la Tesis de Prado Velasco [7].

En el caso de los flujos volumétricos de solvente, sus valores vienen definidos por condiciones de frontera implementadas por el usuario, además de algunos valores que toma del algoritmo de cálculo de volúmenes, *volhd*, durante la sesión de HD[5] . En la fase siguiente o fase de rebote estos flujos volumétricos cambian en su definición y los valores los toman de la función *volreb.*

El flujo entrante al dializador es un valor definido como condición de frontera y basado en valores experimentales. Después de la sesión de HD, durante la fase de rebote, su valor es lógicamente nulo.

Para marcar el final de la sesión de HD y comenzar la fase de rebote usamos un bloque condicional de Simulink denominado *Switch.*

[5] Hemodiálisis.

De esta forma se modela el evento en Simulink, dando paso al flujo pertinente en el momento de finalización de la sesión de HD. Este momento final de sesión es un dato que se incluye en el bloque *Switch* y que compara con el valor que recibe del bloque *Clock.*

En la figura siguiente se muestra como se implementan los bloques que proporcionan al modelo condiciones de frontera, flujos volumétricos y condiciones iniciales de algunas variables, volúmenes. Se marcan en rojo las salidas que luego conectaremos a las entradas vacías del modelo anterior de tres compartimentos y Dializador.

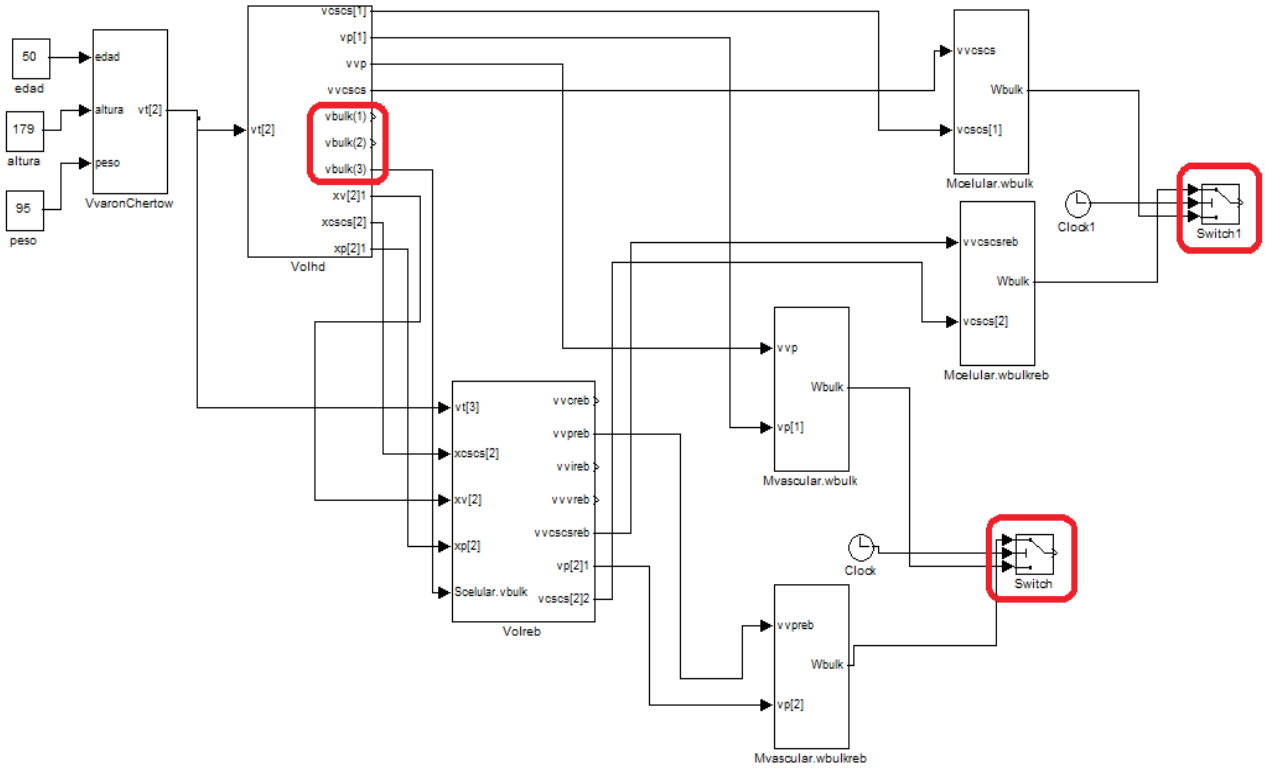


Figura 4.7. Condiciones de frontera del modelo UreaKin3p en Simulink.

Así pues el modelo completo implementado en Simulink de tres compartimentos y dializador, parametrizado para un único soluto, Urea. Con datos y condiciones de frontera basadas en los valores de la Tesis de Prado Velasco [7] se puede ver en la figura 4.8.

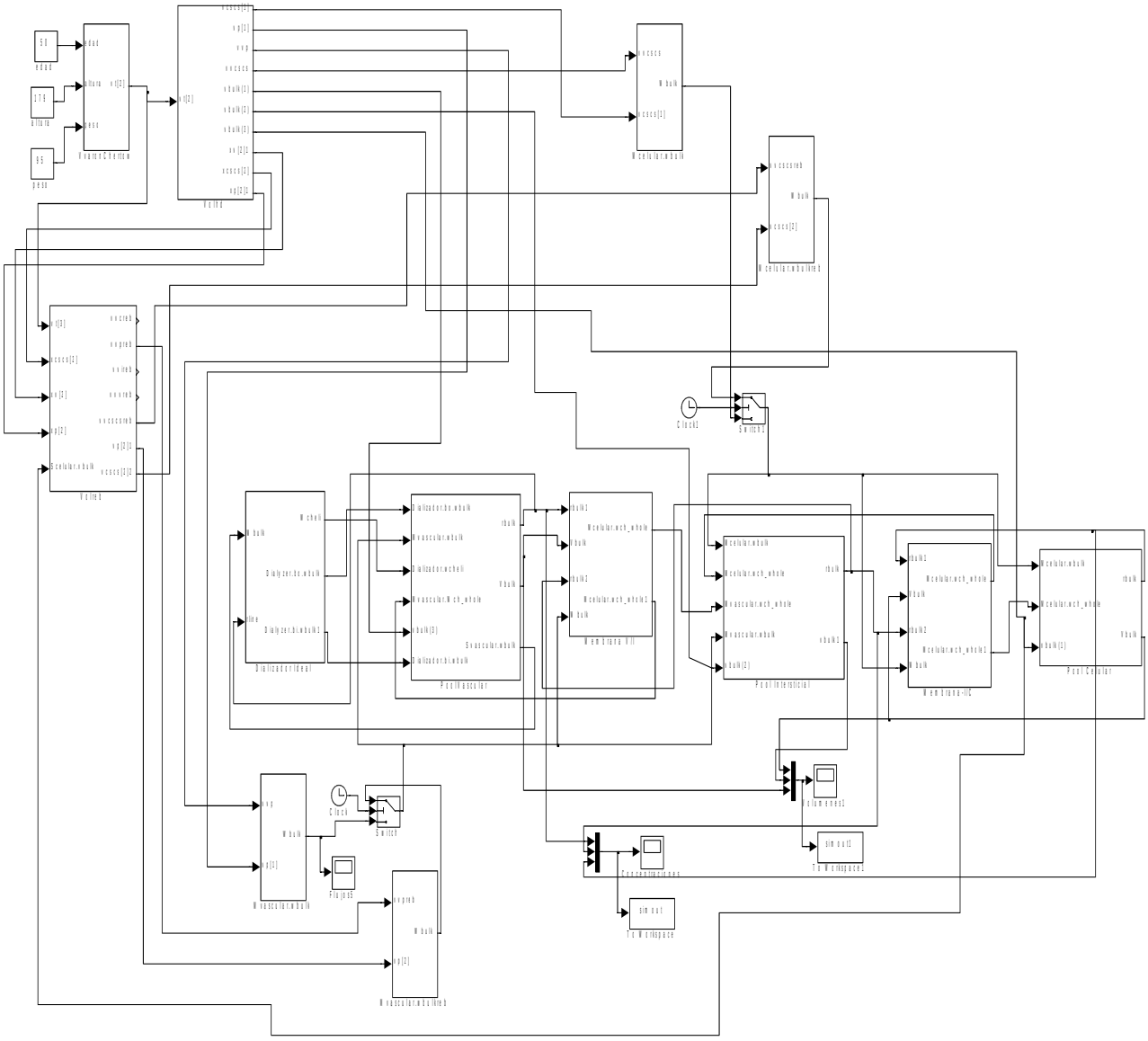


Figura 4.8. Modelo UreaKin3p con Condiciones de frontera.

4.2. Desarrollo de LibPK.

LibPK es una librería de componentes de simulación de sistemas farmacocinéticos compartimentales multinivel. Ha sido desarrollada dentro de un equipo de investigación y comparte marco de investigación con otro proyecto presentado por la alumna Cristina Vallez. La idea es que pueda ser extendida a diferentes dominios fisiológicos.

Un sistema farmacocinético está orientado a la descripción de los procesos Liberación, Absorción, Distribución, Metabolismo y Excreción en los organismos. No considera la biofase ni los mecanismos fisico-químicos que provocan el efecto terapéutico.

La librería LibPK ha sido desarrollada a partir de una metodología de modelado que busca la creación de elementos cuyas caracteristicas principales sean la reusabilidad de los mismos y la capacidad de definir modelos multinivel fuertemente acoplados a partir de ellos.

La reusabilidad se consigue partiendo de la idea de modelos acausales y se aplica no sólo a nivel de elemento mediante la no definición de las tradicionales entradas y salidas, a través de puertos conectables entre ellos que no definen una dirección de flujo. Sino también a nivel de modelo gracias una flexibilidad a la hora de definir las condiciones de frontera, o entradas y variables del sistema, salidas.

También otros aspectos como la capacidad de parametrización de los elementos o la construcción de nuevos elementos mediante herencia ayudan a la reusabilidad, pero estos aspectos son aportados por el lenguaje de programación orientado a objeto del programa elegido para su implementación, EcosimPro.

La capacidad de describir modelos multinivel comienza por el particionado del sistema, fisiológico en nuestro caso, en la librería LibPK usamos un particionado sugerido por la física, encapsuladando procesos fisiológicos y mediante los mecanismos de agregación y herencia somos capaces de construir elementos distintos con sentido físico, los cuales a su vez son conectados entre ellos construyendo un sistema mayor. Esta arquitectura multinivel de los elementos asegura el funcionamiento de modelos fuertemente acoplados y ayuda en la adición de conocimiento.

Diferenciamos de esta forma entre dos capas, la capa privada o de procesos incluye la dinámica de los procesos físicos o fisiológicos. Se consideran a los procesos como las piezas más pequeñas no modificables a través de las cuales formar la siguiente capa, la capa pública. La capa pública contiene los elementos, personalizables y están formados por procesos de la capa privada.

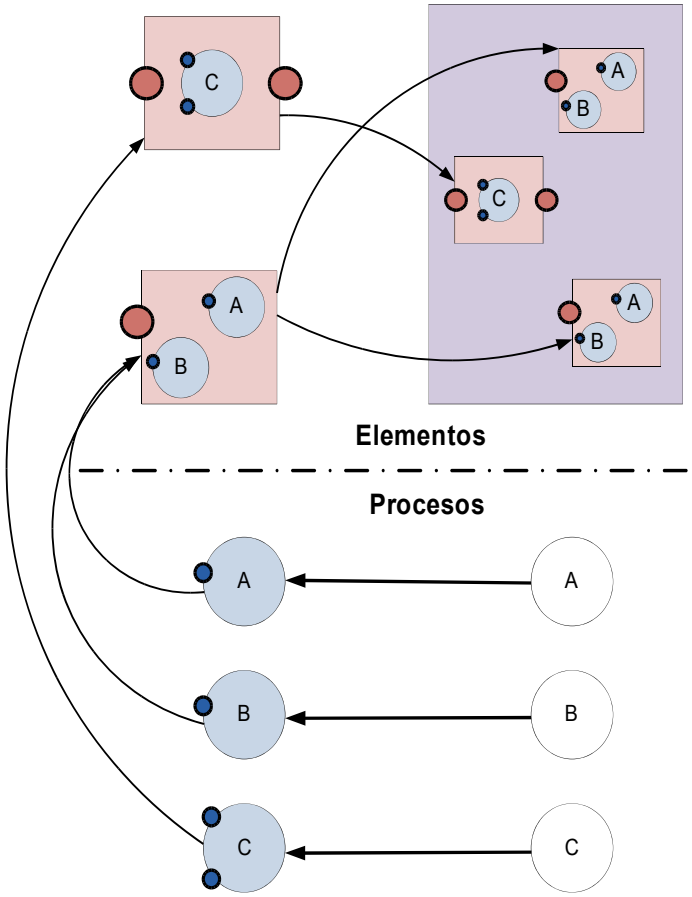


Figura 4.9. Arquitectura de la libreria LibPK.

Para lograr los objetivos metodológicos planteados, el desarrollo de la librería LibPK se hace en tres fases iniciales: versión Alfa, pre_Beta y Beta que serán presentadas en el apartado 5.2 de Resultados.

5. Resultados y discusión.

5.1. Resultados del modelo UreaKin3p en EcosimPro y Simulink.

Mostramos algunos resultados de la simulación de una sesión de hemodiálisis sobre el modelo UreaKin3p en EcosimPro y en Simulink .

El modelo en EcosimPro es simulado usando el método de integración por defecto, DASSL. Usamos un CINT = 60. Esta variable global se usa para indicar el intervalo para producir resultados de la simulación.

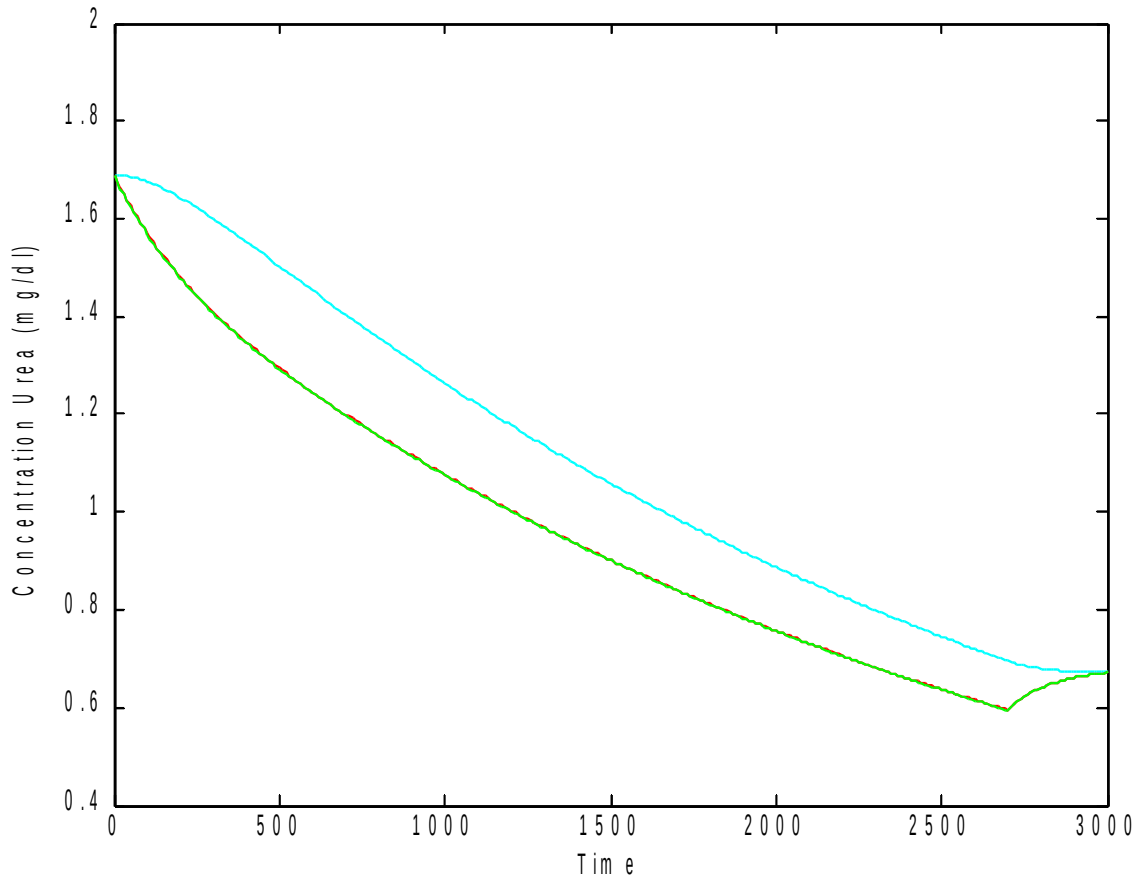


Figura 5.1. Gráfica de la concentración de Urea en los distintos compartimentos (Celular,Intersticial y Vascular) durante una sesión de HD simulada en EcosimPro. Modelo UreaKin3p.

Para la simulación del modelo construido en Simulink en el apartado 4.1.2 del proyecto se requiere que se inicializen algunos parámetros de configuración para la resolución del sistema: el tiempo de simulación, el tipo de solver, tamaño del paso de integración y la tolerancia.

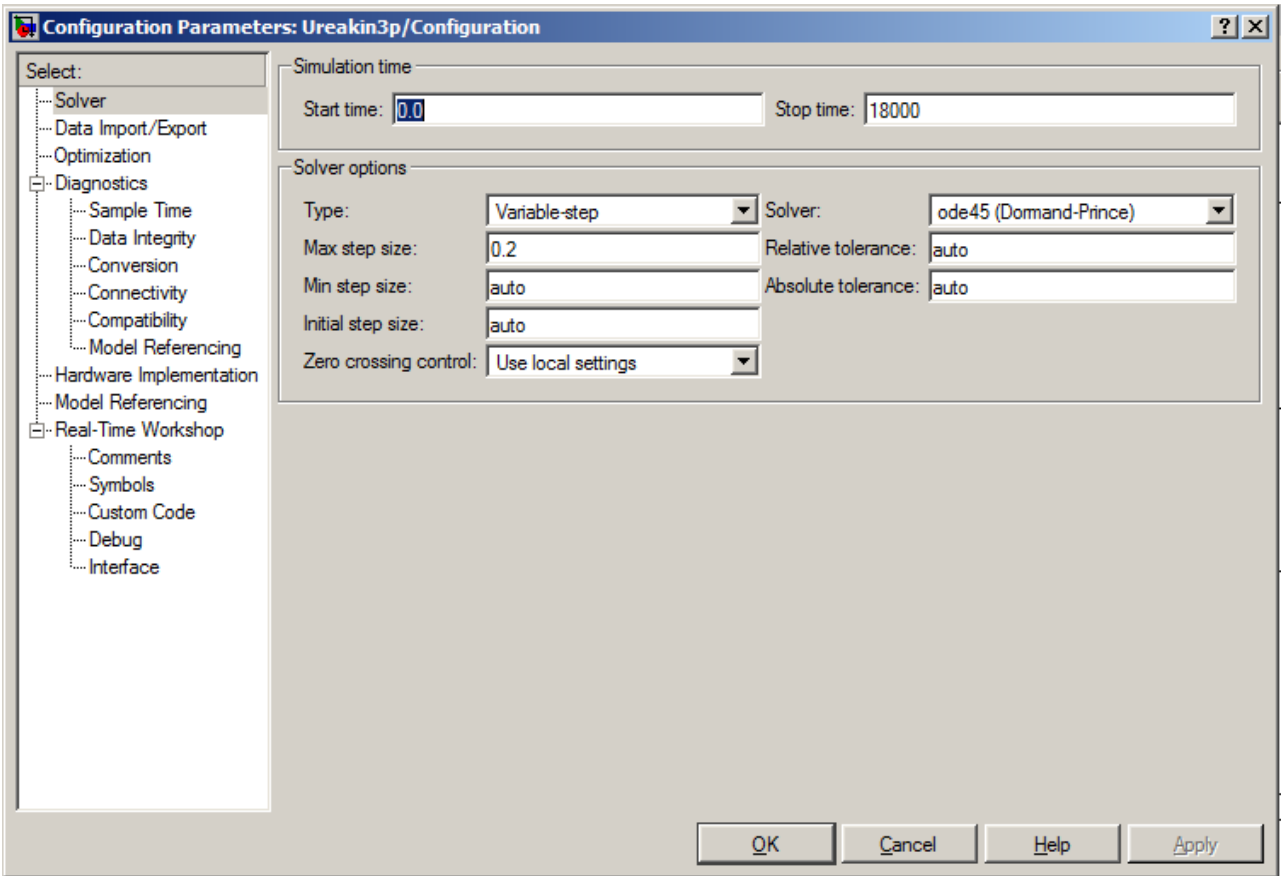


Figura 5.2.Ventana para la configuración de los parámetros de simulación en Simulink.

Con respecto al tiempo total de simulación, usamos el mismo que se usa en la experiencia de la Tesis de Prado Velasco [7] basado en valores experimentales.

El método de integración usado es el estándar ode45.

El tamaño del paso viene definido por sus valores máximos y mínimos, como máximo se toma un valor de 0.2 mientras que el valor mínimo será decidido por el solver.

La tolerancia viene definida por el programa, también de forma automática.

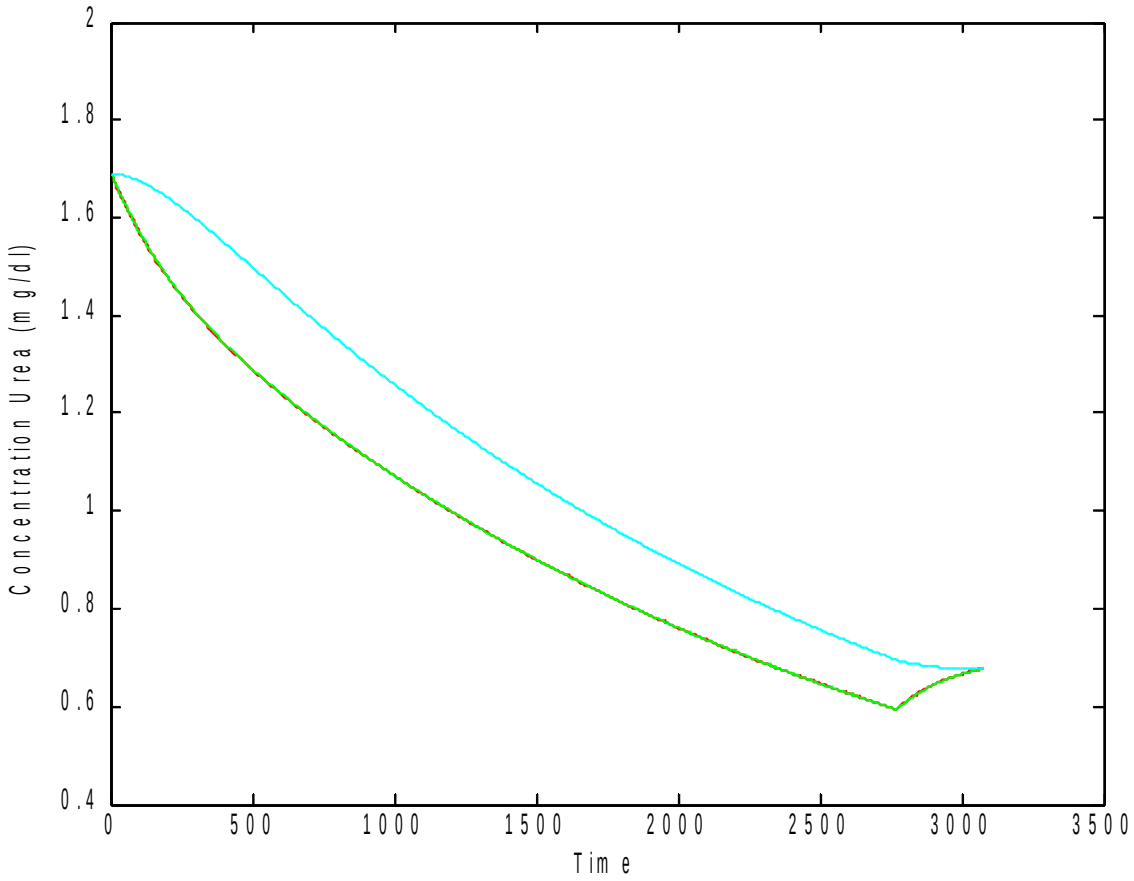


Figura 5.3. Grafica de la concentración de Urea en los distintos compartimentos (Celular,Intersticial y Vascular) durante una sesión de HD simulada en Simulink. Modelo UreaKin3p.

Sus resultados son prácticamente idénticos, las diferencias se deben al método de integración utilizado en cada simulación: en Simulink, ode45 y en Ecosimpro, DASSL. En la siguiente figura mostramos ambas gráficas de forma comparada. Más adelante haremos un análisis más detallado de la precisión de las simulaciones usando el mismo método de integración.

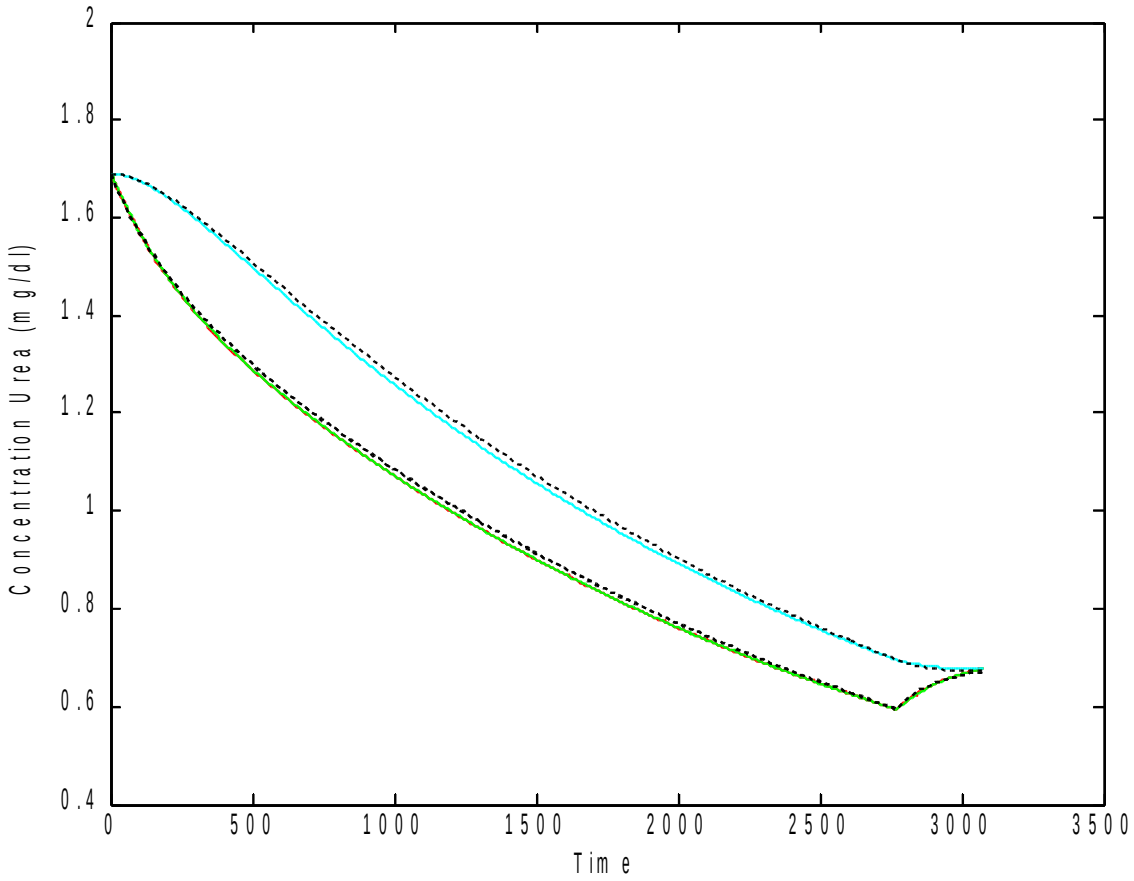


Figura 5.4. Gráfica de las concentraciones de Urea en Modelo UreaKin3p simulado en EcosimPro (puntos negros) y en Simulink (lineas de colores).

5.1.1. Análisis comparativo.

Compararemos ahora las dos implementaciones que hemos analizado en dos programas de modelado, con distinta metodología, del modelo de Diálisis de Urea sobre un modelo tricompartimental del organismo. Este modelo y su simulación fueron primeramente implementados en EcosimPro en la Tesis de Prado Velasco [7].

Por otro lado la implementación en Simulink ha sido realizado como parte de este proyecto fin de carrera, ya que este programa usa el modelado mediante diagrama de bloques, para estudiar las diferencias y ventajas metodológicas y numéricas que pueden existir entre las dos formas de modelado: Orientado a Objeto y mediante Diagramas de Bloques.

5.1.1.1. Aspectos Metodológicos.

El primer aspecto a tener en cuenta en la comparación es la forma de introducir las ecuaciones del modelo.

En EcosimPro las ecuaciones son implementadas de forma directa y parametrizables usando las sentencias que aporta el lenguaje de simulación.

Las ecuaciones son parametrizables para cualquier especie química, también son parametrizables los elementos, para la inclusión de subpools o pools contenidos.

Así por ejemplo la implementación de las ecuaciones de continuidad a partir de las ecuaciones generales:

$$\frac{dc}{dt}\cdot V+\frac{dV}{dt}\cdot c+W_{total}=G$$

$$\frac{dV}{dt}-W_{solvent}=0$$

Se hace de la siguiente forma:

```
EXPAND (i IN ChemicalComp)

vbulk[setofElem(CelComp,1)]'*rbulk[setofElem(CelComp,1),i]+\
rbulk[setofElem(CelComp,1),i]'*vbulk[setofElem(CelComp,1)]+\
     wtotal[setofElem(CelComp,1),i] = generacion[i] +\
     SUM(j IN CelComp EXCEPT setofElem(CelComp,1); wdif[j,i])
```

```
     EXPAND (i IN CelComp)
          vbulk[i]'+wsolvent[i]=0
```

Las sentencias EXPAND se usan para la parametrización, cuando se parametriza para las especies químicas se usa la etiqueta ChemicalComp y cuando se parametriza los subpools o pools contenidos se usa la etiqueta CelComp.

Los grupos de parámetros son previamente definidos mediante la sentencia ENUM.

```
ENUM biochemical = {Urea,Creatinina,Na,K,Ca,Cl,B2M,B12}

ENUM poolmix = {Single,Erythr}
```

En la declaración del elemento se incluye una etiqueta para cada parámetro, ChemicalComp y CelComp.

```
COMPONENT PoolBase (SET_OF (biochemical) ChemicalComp, SET_OF
(poolmix) CelComp)
```

Esta etiqueta no es definida por el usuario hasta la instanciación del elemento, en la cual, el usuario elige entre los grupos de parámetros definidos bajo la sentencia ENUM correspondiente.

```
COMPONENT PoolAisladoUrea
DECLS
     SET_OF (biochemical) especies={Urea}
     SET_OF (poolmix) celulas={Single}
TOPOLOGY
     PoolBase (ChemicalComp=especies,CelComp=celulas)
     Pool1(kacel={{1}})
END COMPONENT
```

Esta forma de parametrización promueve la reusabilidad de los componentes creados, ya que los parámetros se aportan en la instanciación del modelo y no en la definición de este.

En Simulink se requiere un preprocesamiento mucho más manual para la introducción de las ecuaciones [13], ya que éstas son implementadas después de una transformación, de forma que sean relaciones causales entre las variables. Por ejemplo en el caso de las ecuaciones de continuidad, al ser ecuaciones de variables dinámicas, con derivadas, hay que transformar la ecuación de forma que las derivadas queden como resultado, obtener el ODE del sistema:

$$\frac{dc}{dt} = \frac{G - W_{total} - \frac{dV}{dt} \cdot c}{V}$$

$$\frac{dV}{dt} = W_{solvent}$$

Una vez sabemos el resultado de la derivada de la variable, se integra mediante el modulo Integrator, para después añadir este valor a la variable según la siguiente expresión:

$$x_t + \int \left(\frac{dx}{dt}\right) \cdot dt = x_{t+1}$$

Así pues, las ecuaciones de continuidad en diagrama de bloques se implementa de la forma ya vista en la figura 4.4.

En cuanto a la parametrización de las ecuaciones, casi todos los bloques de Simulink aceptan entradas escalares y vectores, por lo que se pueden definir varias especies químicas. Para el caso de parametrizar subpools lo veremos en el apartado referido al Multinivel.

El segundo aspecto que tenemos que tener en cuenta tiene que ver con la creación del modelo y la reusabilidad del mismo.

Por un lado, en EcosimPro, el propio lenguaje e interfaz del programa están pensados para la creación de elementos a través de los cuales construir un modelo, además de incluir mecanismos como herencia y agregación que promueven esta forma de construcción de modelos.

De esta forma, una vez creado un componente básico con las ecuaciones generales de comportamiento se pueden crear componentes que incluyan esa dinámica pero con algunos pequeños cambios mediante el mecanismo de herencia usando el componente básico. Esto permite que en nuestro modelo tricompartimental del organismo cada compartimento: Vascular, intersticial y celular, tengan una configuración y una distribución de flujos particular, teniendo un sólo componente que describa la dinámica, **PoolBase**, para definir el resto como hijos de este componente. A estos componentes hijos se les añaden puertos, a cada componente en función de su configuración. Por ejemplo al **PoolVascular** se le añaden 3 puertos, dos del tipo **kinfree** y uno del tipo **kinmem**. Las variables de los puertos se igualan o relacionan a las variables del componente padre que sean necesarias. Para cada componente hijo esta relación entre las variables externas o de puertos y las variables del componente padre cambia, para ello se usan las ecuaciones virtuales, un comando de EcosimPro

que permite cambiar las ecuaciones heredadas del componente padre.

De esta manera si definimos en **PoolBase,** componente padre, la siguiente ecuación virtual:

```
<solventfluxes> EXPAND_BLOCK (i IN CelComp)
                wsolvent[i]=0
                 END EXPAND_BLOCK
```

Esta ecuación es distinta en sus componentes hijos: **PoolVascular**.

```
<:solventfluxes> EXPAND_BLOCK (TRUE)
          EXPAND_BLOCK (i IN CelComp EXCEPT setofElem(CelComp,1))
               wsolvent[i]=bo.wbulk[i]-bi.wbulk[i]
          END EXPAND_BLOCK
          wsolvent[setofElem(CelComp,1)]= - epithe.wbulk + \
               bo.wbulk[setofElem(CelComp,1)]-
bi.wbulk[setofElem(CelComp,1)]
          END EXPAND_BLOCK
```

Y **PoolIntersticial**.

```
<:solventfluxes> EXPAND_BLOCK (TRUE)
          EXPAND_BLOCK (i IN CelComp EXCEPT setofElem(CelComp,1))
               wsolvent[i]=0
          END EXPAND_BLOCK
          wsolvent[setofElem(CelComp,1)]=     mvascular.wbulk     +
mcelular.wbulk
          END EXPAND_BLOCK
```

Vemos como una ecuación básica del elemento padre se transforma en sus elementos hijos en función de los puertos que tenga cada componente hijo.

Con esta forma de creación de elementos que permite el lenguaje orientado a objeto, tenemos un elemento padre que contiene la dinámica, pero sin posibilidad de conexión o integración en ningún modelo. También tenemos unos puertos para conexionar elementos que contienen las variables que compartirán o se relacionarán entre ellos.

Con estos dos elementos básicos hemos construido 3 elementos diferentes entre sí cuya dinámica es la misma pero ciertas variables, como los flujos, son particularizados en cada caso a los puertos que tenga el elemento.

Esto maximiza la reusabilidad de los elementos, ya que además de que los elementos creados, compartimentos, sean utilizables en diferentes situaciones y modelos hemos creado un elemento base a partir del cual se pueden crear infinidad de elementos combinando con los puertos.

En la figura 5.5 se representa la estructura jerárquica de los elementos

compartimentos o pools.

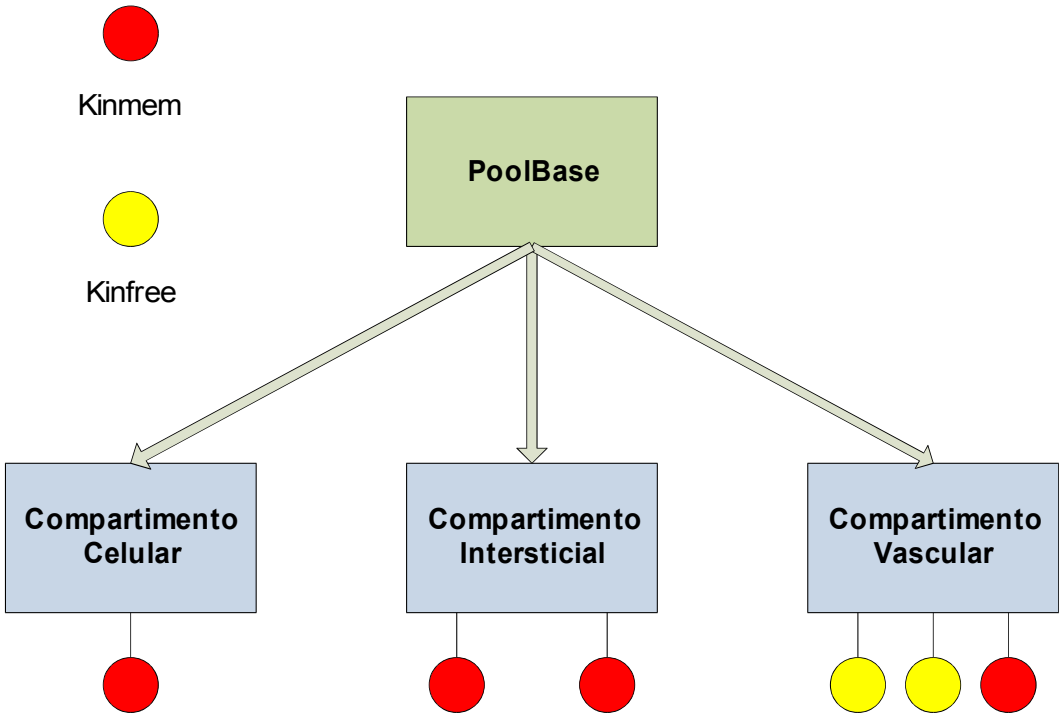


Figura 5.5. Ejemplo de construcción de elementos mediante herencia en EcosimPro.

Además, mediante el mecanismo de agregación, la conexión entre elementos para la construcción final del modelo es sencilla, ya que se hace instanciando los elementos que necesitemos para luego conexionarlos entre si.

En la instanciación se aportan los datos y parámetros de construcción que necesite cada elemento, esto aumenta la reusabilidad de los elementos, pues se aumenta la posibilidad de moldear a tu necesidad el elemento que necesitas para un modelo en concreto, además de poder declarar en el mismo modelo diferentes instancias del mismo elemento.

Por ejemplo, para la creación de nuestro modelo Ureakin3p la instanciación se hizo de la siguiente manera:

```
COMPONENT UreaKin3p

DECLS
     SET_OF (biochemical) especies={Urea}
     SET_OF (poolmix) celulas={Single}

TOPOLOGY

     PoolVascular (ChemicalComp=especies,CelComp=celulas) Svascular
```

```
(kacel={{1}})
     PoolCelular  (ChemicalComp=especies,CelComp=celulas)  Scelular
(kacel={{1}})
     PoolIntersticial       (ChemicalComp=especies,CelComp=celulas)
Sintersticial (kacel={{1}})
     IdealDialyzer          (ChemicalComp=especies,CelComp=celulas)
     Dializador (effic={0.7},wuf=1.4e-7)
     Membrane  (ChemicalComp=especies,  TipoGeometrico  =  Cylindric)
Mvascular (k=1,radius=1)
     Membrane  (ChemicalComp=especies,  TipoGeometrico  =  Spheric)
Mcelular (k=1, radius=1)

     CONNECT       Svascular.epithe       TO       Mvascular       TO
Sintersticial.mvascular
     CONNECT       Scelular.mcelular       TO       Mcelular       TO
Sintersticial.mcelular
     CONNECT Svascular TO Dializador TO Svascular

END COMPONENT
```

En la primera parte, DECLS, se parametriza el modelo para un sólo componente, Urea, y para compartimentos-pools sin subpools, ni tipos celulares en su interior.

En la segunda parte, TOPOLOGY, se incluyen los elementos que formarán el modelo y se instancian sus parámetros de construcción y datos, además de diferenciar los elementos iguales con etiquetas distintas para su identificación.

Al final se conectan los elementos entre sí, en algunos casos indicando directamente que puertos queremos que se conecten entre si.

En la conexión las direcciones de las variables de flujo en el puerto se resuelve automáticamente en función de la declaración de los puertos de los elementos, IN o OUT.

La reusabilidad en Simulink no viene implícita en el lenguaje ya que se trata de un lenguaje de diagrama de bloques en el cual son los bloques los elementos reusables. Aunque, tal y como hemos visto, permite la creación de subsistemas que pueden ser utilizados como elementos en si mismos.

El problema con estos subsistemas es la falta de mecanismos, como la herencia, para el retoque de los mismos elementos para otras situaciones o modelos.

Pero la capacidad de que los subsistemas trabajen a multinivel nos ha facilitado la creación de los 3 compartimentos con la base del subsistema *continuidad.*

Comparamos los subsistemas correspondientes a Vascular_Pool e Interstical_Pool.

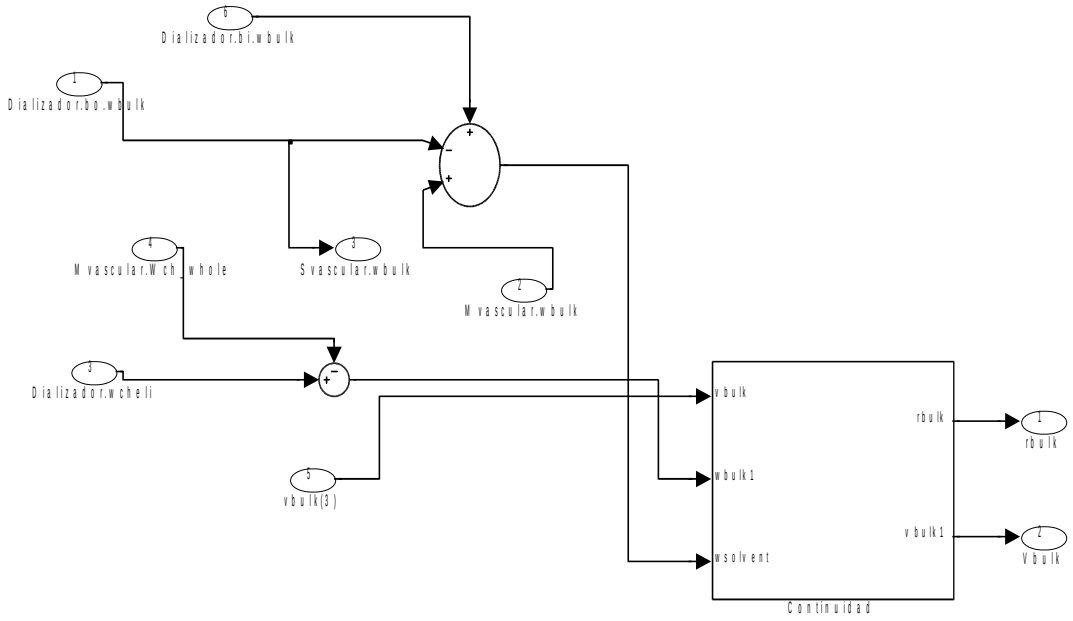


Figura 5.6. Compartimento Vascular modelado en Simulink.

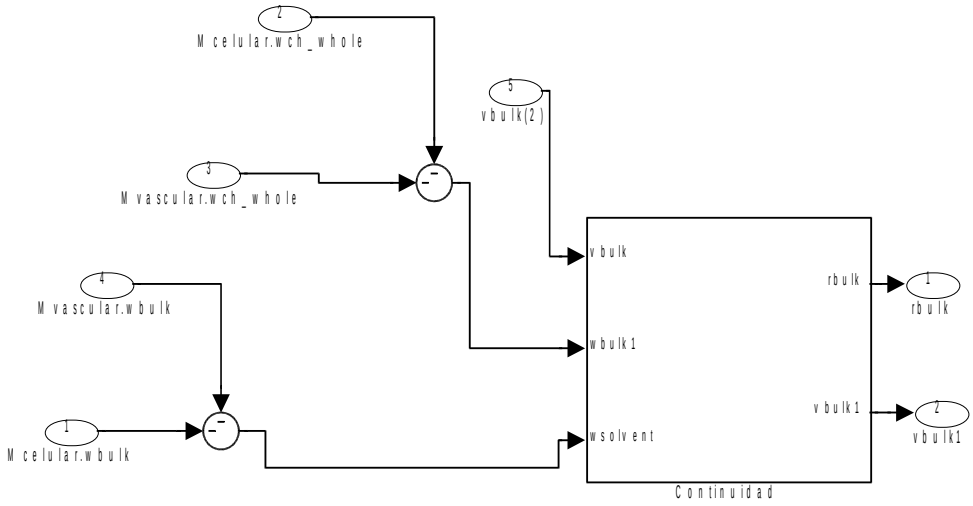


Figura 5.7. Compartimento Celular modelado en Simulink.

En estos subsistemas se observa como hemos tenido que implementar manualmente las direcciones de los flujos incluyendo bloques SUM, mientras que EcosimPro resuelve esto internamente creando las ecuaciones de suma de flujos automáticamente.

Ya que se define la reusabilidad como la capacidad de usar un objeto de un

modelo en otro modelo distinto sin tener que hacer modificaciones importantes [14], si para reusar un elemento de los anteriores hay que hacer alguna modificación importante se considerará ese elemento no reusable completamente.

Así pues los elementos en Simulink son parcialmente reusables y pese a que EcosimPro tiene mecanismos, como la herencia, que ayudan a mejorarla, necesitan de una intervención experta, modificando por ejemplo las ecuaciones virtuales, para llevarla a cabo. En siguientes versiones de la librería, explicada en la sección 5.2, Aportaciones a LibPK, usamos el mecanismo de agregación para la creación de nuevos elementos, eliminando las ecuaciones virtuales, aumentando la reusabilidad de los elementos, ya que reducen la intervención del usuario y permite mayor versatilidad.

El tercer aspecto que queremos comparar entre los programas es la capacidad de descripción de modelos multinivel, el cual se define por la integración de modelos matemáticos de procesos formulados a diferentes niveles, aspecto clave en el modelado de sistemas biológicos, ya que estos incluyen varios niveles desde redes intercelulares, órgano, sistema orgánico y organismo fuertemente acoplados [3].

En nuestro caso el modelo Ureakin3p es la representación de una sesión de HD en el cual el cuerpo humano es definido mediante un modelo de 3 compartimentos o pools.

Uno de ellos se corresponde al sistema vascular humano que se modela como un volumen homogéneo con especies disueltas en su interior, el cual es un modelo muy simplista del sistema vascular humano, que se comprende de un liquido plasmático y tipos celulares diversos en el cual las especies se distribuyen entre el plasma y en el interior de los tipos celulares, entre los que se encuentran los Eritorcitos por ejemplo y a los cuales se les denomina genéricamente subpool. Este sería un ejemplo de descripción multinivel que se utiliza en artículos recientes de modelado del organismo para sesiones de hemodialisis [10].

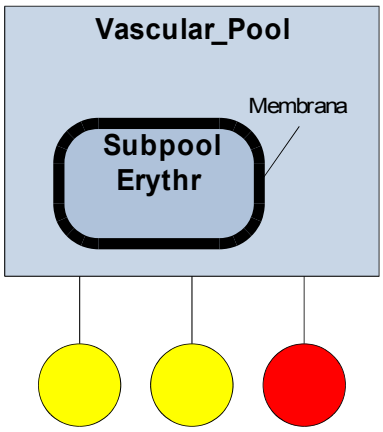


Figura 5.8. Esquema del modelado en EcosimPro del Sistema Vascular con Eritrocitos en su interior.

En EcosimPro hemos parametrizado los elementos compartimentos, pools de forma que se puedan crear elementos subpools o tipos celulares en el interior de estos compartimentos. Incluso se han predefinido que tipos celulares pueden ser definidos mediante la sentencia ENUM.

```
ENUM poolmix = {Single,Erythr}
```

Esta solución no aprovecha el mecanismo de agregación para a partir de los elementos ya definidos crear los nuevos elementos multinivel. Pero en la versión Beta de la librería, explicada en la sección 5.2 Aportaciones a LibPK del presente proyecto, ya se puede realizar la descripción multinivel de estos elementos mediante agregación.

En Simulink, para modelar el Eritrocito, tenemos los elementos definidos para la creación del nuevo elemento vascular: *Continuidad1* y *membrana*. Los cuales conectamos correctamente al subsistema continuidad ya existente del pool vascular, que es el que se conecta al exterior. De esta forma se desarrolla el modelo multinivel en Simulink a través de elementos ya creados, pero con los mismos problemas de dirección de las variables de flujo.

Hemos añadido las condiciones de frontera propias del subsistema continuidad, imponiendo flujo volumétrico nulo y un volumen no nulo.

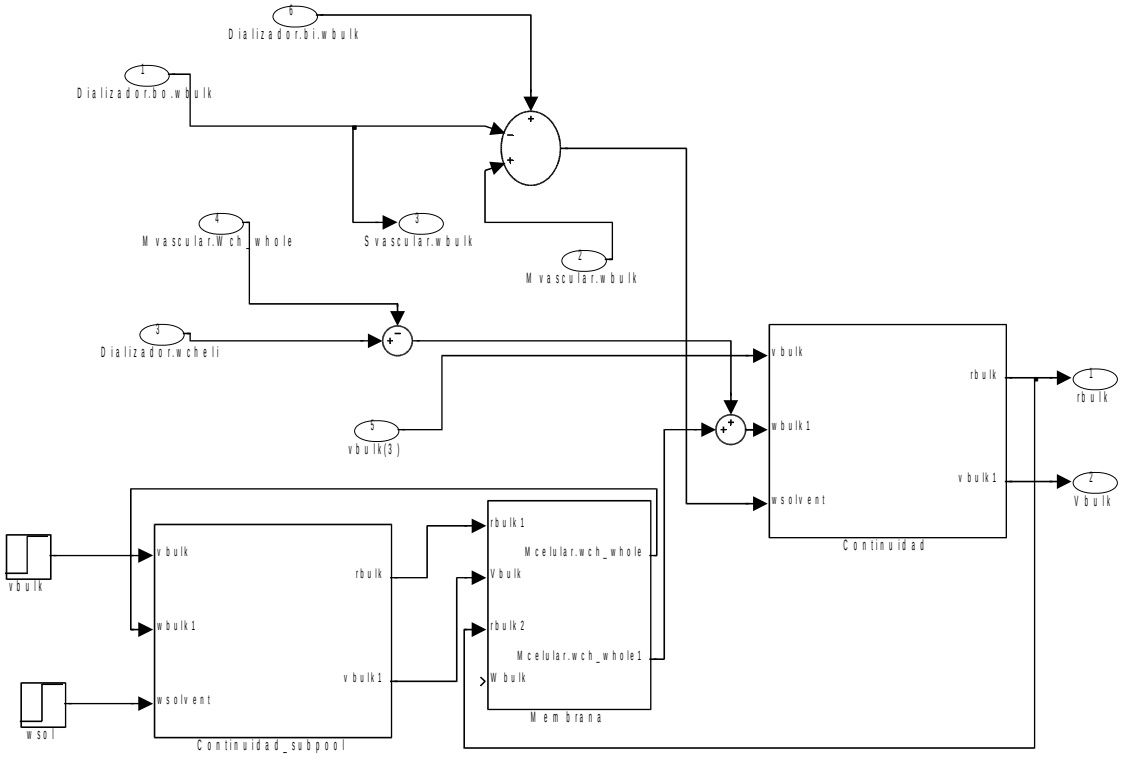


Figura 5.9. Compartimento Vascular con compartimento celular (Eritrocito) en su interior.

El cuarto aspecto metodológico que queremos comparar es el modelado de la detección de eventos en ambos entornos.

En EcosimPro la inclusión de eventos se realiza desde el propio lenguaje con las sentencias que tienes disponibles para ello: ZONE, IF, etc...

En el elemento membrana por ejemplo se incluye una de estas sentencias en la ecuación de flujo arrastrado. Ya que el flujo arrastrado de especie en una membrana depende del sentido del flujo.

```
mi.wch_whole[i]= ZONE (mi.wbulk >=0)
                 mi.wch_nobulk[i] + mi.rline[i]*mi.wbulk
          OTHERS mi.wch_nobulk[i] + mo.rline[i]*mi.wbulk
```

Para la inclusión de tal evento en el modelado del sistema en Simulink, se usa el bloque SWITCH, el cual elige entre la concentración adecuada en función del valor de la variable que se le transmite por el conector central.

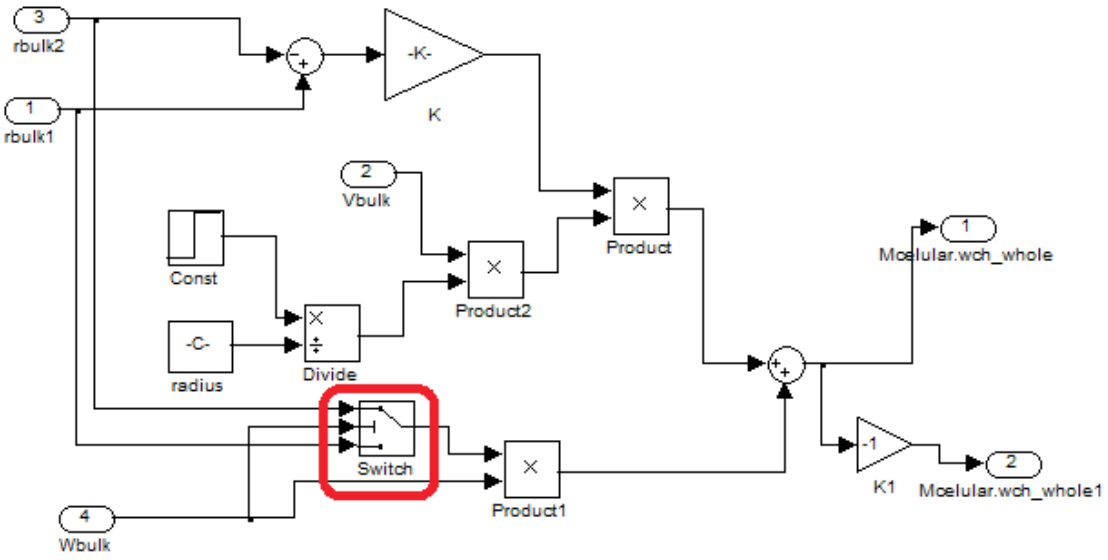


Figura 5.10. Membrana modelada en Simulink.

Para ello hay que programar el bloque SWITCH para darle la condición.

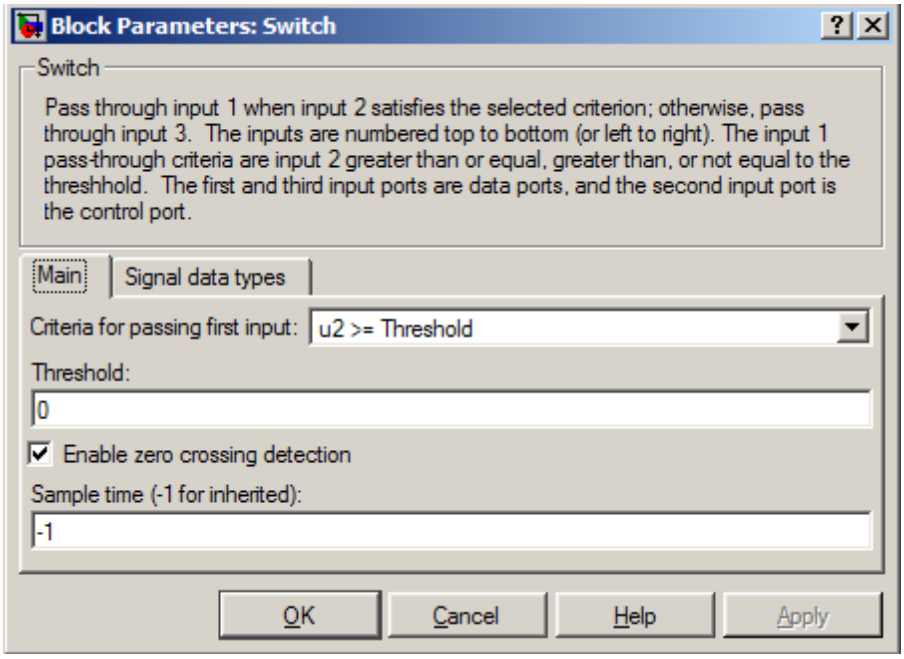


Figura 5.11. Parametrización del bloque SWITCH en Simulink.

Este bloque ya fue usado para proporcionar los flujos adecuados en cada una de las fases de la sesión de HD, teniendo como condición el tiempo de simulación.

Puesto que los flujos proporcionados en la fase de HD y en la fase de rebote son condiciones de frontera, el modelado de este tipo de condiciones en EcosimPro se realiza en la parte del experimento, o código de simulación. En esta parte además han sido instanciadas las funciones *volhd* y *volreb* cuyos resultados son usados en los flujos.

También se permiten modificar valores en los datos, siempre que no sean datos que necesiten cerrarse antes de la generación del experimento.

```
BOUNDS
-- Caudal de sangre al dializador
   Dializador.bi.wbulk[Single] = BloodflowToDial()

BODY

-- Datos ligados a la etapa de simulacion (HD)
   Dializador.wuf = vuf/thd        -- flujo de ultrafiltrado

-- Caudal total a traves de la mem celular
   Mcelular.mi.wbulk = vcscs[1]/tau_celular*exp(-TIME/tau_celular)

-- Caudal a traves de la mem vascular
    Mvascular.mi.wbulk = vp[1]/tau_vascular*exp(-TIME/tau_vascular)
-vuf/thd

   TIME = 0
   TSTOP = thd
   CINT = 60
   INTEG()


 -- Modificacion de datos en el dializador
 Dializador.wuf = 0 -- flujo de ultrafiltrado (promedio).

-- Caudal total a traves de la mem celular
          Mcelular.mi.wbulk   =   vcscs[2]/tau_celular*exp(-(TIME-
thd)/tau_celular)

-- Caudal a traves de la mem vascular
           Mvascular.mi.wbulk   =   vp[2]/tau_vascular*exp(-(TIME-
thd)/tau_vascular)

-- flujo de entrada al dializador
 Dializador.bi.wbulk[Single]=0
```

```
CINT = 60
TSTOP = thd+treb
INTEG()
```

Para modelar en Simulink cambios de valores en condiciones de frontera, entradas y en datos usamos un tipo de generador de señal.

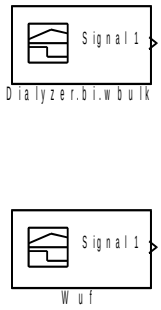


Figura 5.12.
Generadores de Señales
en Simulink.

Este tipo de generador de señal puede proporcionar una señal con la forma que el usuario prefiera. En nuestro caso, se trata de una onda cuadrada con dos valores constantes en distintos espacios de tiempo.

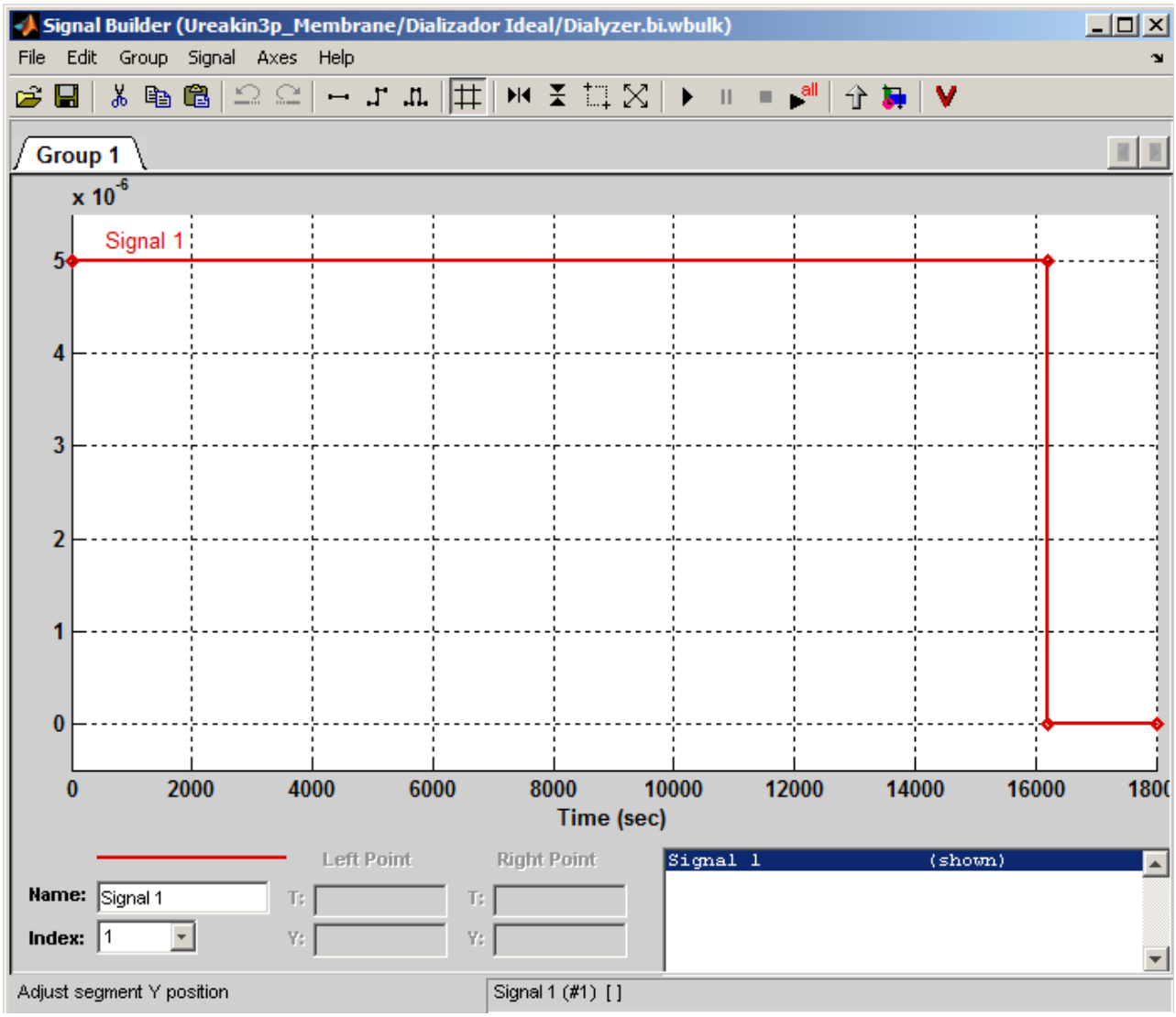


Figura 5.13. Condición de frontera implementada en Simulink mediante Generador de Señal.

En Ecosimpro tenemos dos tipos de eventos, uno automático que viene modelado en el interior de los elementos, código de modelo, y otro tipo de evento que es modelado manualmente en el código de simulación. Se modela en el experimento y por lo tanto se puede modificar fácilmente, esto maximiza la reusabilidad de los componentes ya que los eventos no tienen que ser incluidos a priori.

Vemos como Simulink tiene varios sistemas para la detección de eventos dependiendo del tipo de evento, pero no tienen una separación entre modelo y simulación del mismo, por lo que todos los eventos tienen que ser modelados a la misma vez que el resto del modelo, limitando la reusabilidad del mismo.

Como quinto y último aspecto metodológico compararemos la causalidad matemática impuesta en ambos entornos.

Esta es una característica metodológica de la etapa de partición del sistema que define la dirección de las variables matemáticas intercambiadas entre los componentes resultantes [15]. Esta técnica limita fuertemente la reusabilidad de los modelos creados.

Los sistemas biológicos son particularmente sensibles a este asunto [16].

Ya hemos visto como la causalidad en Simulink debe ser impuesta por el usuario, pues todo el sistema tiene sólo una forma de funcionar, calculando las variables que el usuario desee a partir de otras variables aportadas por el mismo. No existe separación entre el modelo y la estructura de cálculo del programa, así que todo debe ser modelado de forma que la estructura de cálculo que siga el programa no puede ser modificada. Esto lo hace Simulink durante la fase de compilación, imponiendo un orden en los bloques para su posterior cálculo.

Las variables de cada elemento o subsistema se dividen en entradas, o condiciones de frontera y salidas, el resto de variables tienen que ser calculadas a partir de estas entradas. Esto puede provocar lazos algebraicos en sistemas físicos dinámicos.

Una de las maneras del usuario de modificar el orden de cálculo automático es la creación de subsistemas que sean calculados de forma atómica e independiente al resto del modelo, esto se utiliza para asegurarte que el subsistema sea ejecutado completamente antes de pasar a otro bloque.

En contraposición a los subsistemas virtuales, llamados así, ya que a la hora del cálculo son reducidos a los bloques por los cuales están compuestos, quedando ordenados de la forma más eficiente para el cálculo en conjunto del modelo, a este proceso se le llama aplanamiento e la jerarquía.

En el caso de nuestro modelo UreaKin3p implementado en Simulink mediante bloques virtuales, si observamos el orden asignado por el motor de Simulink, es un orden donde se mezclan bloques de distintos subsistemas.

Si modificamos los subsistemas creados, de forma que todos sean tratados como unidades atómicas, activando el comando “Treat as atomic unit” nos encontramos con un problema de lazo algebraico, aún activando la opción “Minimize algebraic loops”, ya que algunos subsistemas necesitan los resultados de otros para poder calcular sus salidas.

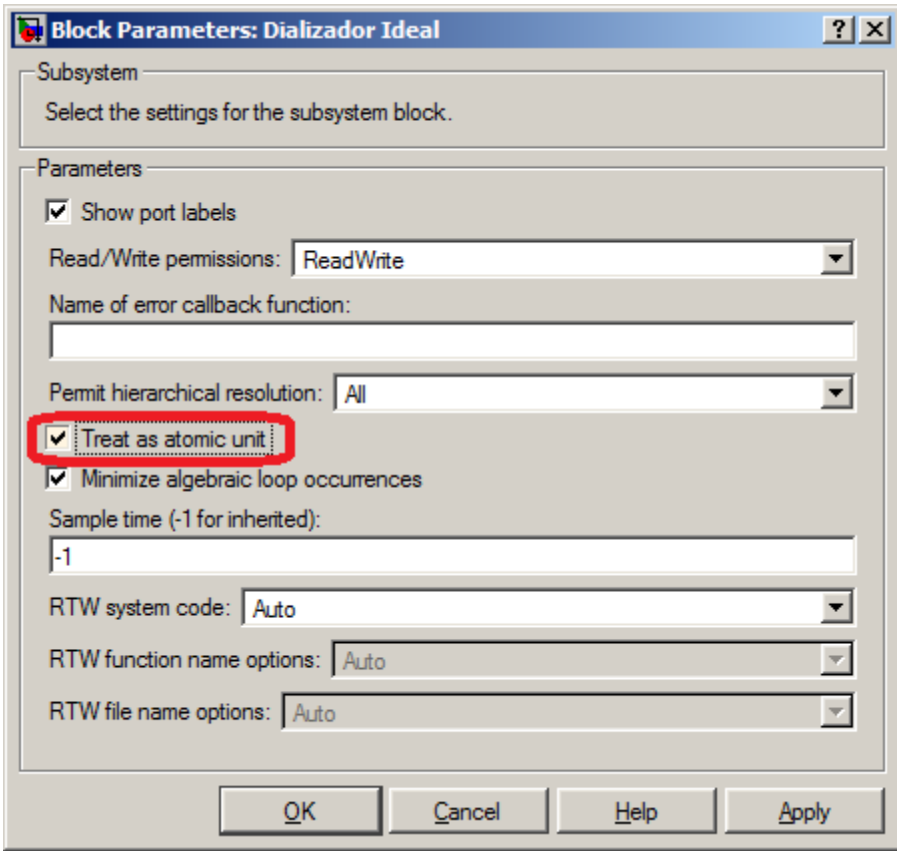


Figura 5.14. Opciones de Subsistema en Simulink.

La mayoría de estos lazos algebraicos son resolubles para Simulink, excepto uno el que involucra al elemento Dializador y Pool_Vascular, que pese a tener activado en modo “Warning” la detección de lazos algebraicos da error, siendo incapaz de resolverlo.

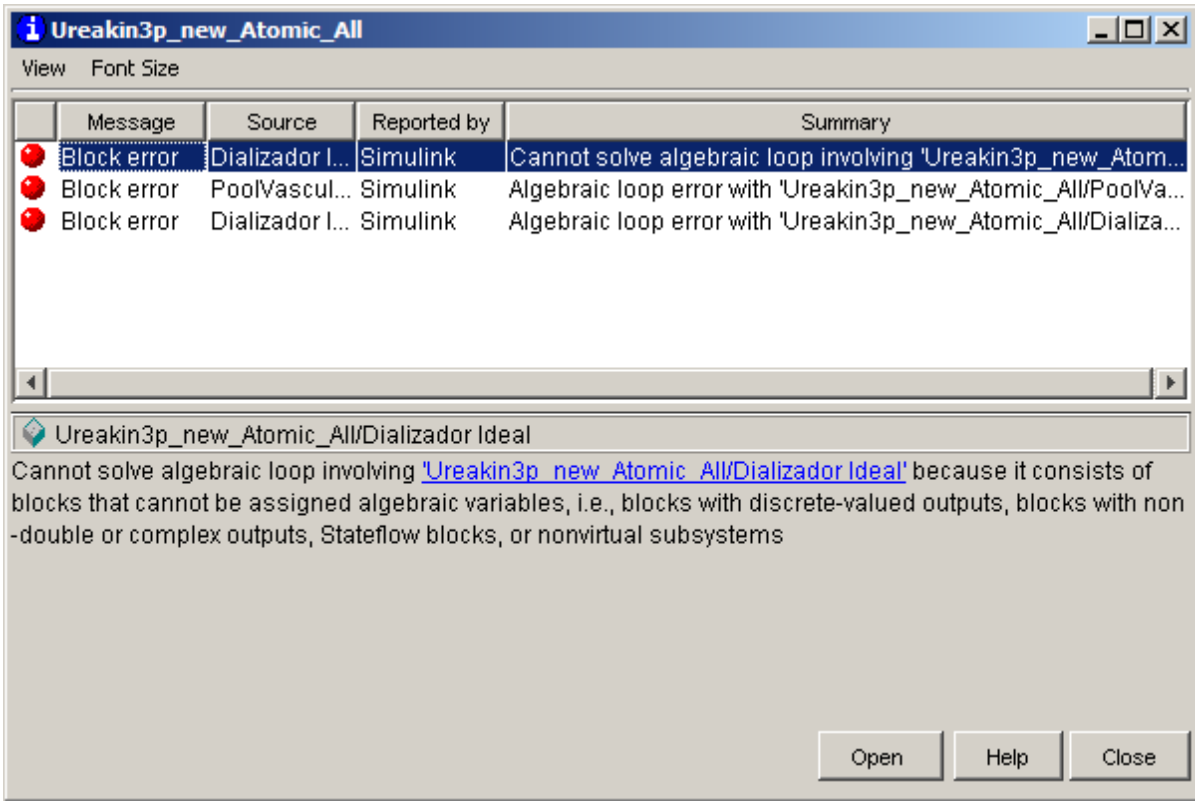


Figura 5.15. Mensaje de error mostrado por Simulink.

Este error sólo se puede solucionar tratando el subsistema Dializador como virtual. Pese a eso quedan varios lazos algebraicos resolubles pero que retrasarán el funcionamiento del sistema.

Así pues, al no existir separación entre modelo y estructura de cálculo en Simulink, la causalidad viene impuesta desde la definición del modelo.

Dentro del modelo tiene una importancia extrema si queremos hacer un particionado del sistema ,ya que este particionado puede verse afectado por problemas de lazos algebraicos si no está realizado de manera totalmente causal e independiente, tal y como hemos visto en el modelo UreaKin3p para subsistemas no virtuales.

Aunque desde el punto de vista del usuario final EcosimPro es un lenguaje acausal, como se indicó en el punto 2.1.2 en la parte de Tratamiento Matemático, para realizar cualquier simulación con un modelo es necesario convertir el conjunto de ecuaciones escritas por el usuario en otro conjunto de ecuaciones escritas de forma secuencial, donde cada línea contenga variables que hayan sido previamente calculadas. Este proceso se denomina asignación de la causalidad.

En EcosimPro si existe una separación entre el modelo y la estructura de cálculo del programa de forma que el usuario define un modelo y a la hora de crear la partición del modelo puede elegir cuales dentro de ese modelo que variables serán condiciones de frontera, el programa te sugiere algunas, y cuales serán explicitas o calculadas por el programa.

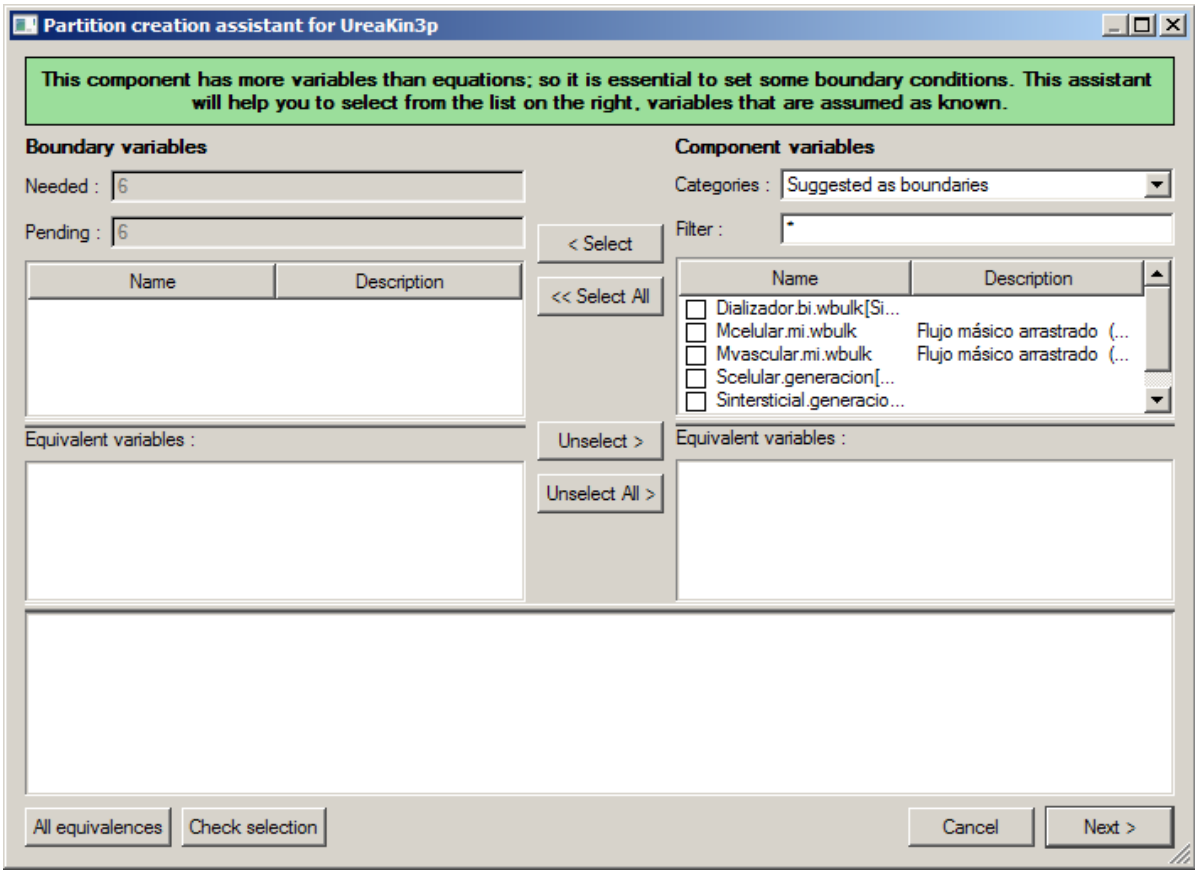


Figura 5.16. Asistente de EcosimPro para seleccionar las condiciones de frontera de un modelo.

Esto proporciona mayor flexibilidad y reusabilidad en los elementos creados, ya que las variables que contiene un elemento no tienen porque ser definidas como entradas y salidas, pero requiere mayor conocimiento del sistema y del modelo creado por parte del programador.

Para la asignación de la causalidad, los algoritmos matemáticos de EcosimPro llevan a cabo transformaciones tanto simbólicas (busca de variables, derivación simbólica) y numéricas (simplificación de coeficientes) utilizando la teoría de grafos para analizar modelos de ecuaciones y simplificar modelos complejos, así como detectar inconsistencias y redundancias [6].

Para definir la estructura de cálculo usa matrices, de esta manera se implementa la asociación de ecuaciones-variables. En primer lugar EcosimPro establece la matriz de incidencia original, M, donde se recoge por cada ecuación las variables que están presentes. Esta matriz esta formada por unos y ceros.

Luego la transforma en una matriz triangular inferior por bloques (BLT) para que todas las variables puedan ser resueltas a partir de las ecuaciones.

Mostramos la información general del modelo matemático UreaKin3p.

GENERAL STATISTICS

Info	#
Number of equations:	52
Number of boxes (coupled subsystems of equations):	0
Number of linear boxes:	0
Number of nonlinear boxes:	0
Number of EXPLICIT variables:	46
Number of DERIVATIVE variables:	6
Number of ALGEBRAIC variables:	0
EXPLICIT + DERIVATIVE + ALGEBRAIC variables:	52
Number of BOUNDARY variables:	6
Size of Jacobian matrix (DERIVATIVE+ALGEBRAIC):	6
Sparsity factor in Jacobian matrix (% of zeros):	58.3333333
Default integration method:	DASSL

El programa ha creado 52 ecuaciones que resolverá de forma secuencial a partir de las 6 condiciones de frontera dadas.

Vemos el acoplamiento del sistema en el dato referido como “Sparsity factor in Jacobian matrix”, que indica el número de ceros en la matriz de incidencia, por lo que muestra lo relacionado que están las variables y ecuaciones entre si.

Pueden surgir otros modelos en los cuales se cree un sistema no secuencial de ecuaciones o lazo algebraico, lo que EcosimPro llama “Linear Boxes” o si es no lineal, “Nonlinear Boxes”. Para ello el programa sugiere al usuario la definición de una variable algebraica para la resolución del sistema mediante iteraciones.

Podemos decir que EcosimPro es un programa que incluye un lenguaje no causal y resuelve los modelos matemáticos de forma semi-automaticamente (requiere una pequeña intervención del usuario seleccionando condiciones de frontera) causal.

5.1.1.2. Aspectos Numéricos.

Estudiaremos como primer aspecto numérico la precisión de los modelos UreaKin3p implementados. Para ello los resultados serán comparados con el modelo implementado en Matlab, que será tomado como referencia, ya que los errores serán debidos exclusivamente al particionado y al truncamiento.

Matlab utiliza el método ode45 para la integración del resultado el cual se basa en el método de Runge-Kutta utilizamos este mismo método para la simulación en Simulink y Ecosimpro, también se tomarán valores iguales de errores relativos y absolutos.

Para comparar el resultado de Matlab con los de los otros programas utilizamos un método de paso fijo de forma que los valores estén calculados en los mismos puntos para los tres programas. Así pues elegimos un paso fijo de integración en Matlab. Para ello usamos un vector de tiempos donde ya definimos el paso, para el sistema estudiado sabemos que tiene una dinámica lenta y un tiempo total de simulación largo (18000 segundos) por lo que usaremos un paso de 1 segundo.

```
tspan=t0:1:tfinal
```

En el caso de Simulink usamos el método de paso fijo (fixed step) ode4 y directamente ponemos el paso de integración igual a 1 segundo.

Comparamos el resultado de las concentraciones de los compartimentos de Simulink con el resultado de Matlab. Para ello evaluamos el error cometido entre el resultado de Matlab tomado como referencia y el resultado de Simulink.

Se prueba también con el modelo de Simulink tomando los bloques no virtuales, excepto el bloque Dializador, pero no hay ninguna diferencia en sus resultados.

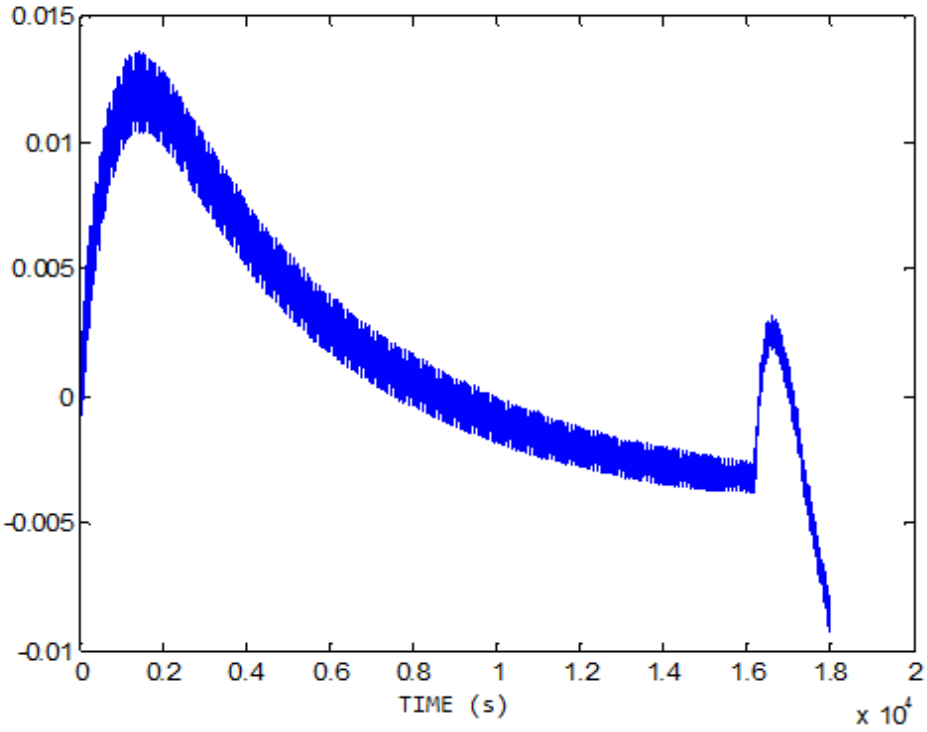


Figura 5.17. Error cometido por Simulink en las concentraciones en el compartimento Vascular.

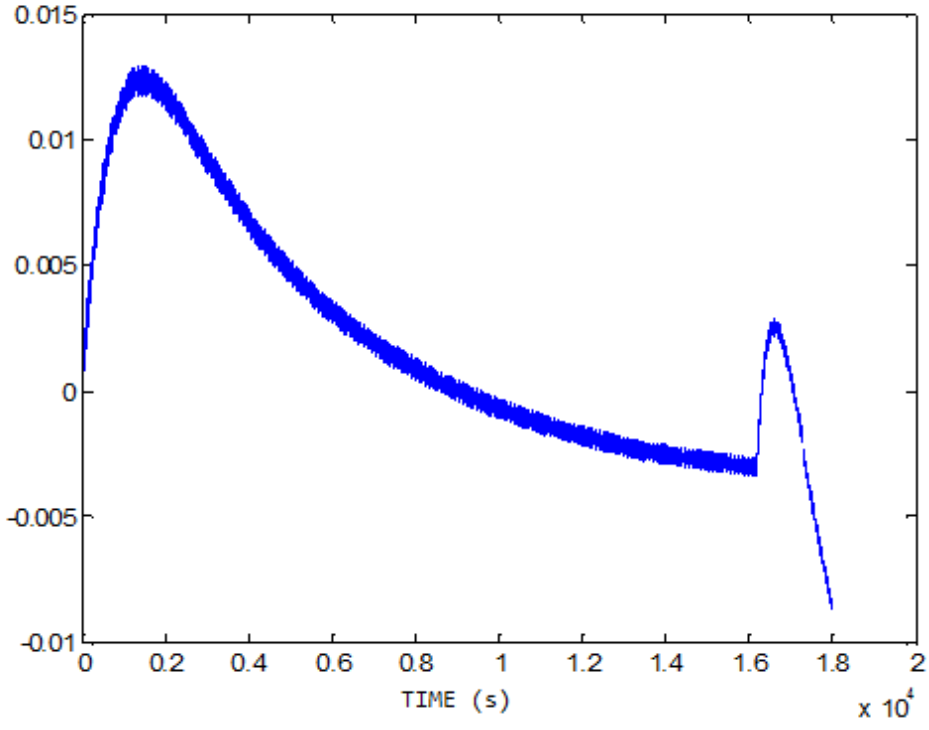


Figura 5.18. Error cometido por Simulink en las concentraciones en el compartimento Intersticial.

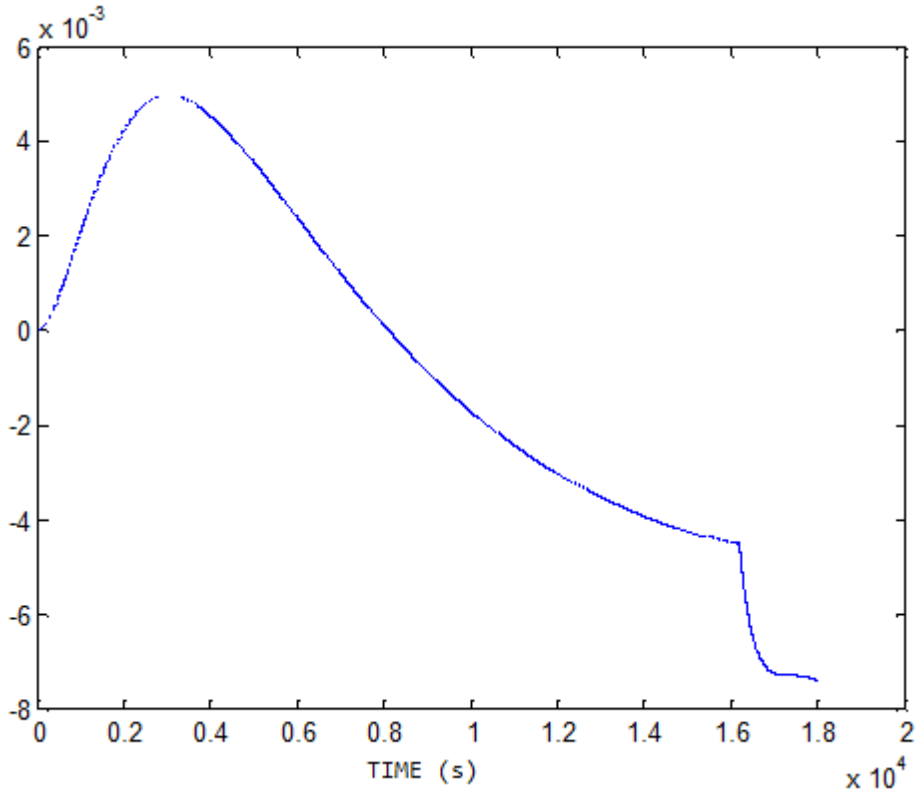


Figura 5.19. Error cometido por Simulink en las concentraciones en el compartimento Celular.

Analizamos por otro lado los errores de precisión de EcosimPro con el modelo de referencia de Matlab.

Para ello cambiamos el método de integración de EcosimPro, por el método Runge Kutta de orden 4, con un paso de integración de 1.

```
IMETHOD= RK4
CINT = 1
```

Esto nos da los siguientes resultados:

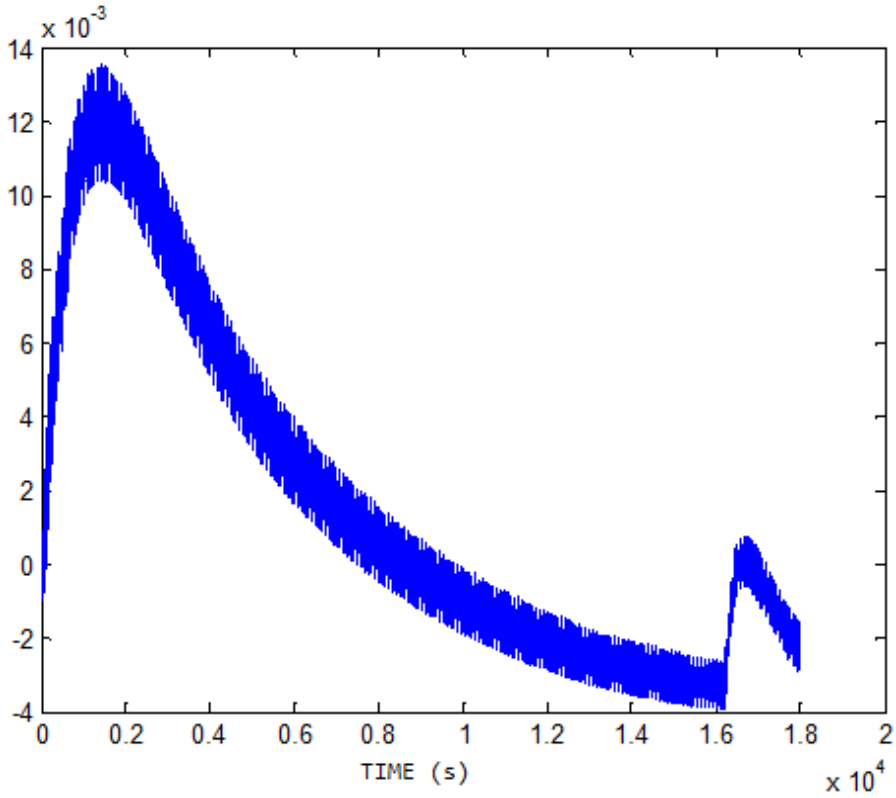


Figura 5.20. Error cometido por EcosimPro en las concentraciones en el compartimento Vascular.

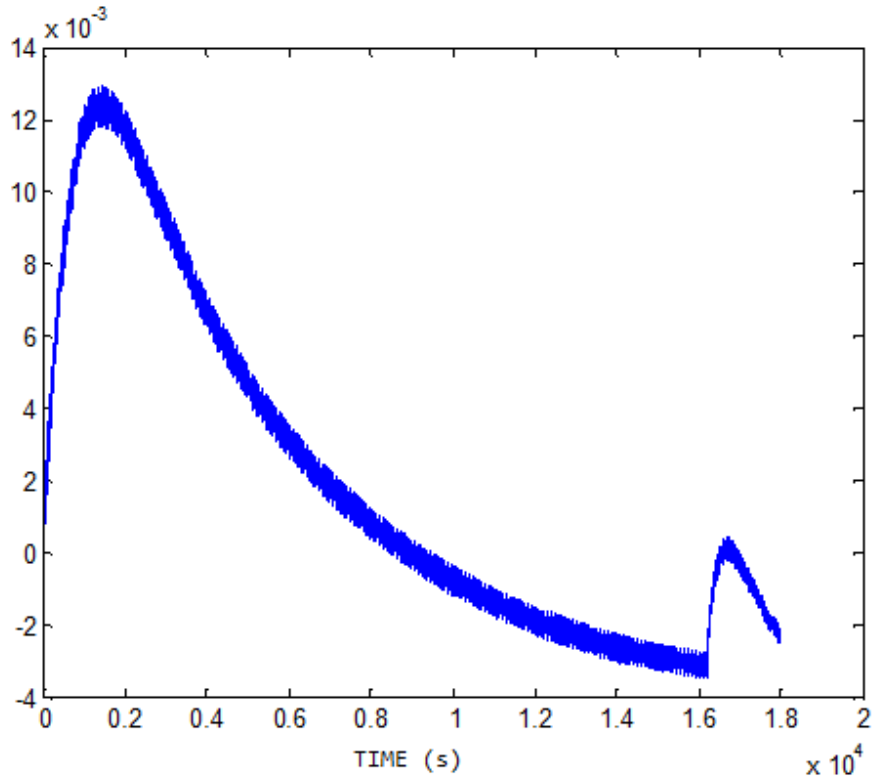


Figura 5.21. Error cometido por EcosimPro en las concentraciones en el compartimento Intersticial.

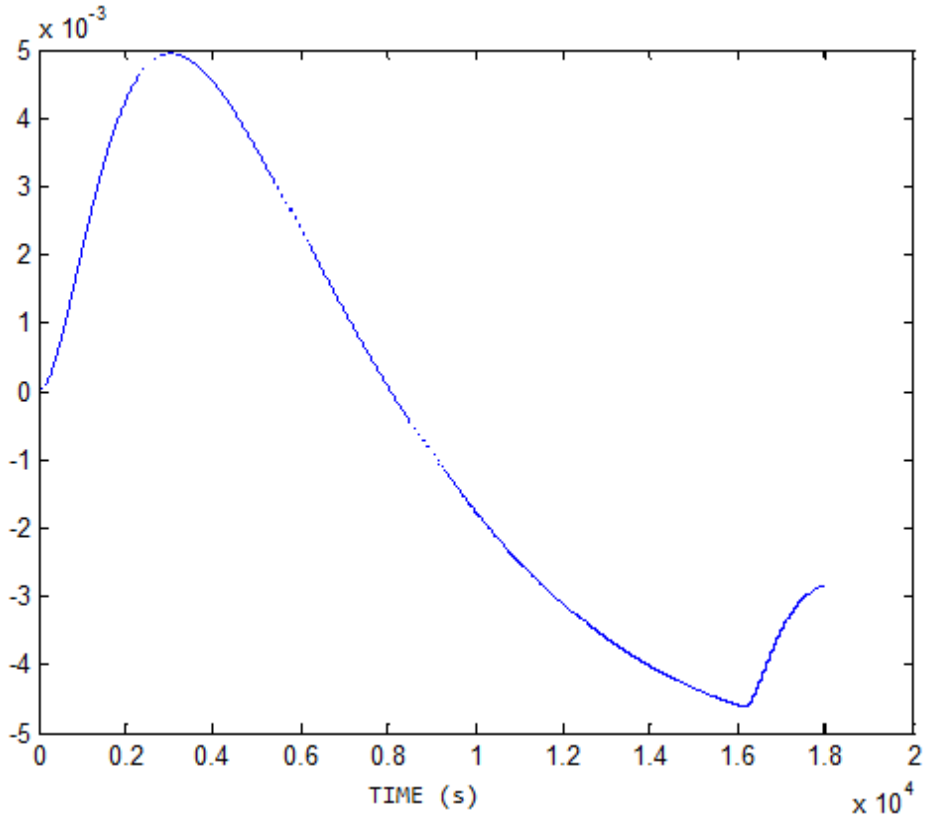


Figura 5.22. Error cometido por EcosimPro en las concentraciones en el compartimento Celular.

Si comparamos los errores cometidos vemos muchas similitudes y una gran diferencia a partir de la fase de rebote del experimento, pero no muy significativa debido a que sigue estando en los mismos órdenes de magnitud.

La mejora en la precisión de EcosimPro con respecto a Simulink después de la fase de rebote se debe a la detección del evento, ya que una vez detectado se comienza la integración de un nuevo sistema.

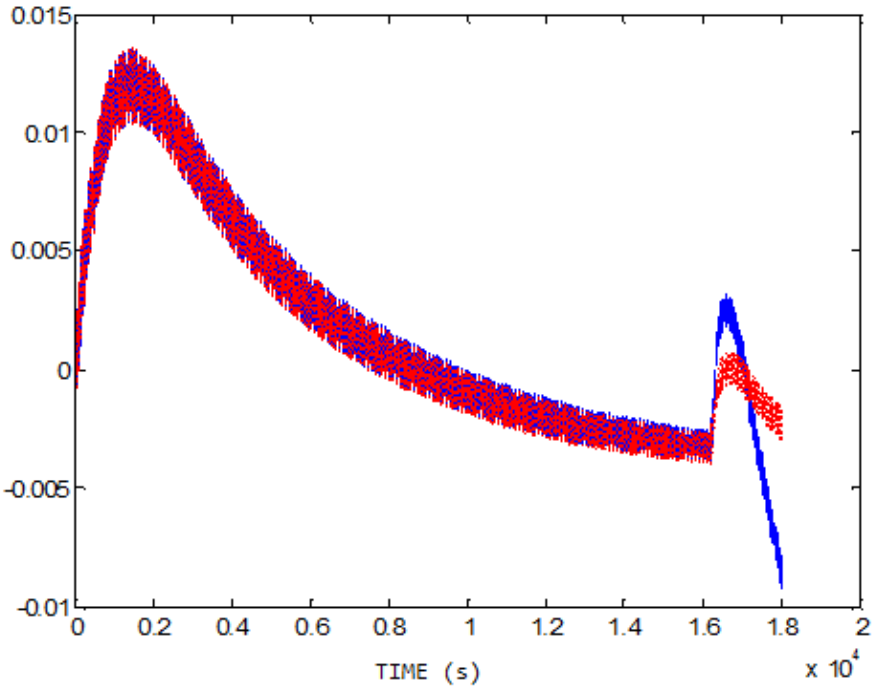


Figura 5.23. Comparación entre errores en Simulink (en azul) y EcosimPro (en rojo) en compartimento Vascular.

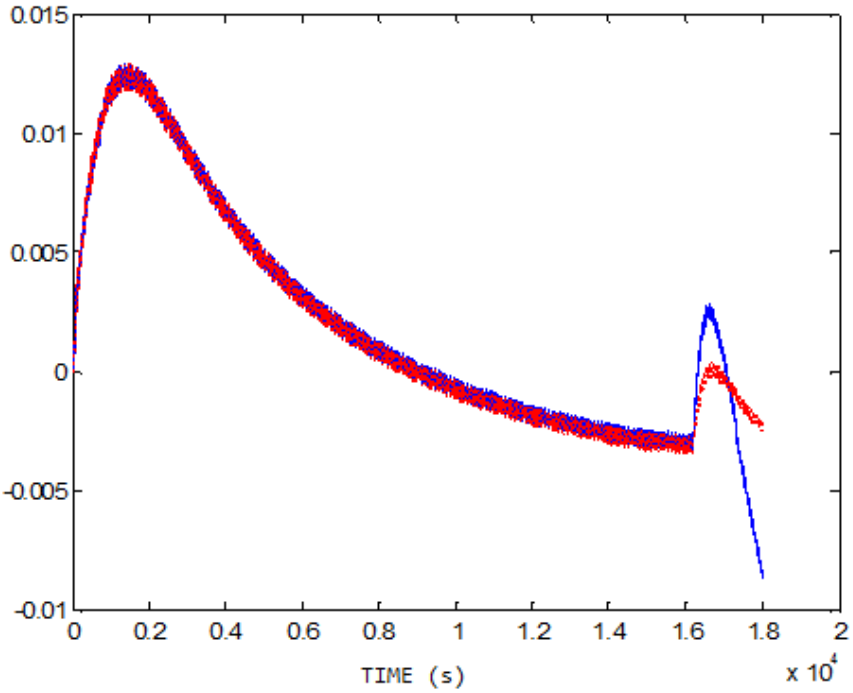


Figura 5.24. Comparación entre errores en Simulink (en azul) y EcosimPro (en rojo) en compartimento Intersticial.

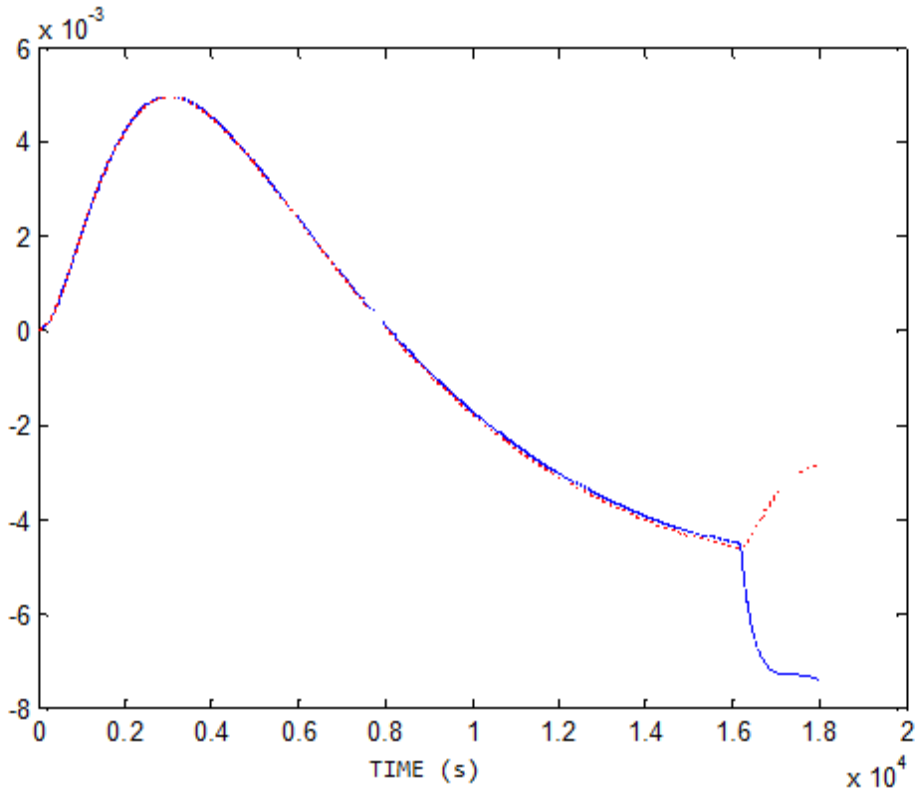


Figura 5.25. Comparación entre errores en Simulink (en azul) y EcosimPro (en rojo) en compartimento Celular.

El segundo aspecto numérico que analizaremos será el del tiempo de cómputo de las diferentes plataformas de simulación que estamos estudiando, Simulink y EcosimPro. Analizaremos su capacidad de cómputo bajo distintas condiciones de simulación. Es decir, cambios en el método y paso de integración, además de influencia de eventos y lazos algebraicos.

El equipo usado para estudiar los tiempos de cómputo consta de las siguientes características:

- Procesador Intel Core 2 Duo a 1,80 Ghz.
- 2 GB de Memoria RAM.
- Windows 7 Professional.

Empezamos evaluando los tiempos en condiciones normales de simulación en EcosimPro, estas condiciones normales se refieren usando el método de integración por defecto de EcosimPro, el método DASSL y usando el monitor de simulación que tiene EcosimPro para la fácil visualización de los resultados.

El paso de integración lo calcula EcosimPro como parte del método de simulación DASSL, pero con la variable CINT podemos controlar el intervalo de

comunicación y el número de resultados que se muestran en el monitor de simulación.

```
CINT = 60
Total Jacobian evaluations: 111
Total residues evaluations: 1723
Total processor time: 7.629 seconds
```

Cambiamos el intervalo de comunicación por un valor diez veces menor.

```
CINT = 6
Total Jacobian evaluations: 47
Total residues evaluations: 6459
Total processor time: 107.901 seconds
```

Esto produce un tiempo mucho mayor que el tiempo lógico de diez veces más. De nuevo volvemos a aumentar el intervalo de comunicación y efectivamente el tiempo empleado por el procesador es visiblemente inferior. A costa de una menor precisión en la gráfica representada por el monitor de simulación.

```
CINT = 120
Total Jacobian evaluations: 98
Total residues evaluations: 1299
Total processor time: 5.552 seconds
```

Si cambiamos el método de integración en EcosimPro de DASSL a RK4, por ejemplo, nos encontramos con que para este método necesita un CINT muy bajo para seguir siendo estable ya que no se trata de un método de resolución implícito como DASSL, lo que hace que el sistema sea lentísimo para su computación.

```
CINT = 3
Total Jacobian evaluations: 0
Total residues evaluations: 24006
```

```
Total processor time: 1880.25 seconds
```

Ahora usamos el método DASSL-SPARSE que no nos proporciona valores mejores de tiempo de cómputo para sistemas con una densidad de ceros en la matriz Jacobiana menor del 93%.

Nuestro sistema posee una densidad de:

Sparsity factor in Jacobian matrix (% of zeros): 58.3333333

Por ello no conseguimos un mejor tiempo de cómputo.

```
CINT = 60
Total Jacobian evaluations: 128
Total residues evaluations: 1656
Total processor time: 10.037 seconds
```

Ahora probamos la máxima velocidad del motor de Ecosimpro desactivando opciones que hacían que se ralentizara como por ejemplo la comprobación de funciones o generar un archivo con los datos de la simulación, además no usamos el monitor de simulación, sólo la simulación.

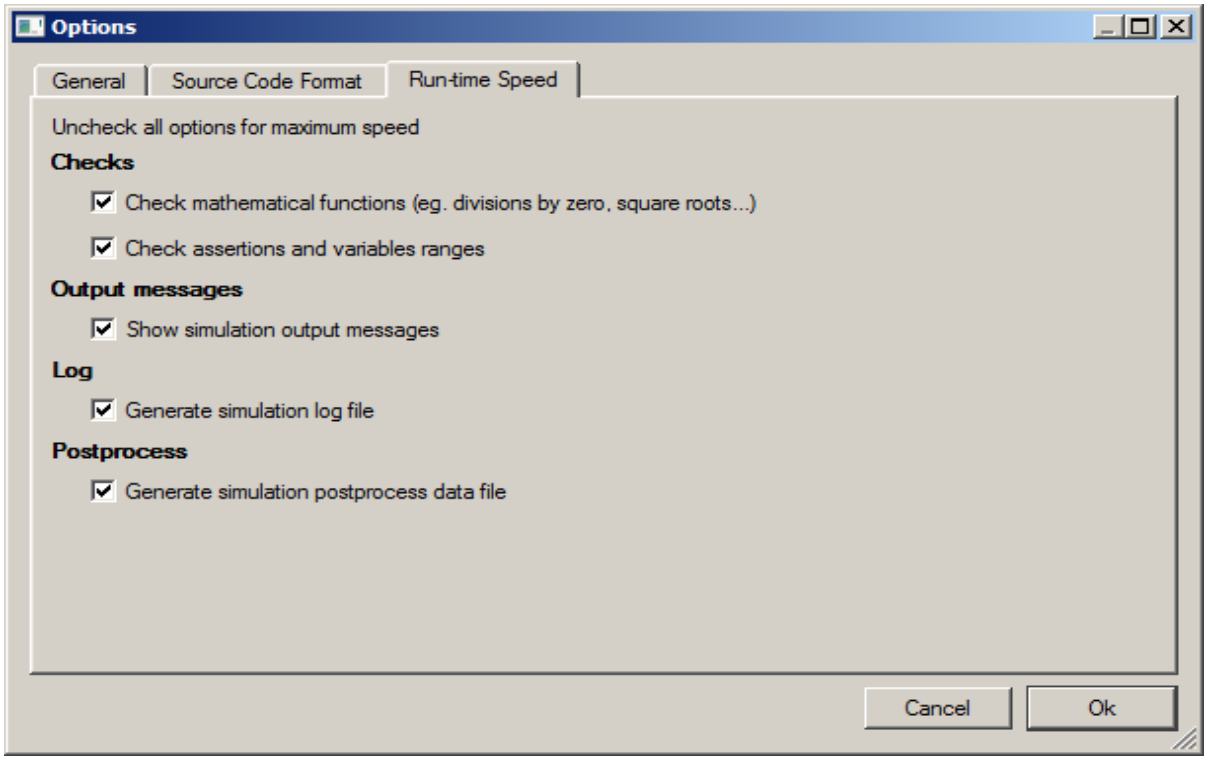


Figura 5.26. Opciones de simulación de EcosimPro.

```
CINT = 60
Total Jacobian evaluations: 111
Total residues evaluations: 1723
Total processor time: 0.484 seconds
```

Por otro lado medimos los tiempos de simulación empledos por Simulink, gracias a la herramienta "profiler" cuyo uso nos proporciona el tiempo de computación pero que también retarda el proceso. Aún así proporciona información útil.

Esta herramienta también nos informa del número total de veces que son invocadas las funciones:

- A nivel de bloque (ej. Output()) en Number of Block Methods.

- A nivel de sistema (ej. ModelExecute) en Number of Internal Methods.

- Novirtuales en Nonvirtual Subsystem Methods.

Usamos en las siguientes experiencias el método de integración estándar ode45 con un valor de paso máximo de 0.2 y demás valores máximos y mínimos y tolerancias de forma automática.

Total recorded time:	32.17 s
Number of Block Methods:	343
Number of Internal Methods:	10
Number of Nonvirtual Subsystem Methods:	5
Clock precision:	0.00000006 s
Clock Speed:	1800 Mhz

Simulink ofrece un sistema para aumentar la velocidad de simulación de tu sistema activando el modo "Accelarator". Este modo genera código que conexiona con otro motor de cálculo mediante matlab llamado C-MEX.

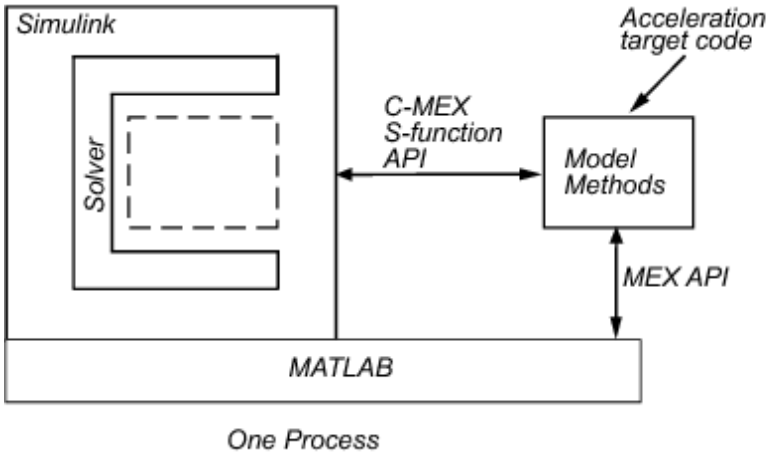


Figura 5.27. Esquema de conexión del módulo C-MEX con Simulink.

Total recorded time:	5.18 s
Number of Block Methods:	27
Number of Internal Methods:	5
Number of Nonvirtual Subsystem Methods:	5
Clock precision:	0.00000006 s
Clock Speed:	1800 Mhz

Este nuevo motor mejora el tiempo de cómputo y optimiza el sistema, al utilizar 27 métodos de bloque de los 343 que usaba antes. Aunque en su conexión con Matlab usa unos segundos no contabilizados en el tiempo de cómputo presentado.

Si usamos el modelo de Simulink con los subsistemas no virtuales, de forma que sean tratados de forma atómica en su cálculo, excepto el subsistema del Dializador, que ya vimos que no podía ser tratado así ya que creaba un lazo algebraico sin solución.

Total recorded time:	48.11 s
Number of Block Methods:	344
Number of Internal Methods:	10
Number of Nonvirtual Subsystem Methods:	51
Clock precision:	0.00000006 s
Clock Speed:	1800 Mhz

No hay un cambio significativo con respecto al modelo usado anteriormente con subsistemas virtuales. Aunque si un mayor tiempo de cómputo debido a la optimización que hace Simulink de los bloques incluidos en los subsistemas virtuales y que en este caso no puede realizar, eso aumenta tambien el número de invocaciones a las funciones de bloque, sobre todo el número de métodos novirtuales en subsistemas.

Esta falta de optimización y el tratamiento atómico de los subsistemas crean bucles algebraicos resolubles por Simulink, pero que le hacen perder tiempo de cómputo.

Si usamos el modo “Accelerator” se vuelve a optimizar el sistema, dando el mismo número de métodos de bloque, internos y novirtuales que teníamos antes. Lo que nos hace pensar que el uso del modo “Accelerator” anula los subsistemas tratados de forma atómica.

Total recorded time:	6.13 s
Number of Block Methods:	27
Number of Internal Methods:	5
Number of Nonvirtual Subsystem Methods:	5
Clock precision:	0.00000006 s
Clock Speed:	1800 Mhz

Ahora usaremos las opciones de optimización que nos ofrece Simulink.

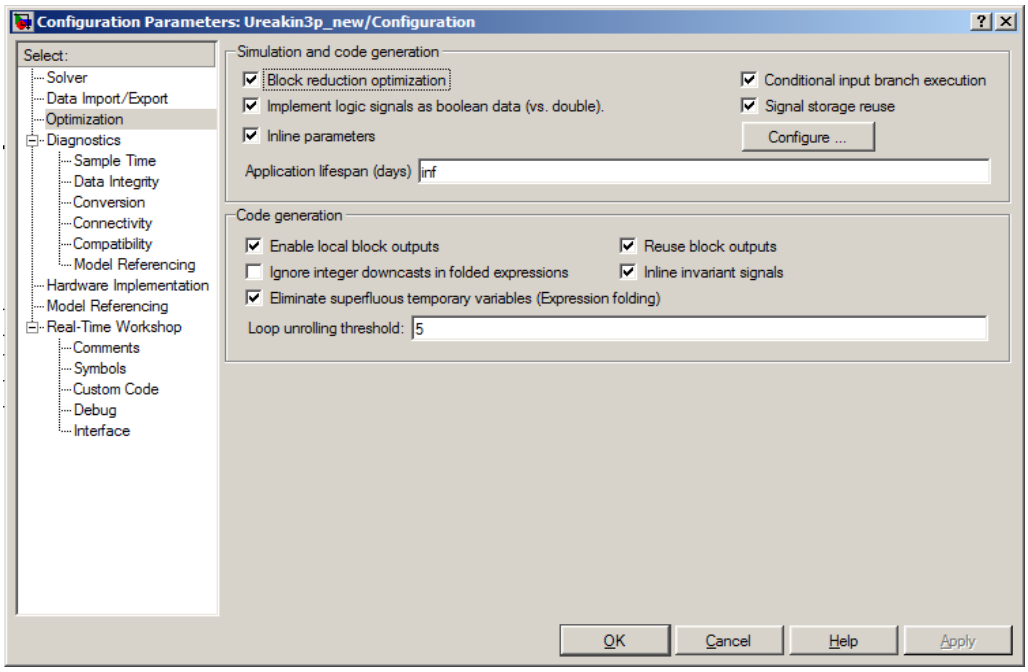


Figura 5.28. Opciones de optimización de Simulink.

Con estas opciones activadas el tiempo empleado es el siguiente:

Total recorded time:	27.72 s
Number of Block Methods:	336
Number of Internal Methods:	10
Number of Nonvirtual Subsystem Methods:	5
Clock precision:	0.00000006 s
Clock Speed:	1800 Mhz

Además de reducir el tiempo de cómputo también disminuye el número de invocaciones a funciones de bloque, pero no llega al nivel del modo “Accelerator”.

Para estudiar la influencia de los eventos en el tiempo de cómputo en Simulink usamos un modelo más simple que el UreaKin3p, por ejemplo un modelo de dos compartimentos pools con Agua y Urea en su interior divididos por una membrana en la cual actúan el mecanismo de difusión y de arrastre en el caso de flujo de solvente (wbulk), que es tratado como condición de frontera.

El evento que queremos que se active varias veces lo encontramos en el subsistema membrana y en el Bloque SWITCH, incluido para tener en cuenta el cambio de sentido del flujo de solvente en el proceso de arrastre.

Este modelo lo simularemos bajo dos condiciones para poder ver la influencia de los eventos: Condiciones de flujo de solvente nulo y condiciones de flujo de solvente cambiante entre positivo y negativo.

Estos cambios en las condiciones se hacen gracias al generador de señales de Simulink.

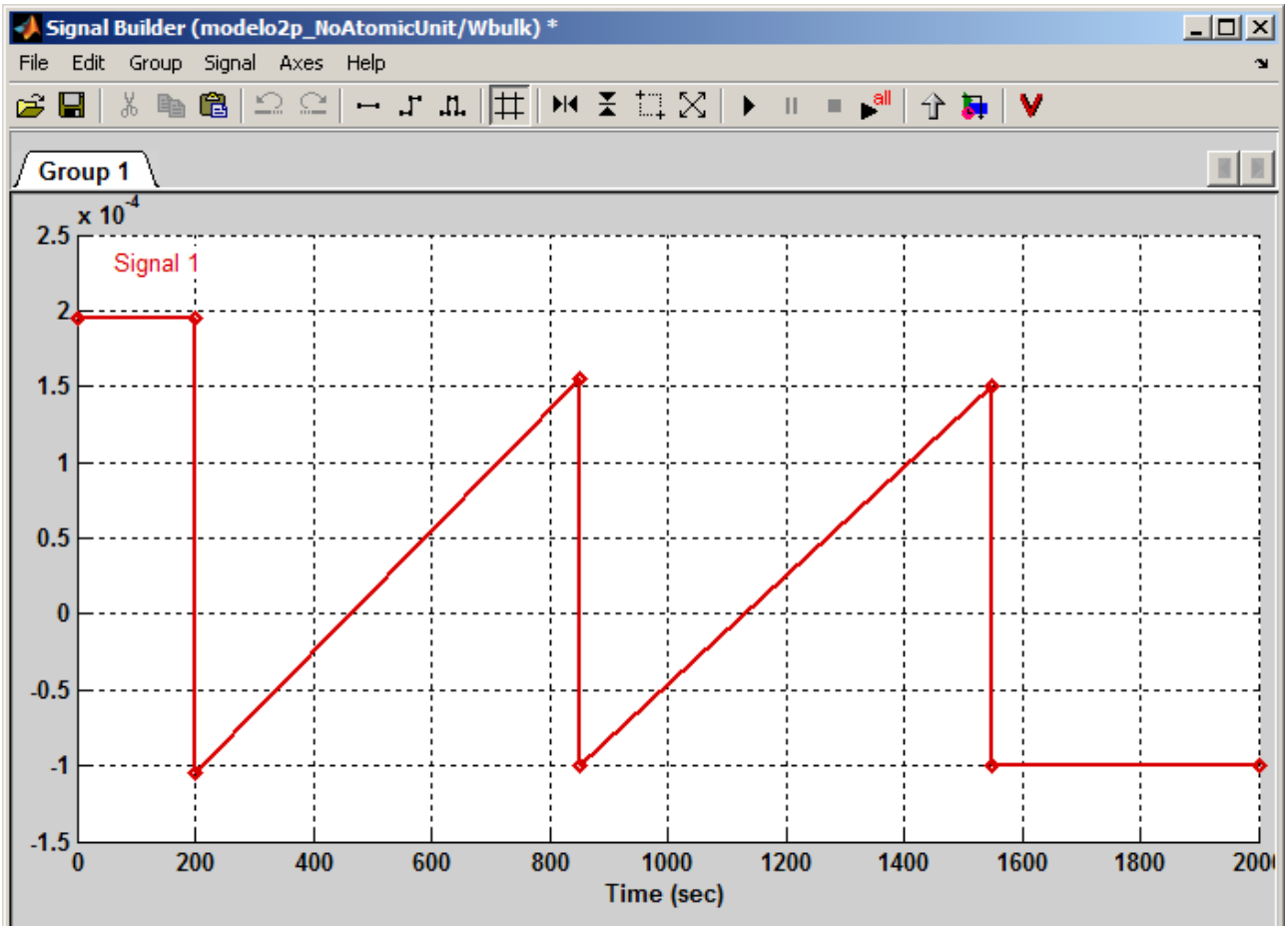


Figura 5.29. Señal de ejemplo generada en el bloque Generador de Señales de Simulink.

Se observan claramente como se tendría que activar el evento 5 veces, una por cada vez que pasa por 0.

El tiempo de cómputo para el modelo con condiciones de flujo nulas:

Total recorded time:	0.39 s
Number of Block Methods:	51
Number of Internal Methods:	10
Number of Nonvirtual Subsystem Methods:	5
Clock precision:	0.00000006 s
Clock Speed:	1800 Mhz

Y para condiciones de flujo cambiantes y por lo tanto simulación de eventos:

Total recorded time:	0.50 s
Number of Block Methods:	51

Number of Internal Methods:	10
Number of Nonvirtual Subsystem Methods:	5
Clock precision:	0.00000006 s
Clock Speed:	1800 Mhz

Es un tiempo de cómputo un 28,2 % más lento sin duda debido a los cinco eventos creados ya que el resto del modelo es idéntico, evaluando el mismo número de métodos de bloque, de sistema y no virtuales.

Se realiza el mismo experimento en EcosimPro para comprobar el tiempo de cómputo que emplea ante la detección de eventos. De nuevo se busca el cambio en el flujo dado como condición de frontera de forma que active el evento del elemento membrana. Se simula sin usar el monitor de simulación.

Ante flujo nulo, o lo que es lo mismo sin activación de eventos.

```
CINT = 10
Total Jacobian evaluations: 47
Total residues evaluations: 717
Total processor time: 0.153 seconds
```

Si aplicamos las mismas condiciones de frontera aplicadas anteriormente en Simulink.

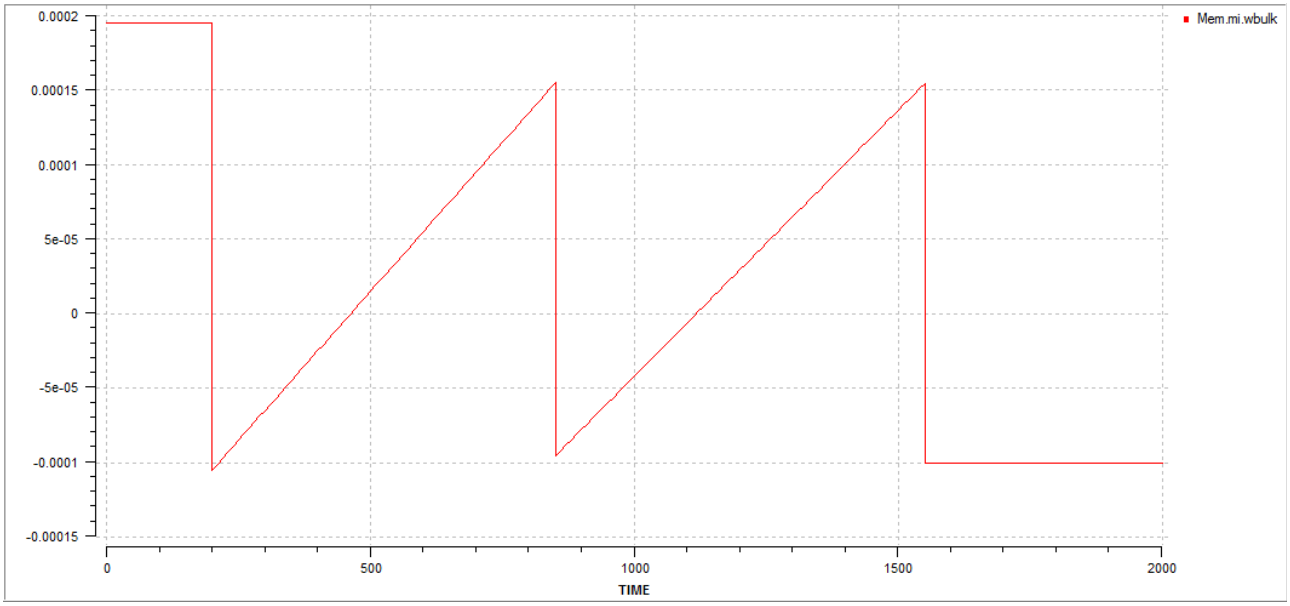


Figura 5.30. Señal de ejemplo generada en EcosimPro.

Los eventos son detectados por el monitor de simulación, algunos a la vez que se produce el cambio en las condiciones de frontera:

```
[TIME: 200]    End of integration (Jacobian evals: 27, Residues
evals: 253)
[TIME: 200]    Detected new events
```

Y otros son detectados automáticamente por el ZONE:

```
[TIME: 460]         Integration step 46
[TIME: 462.500028]  Detected new events (Mem.mi.wbulk >= 0 [TRUE])
```

El tiempo de cómputo total con la detección de los 5 eventos.

```
CINT = 10
Total Jacobian evaluations: 170
Total residues evaluations: 1537
Total processor time: 0.242 seconds
```

Esto supone un tiempo de cómputo un 36 % mayor que si no hubiera eventos, además de mayores evaluaciones del Jacobiano y de los residuos.

5.2. Aportaciones al desarrollo teórico e implementación sobre EcosimPro de una librería LibPK.

5.2.1. LibPK versión Alfa.

Se desarrolló primeramente la versión alfa donde se hace un primer desarrollo de los procesos y elementos de los cuales va a constar la librería.

El alcance de la misma se encuentra limitado a la descripción de sistemas farmacocinéticos, y por tanto considera:

- Fluidos incompresibles y newtonianos.
- Los fenómenos de transporte en las membranas biológicas están limitados a la difusión y a los fenómenos de arrastre, basados en flujos laminares en medios porosos.
- Modelos de parámetros concentrados.
- Las membranas son consideradas homogéneas. La descripción de la membrana incluye la propia capa límite.

La librería LibPK se subdivide en 5 librerías que contienen las distintas partes de la librería.

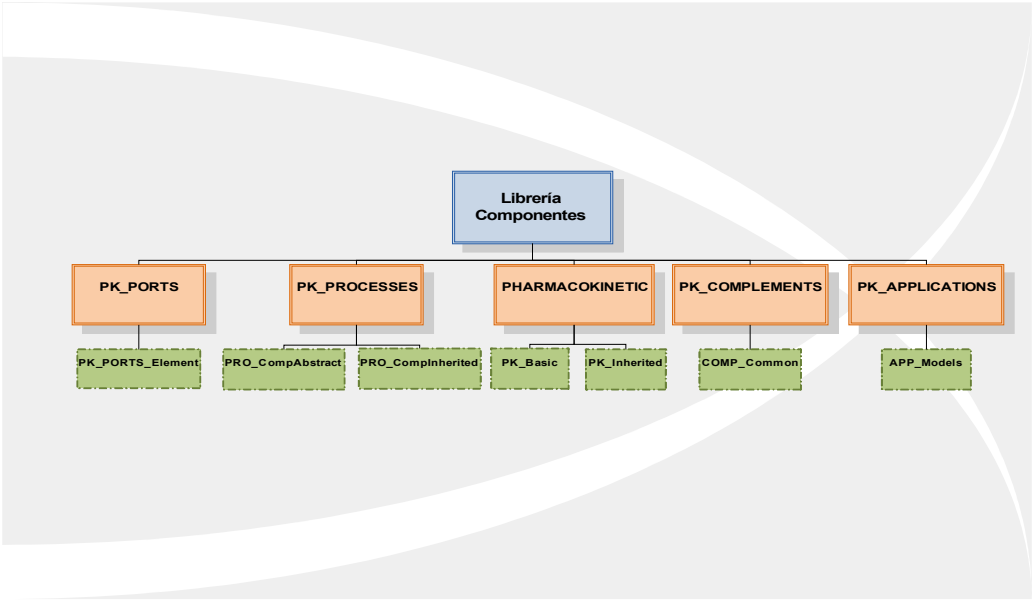


Figura 5.31. Estructura de las librerías de LibPk version Alfa.

- Pk_Ports. En esta librería se guardan los puertos que tenemos. En esta versión tenemos un puerto para cada proceso, uno para los elementos y dos auxiliares para la conexión de elementos externos.
- Pk_Processes. En esta librería se guardan los tipos enumerados, las funciones y los procesos como procesos padres en PRO_CompAbstract y los procesos hijos en PRO_CompInherited. Los procesos definidos en esta versión son: *Continuity, Diffusion, Convection*, *Elimination, Metabolic y Doses.*
- Pk_Pharmacokinetic. Guardamos en esta librería los elementos creados a través de los procesos por agregación y herencia. Definimos dos elementos padres, Pool y Membrana en el archivo PK_Basic y los elementos hijos en PK_Inherited.
- Pk_Complements. Se guarda en esta librería los elementos modelados externos a la fisiología. El único elemento externo definido es el *Dialyzer*.
- Pk_Applications. En esta librería es donde se guardan los modelos construidos con los elementos definidos en las otras librerías.

5.2.1.1. Aportaciones a pruebas de concepto sobre la versión alfa de la librería de LibPK.

Se han realizado varias pruebas de concepto sobre LibPK versión alfa con el fin de validar el cumplimiento de los requisitos funcionales principales: reusabilidad y modelado multinivel, a la vez que comprobamos la precisión en su predicción de procesos cinéticos y fisiológicos.

Se han definido dos escenarios:

- Modelo Urea3pk, ya validado con la librería KINETIC que servía como base de la librería LibPK versión alfa, el cual consiste en la descripción de la evolución en plasma y otros compartimentos de la concentración de toxinas urémicas durante una sesión de hemodiálisis. Para ello se usaba un modelo de 3 pools del organismo, de volumen variable con flujos hídricos definidos externamente.

Para la prueba de concepto sobre la librería LibPK también usaremos un modelo de 3 pools del organismos, pero incluyendo los eritrocitos como un pool interno al pool, “subpool”, que corresponde con el sistema vascular. Usando así un modelo multinivel.

Esta adición de tipos celulares en el interior del sistema vascular implica la restricción de la eliminación de los mismos en el proceso de hemodiálisis por parte del dializador.

Los resultados obtenidos son bastante parecidos al modelo de tres compartimentos sin eritrocitos.

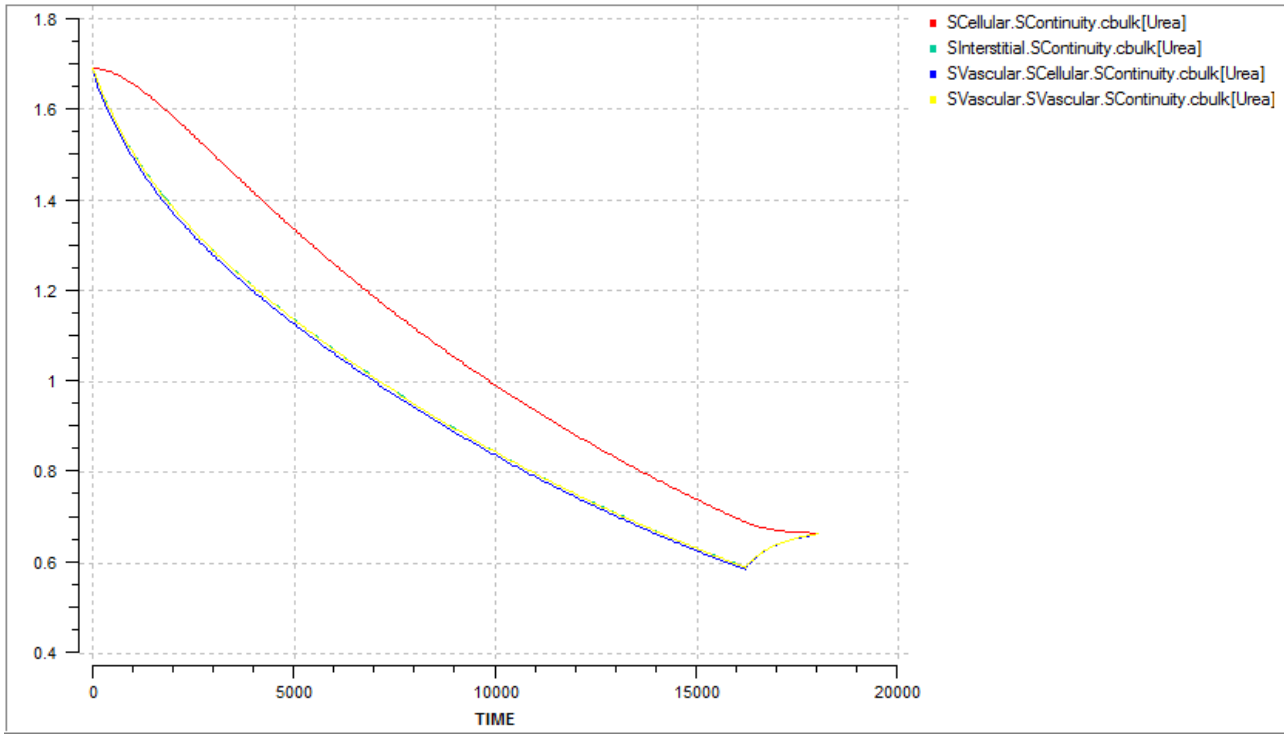


Figura 5.32. Concentraciones en los compartimentos: Vascular (amarillo), Eritrocito (azul), Intersticial (verde) y Celular (rojo), simulado con la versión Alfa de LibPk.

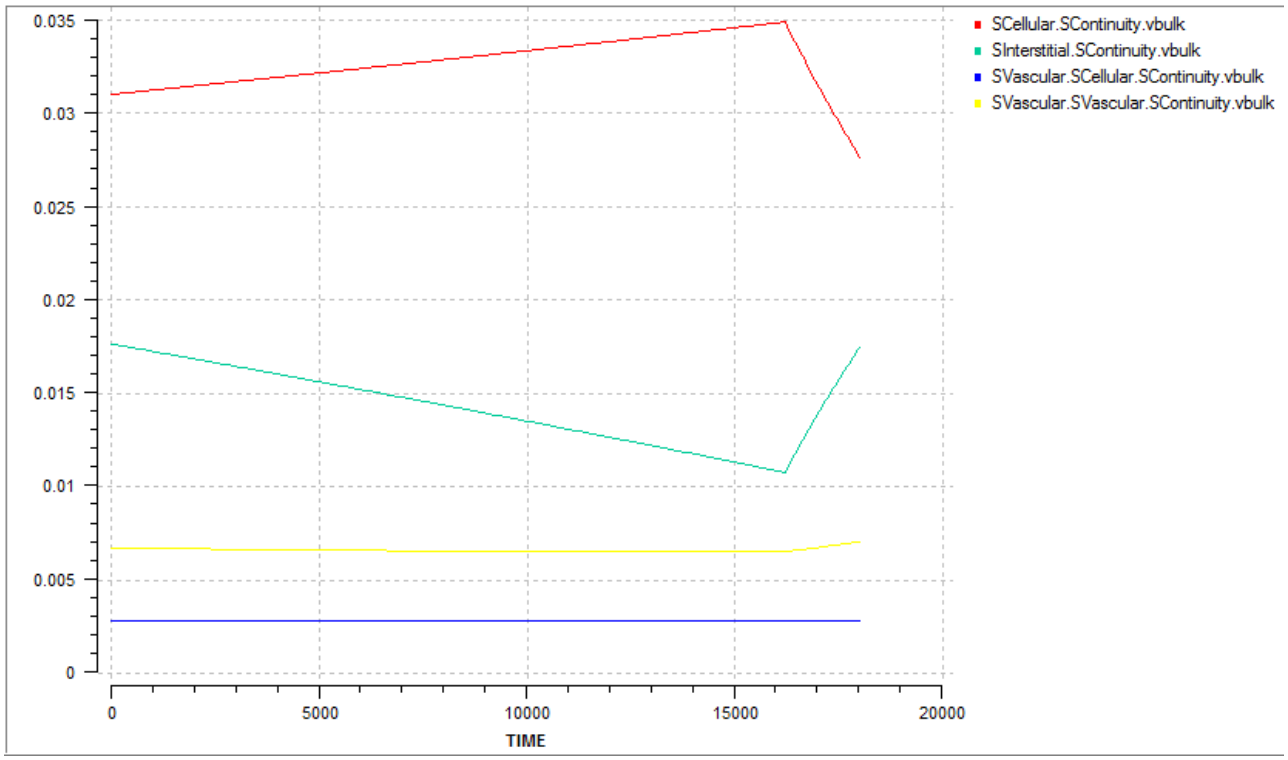


Figura 5.33.Volumenes en los compartimentos: Vascular (amarillo), Eritrocito (azul), Intersticial (verde) y Celular (rojo), simulado con la versión Alfa de LibPk.

- Estudio de la cinética de fármacos en el organismo mediante una prueba de concepto LADME[6] sobre diazepam y sus principales metabolitos: desmetildiazepam, temazepam y oxazepam.

El modelo será comparado con los resultados experimentales de un estudio clínico revisado de Meineke [17] y contrastado con el paquete matemático pk-engine, proporcionado por los autores del estudio.

El modelo del organismo se construye en la versión Alfa de LibPK como dos pools, uno accesible y otro no accesible, unidos por una membrana.

Dentro del pool accesible, o central, se declaran, con los datos aportados por el estudio clínico revisado de Meineke [17], algunos de los procesos LADME que van a utilizarse: *Doses, Elimination y Metabolic.* Y se conectan al proceso continuidad definido para todas las especies químicas que intervienen: diazepam, desmetildiazepam, temazepam y oxazepam.

El otro pool, no accesible o periférico, sólo tiene el proceso continuidad, pero la membrana que lo separa del pool central sólo permite la difusión del diazepam.

[6]Liberación, absorción, distribución, metabolismo y excreción.

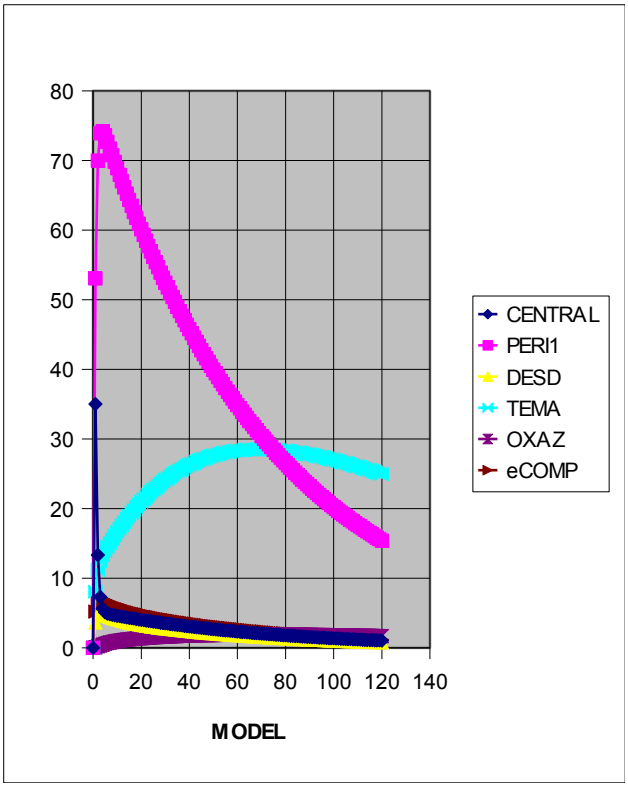


Figura 5.34. Concentraciones de las especies en el modelo de Meineke.

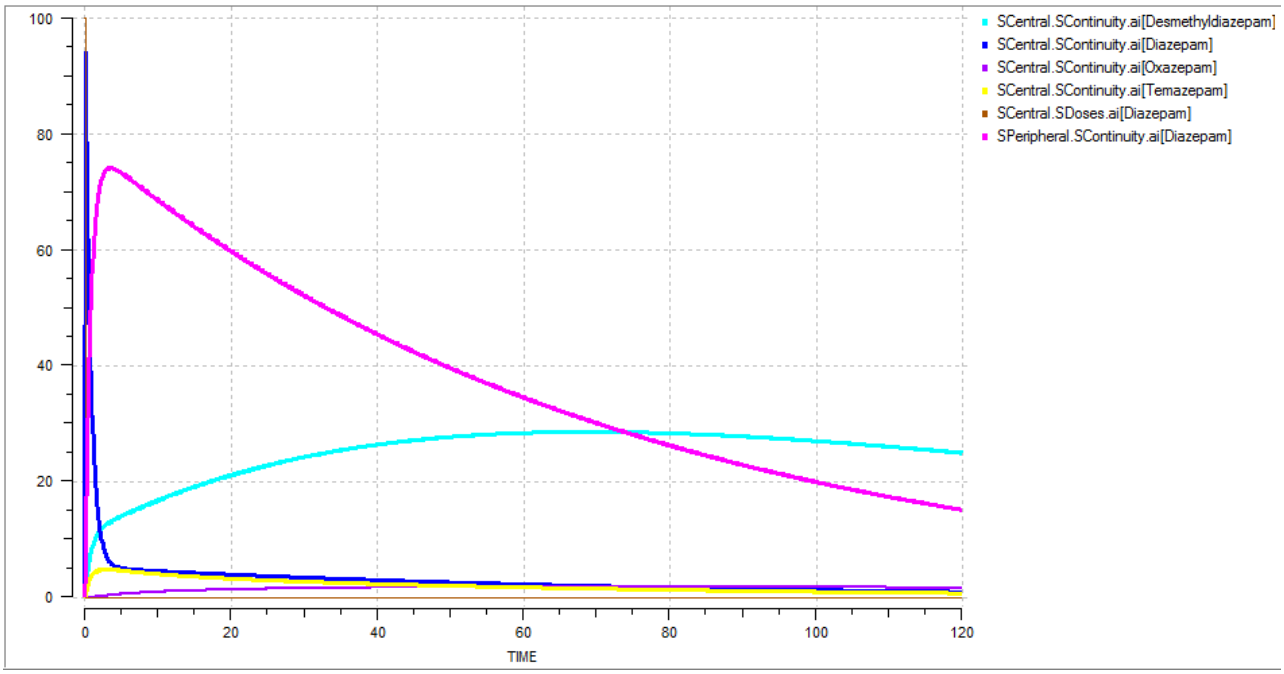


Figura 5.35. Concentraciones de las especies según la versión Alfa de LibPk.

Los resultados de simulación del modelo implementado en la librería LibPk versión Alfa son coincidentes con los presentados en la figura 5.34.

5.2.2. LibPK versión pre_Beta.

El alcance es prácticamente el mismo que en la versión anterior, sólo cambia el que se refiere a los fenómenos de transporte en las membranas biológicas que no sólo se limitan a la difusión y el fenómeno de arrastre sino que incorpora el movimiento por presiones osmóticas e hidráulicas.

En esta versión se mejoran varios aspectos de la anterior librería:

- En la versión anterior la definición de las cantidades de materia se daban en el proceso continuidad, *Continuity* a través de las concentraciones de los solutos en una disolución cuyo disolvente es el agua, *Water*. En la versión Beta se tratan todas las sustancias no como solutos y solventes sino como una mezcla de elementos químicos donde la concentración viene dada por el volumen específico de los propios elementos químicos. Aumentando la reusabilidad de los elementos y promoviendo la adición de conocimientos.
- Lo anterior también se aplica a las variables de flujo, rompiendo la separación que existía antes entre flujo de solvente y flujo de soluto y quedando únicamente el flujo másico.
- Creación de un puerto único "*pharma*" para todos los procesos y elementos. Esto es posible gracias a la unificación del flujo y como consecuencia nos quedan unos componentes capaces de la descripción multinivel más fácilmente.
- Desarrollo de interfaz gráfico para la fácil construcción de modelos por parte del usuario. También en modelos multinivel.
- La organización de la misma con sus sublibrerías también cambia.

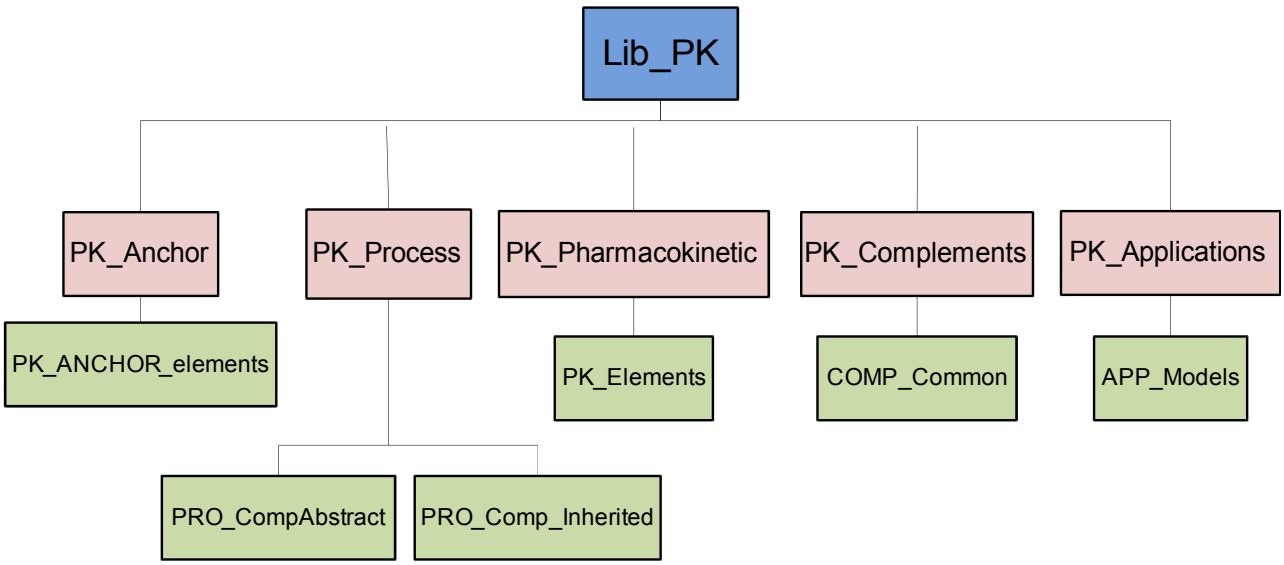


Figura 5.36. Estructura de la libreria LibPk version Beta.

- Pk_Anchor. Ahora en esta librería se guarda la definición de los tipos enumerados, la declaración de los grupos de tipos enumerados (SET_OF), los valores de las constantes globales (ej: R, constante de los gases) además del único puerto que se utilizará para todos los componentes: *phama.*
- Pk_Processes. Se modifican los procesos existentes *Continuity, Convection, Diffusion y Elimination* y se añaden nuevos como *Generation, Liberation, Absortpion y Metabolism.*
- Pk_Pharmacokinetic. Se modifica la estructura dejando sólo un archivo donde se definen los elementos a través de agregación de procesos: *Pool_Base, Membrana, Cellular_Pool, Intersticial_Pool y Vascular_Pool.* Se definen directamente nuevos elementos como *Mixer y Collector* necesarios para la conexión con el Dializador.
- Pk_Complements. El elemento *Dialyzer* es modificado traspasando parte de la dinámica al elemento *Mixer*.
- Pk_Applications. Se guardan en esta librería los modelos construidos.

5.2.3. LibPK versión Beta.

Tal y como hemos visto en el apartado anterior, el diseño de la versión Beta de la librería parte del diseño de la versión alfa con algunas modificaciones para aumentar su reusabilidad y su capacidad de descripción multinivel de forma que los elementos de la librería puedan ser construidos tal y como se muestra en la figura 4.9.

Aspectos globales:

- Uno de los aspectos claves ha sido la definición de un único puerto "pharma".

```
PORT pharma (SET_OF (biochemical) ChemicalComp)
SUM REAL wbulk[ChemicalComp]  UNITS "kg/s"   "Mass flux of solutes"
EQUAL REAL vbulk              UNITS "m**3"    "Bulk volume"
EQUAL REAL cbulk[ChemicalComp]UNITS "kg/m**3""Bulk concentrations"
EQUAL REAL phid               UNITS "Pa"      "Hydraulic pressure"
END PORT
```

Tenemos 3 variables causales cbulk, wbulk[] y phid que son las mínimas necesarias para la descripción de los procesos de transporte en membranas semipermeables definidos. La variable volumen se aporta ya que se utiliza para el cálculo del área de transferencia de la membrana.

- Se define una función que devuelve el volumen específico de un elemento químico, necesario para el cálculo de concentraciones.
- Se define una nueva geometría para las membranas, Multiesférica, necesaria para la definición de los eritrocitos en el compartimento vascular.

```
{Spheric,Cylindric,Multi_Spheric}
```

Procesos:

Primero describimos los procesos base, los cuales han sido declarados abstractos de forma que no puedan ser modificados ni utilizados de forma irregular, estos procesos base son heredados por los procesos a los cuales se les agrega el puerto o los puertos quedando elementos utilizables.

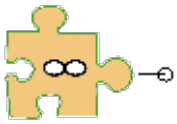
Los dividimos en 3 grupos:

- Procesos de pool. Tienen sólo un puerto para su conexión con el exterior.

Continuity. Este proceso esta basado en la ley física de la conservación de la masa. Se aplica para un grupo de compuestos químicos con una distribución homogenea en el volumen de control.

[Cambio de masa en el sistema]= [Generación neta]+[Flujo neto]

También proporciona los valores de concentración para cada especie química y el volumen total de la mezcla, a través de la función volumen específico.



Generation: Proceso que representa la creación anabólica de una especie química sin reacción química involucrada, o cuya reacción no se conozca o no sea relevante. Se usa una ecuación kinetic de primer orden:

[Flujo de compuesto químico generado]=[Coeficiente de generación][Masa del compuesto químico]*

Elimination: Proceso basado en el proceso LADME eliminación, en el cual los residuos del metabolismo y otros materiales inservibles son eliminados del organismo a través de pulmones, riñones o piel. Se usa una ecuación kinetic de primer orden:

[Flujo de compuesto químico eliminado]=[Coeficiente de eliminación][Masa del compuesto químico]*

Degradation: Representa el catabolismo de una sustancia química sin reacción química involucrada, o cuya reacción no se conozca o no sea relevante. Se usa una ecuación kinetic de primer orden:

[Flujo de compuesto químico degradado]=[Coeficiente de degradación][Masa del compuesto químico]*

- Procesos de membrana. Estos procesos tienen dos puertos para la conexión de dos compartimentos.

Diffusion. Proceso basado en la primera ley de Fick. Calcula el transporte difusivo del soluto debido al gradiente de concentraciones en las superficies de la membrana.

[Flujo másico de soluto]=[Permeabilidad difusiva][Gradiente de concentraciones]*

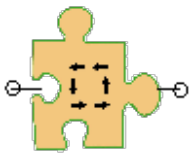
Se consideran solutos todos los compuestos químicos menos el agua, que es considerada solvente.

Convection. Representa el flujo de compuestos químicos debido al diferencial de presiones (hidráulica y osmótica) a través de la membrana y el flujo de soluto debido al arrastre de los mismos. Consideramos membrana porosa, por lo que existe una fracción de la materia que la atraviesa que es rechazada.

Sólo se considera flujo laminar en el interior de la membrana, por lo que la concentración en el interior depende del sentido del flujo volumetrico.

[Flujo volumétrico de solución]=[Permeabilidad hidráulica]([Gradiente de presiones hidráulicas]-[Gradiente de presiones osmóticas]*[Coeficiente de rechazo de stavermann])*

[Flujo másico de solutos]=[Concentración en la membrana](1-[Coeficiente de rechazo de stavermann])*Flujo volumétrico de solvente*



- Procesos LADME. Estos procesos se pueden considerar de pool ya que tienen sólo un puerto para su conexión con el proceso continuidad, excepto el proceso *absorption* que tiene dos puertos ya que se considera un tipo de membrana.

Metabolism. Este proceso representa el grupo de reacciones bioquímicas y procesos físico-químicos que tienen lugar en el organismo. Son conocidos como rutas metabólicas, donde los compuestos bioquímicos son creados y degradados en otros.

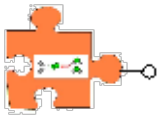
Se pueden usar reacciones sin metabolitos como:

$$A \rightarrow B$$
$$A + B \rightarrow C + D$$

El modelador diseña el tipo de ecuación gracias a las matrices de datos que tiene que aportar: km[i,j] y kd[i,j]. Estas matrices proporcionan dos tipos de flujo, el flujo catabólico y anabólico, cuya suma para cada compuesto químico proporciona el flujo total.

Para los cálculos de flujo se usa la ecuación general de kinetic:

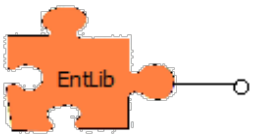
[Flujo de compuesto químico]=[Constante metabólica][Masa del compuesto]*



Enteral_liberation. Su dinámica está basada en el proceso LADME liberación, considerado de liberación gástrica, o lo que es lo mismo el compuesto químico entra al organismo vía oral o rectal. Utilizamos la ecuación general de primer orden de kinetic.

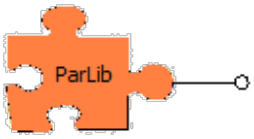
[Flujo de compuesto liberado]=[Constante de liberación][Masa del compuesto]*

También da posibilidad al usuario de definir un retraso.



Parenteral_liberation. Su dinámica también está basada en el proceso LADME liberación, pero se considera sólo la liberación por ruta parenteral o lo que es lo mismo los compuestos químicos entran directamente a través de la piel por inyección o perfusión.

Se permite al usuario una administración periódica de los compuestos químicos, para lo cual puede decidir definir el número total de dosis o el tiempo total de administración deseado.

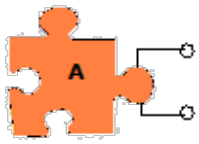


Absorption. Su dinámica esta basada en el proceso LADME absorción. El cual está asociado a una liberación enteral de un compuesto químico para su absorción sublingual, sublabial o gastrointestinal.

El proceso se formula con la ecuación cinética de primer orden, incluyendo un factor de biodisponibilidad que representa la fracción de compuesto químico rechazada por la membrana de absorción y excretada por el organismo.

[Flujo de compuesto absorvido]=[Constante de absorción][*factor de biodisponibilidad*]*[Área de la membrana]*[Masa del compuesto]*

También da posibilidad al usuario de definir un retraso.



Elementos:

La creación de elementos está pensada para que sea posible a través de la unión de los procesos y el o los puertos pharma mediante agregación, este era uno de los objetivos metodológicos de la librería. La unión puede hacerse gráficamente o mediante código.

Además los procesos que usamos para la construcción del elemento son parametrizables desde el elemento, por lo que no requieren un tratamiento previo.

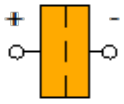
Hay algunos elementos auxiliares como el *Mixer* o el *Dialyzer* que se utilizan en el experimento de hemodiálisis.

Membrane. El elemento membrana se construye con la agregación de los procesos *diffusion* y *convection.* Por lo que encapsula la dinámica de esos procesos.

Así pues los mecanismos de transportes en la membrana solo consideran transferencia de masa orgánica y difusión en los poros de soluto. Además del flujo de solvente debido a la presión total (Osmótica e hidráulica) y el soluto rechazado en los poros de la membrana.

Se considera el particionado del elemento sugerido por la física por lo que no consideramos la resistencia de película en el límite.

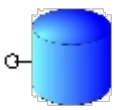
Se utiliza para la separación de compartimentos o espacios compartimentales, pools.



Pool_base. El elemento pool se construye con la agregación del proceso *continuity*.

Este elemento define un compartimento interno no específico del organismo, tratado como un tanque bien removido con volumen homogéneo.

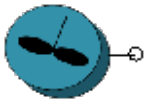
Se conecta a otros pools a través de los elementos membranas o también pueden ser conectados directamente a elementos externos.



Cellular_pool. Este elemento se construye con la agregación del proceso *continuity* y *metabolism*.

Define el compartimento interno celular del organismo, también se trata como un tanque bien removido con volumen homogéneo, al cual se le agrega un proceso de metabolismo típico del sistema real que representamos, las células del organismo.

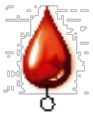
Se recomienda la conexión a otros compartimentos a través membranas o directamente a elementos externos.



Vascular_pool. Este elemento es construido con la agregación de los siguientes procesos: *continuity, generation y elimination*.

Define el compartimento interno vascular del organismo, también se trata como un tanque bien removido con volumen homogéneo, al cual se le agregan un proceso de generación y otro de eliminación típico del sistema real que representamos, el sistema vascular.

Se recomienda la conexión a otros compartimentos a través membranas o directamente a elementos externos.



Enteral_drug. Elemento que agrega el proceso *enteral_liberation*.

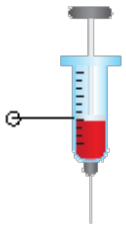
Describe la administración de un compuesto químico a través de una ruta enteral.

Se recomienda su conexión a un compartimento pool a través de una membrana de absorción.

Parenteral_drug. Este elemento agrega el proceso parenteral_liberation.

Describe la administración de un compuesto químico a través de una ruta parenteral, es decir, intravenosa, intramuscular, subcutánea etc.

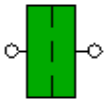
Su conexión es directa al compartimento pool, ya que la liberación es instantanéa.



Absorption_Membrane. Agrega el proceso Absorption.

Simula el flujo de compuestos químicos a través de una membrana de absorción, como podría ser el estómago, los intestinos, la boca etc.

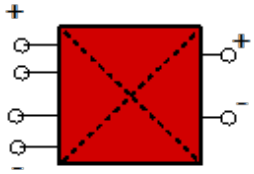
Se recomienda su uso para conectar el elemento liberación a un compartimento pool.



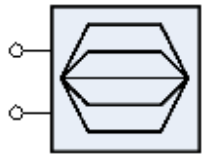
- Elementos auxiliares. Son elementos cuya creación no se hace a partir de procesos que son agregados, sino que son definidos in situ.

Mixer. Elemento auxiliar que no se corresponde con ningún sistema orgánico, pero que es necesario en el experimento de Hemodiálisis para el caso de que el modelador haya creado un compartimento vascular con células, eritrocitos, en su interior. En ese caso, el elemento *Mixer* se usa como adaptador en la conexión de este pool vascular con subpool interior al dializador.

El elemento *Mixer* se encarga de calcular los flujos entrantes y salientes del organismo en función de la fracción volumétrica de los compartimentos conectados.



Dialyzer. Elemento externo que contiene la dinámica de un dializador ideal. Se considera un modelo unidimensional del dializador. Su aplicación se restringe a una sesión de hemodiálisis con condiciones operativas constantes, es decir, su eficiencia se considera constante.



5.3. Discusión.

En la primera parte del proyecto realizamos un análisis metodológico y numérico del modelo UreaKin3p en EcosimPro y Simulink.

En los aspectos metodológicos estudiados se demuestran algunas carencias del programa Simulink para implementar una metodología de modelado basado en la reusabilidad, isomorfismo, acausalidad matemática y descripción multinivel. Ya que pese a tener la posibilidad de definir subsistemas que pueden trabajar a multinivel, la interfaz de estos subsistemas es causal (entradas y salidas) y poco reusable. Como vimos en el segundo y tercer aspecto de la comparación metodológica es necesario que el modelador implemente la dirección de los flujos mediante bloques SUM manualmente, mientras EcosimPro posee los Puertos, como interfaces de sus elementos, que realizan esos cálculos automáticamente.

También demostramos en el quinto apartado de la comparación que EcosimPro posee más versatilidad en el cálculo, ya que existe una separación entre el modelo y la estructura de cálculo del programa que no existe en Simulink. En EcosimPro, por ejemplo, puedes definir las condiciones de frontera después de crear el modelo cambiando la estructura de cálculo. Ya vimos como la única forma de intervenir en la estructura de cálculo en Simulink era definir bloques no virtuales, aunque esta definición de los subsistemas puede crear lazos algebraicos irresolubles.

La comparación de los resultados de EcosimPro y Simulink con Matlab demuestra que los errores de estos programas con respecto a Matlab se deben sólo a la traslación o partición del modelo, puesto que los errores en ambas plataformas son idénticos entre ellos. Sólo existía diferencia en los errores a partir de la fase de rebote del experimento, pero es debido a que EcosimPro reinicia la integración del sistema a partir del momento en que detecta el evento.

Hemos analizado también el consumo de recursos en diferentes condiciones. Lo hemos hecho gracias a unas herramientas propias de los programas de simulación que retardan el funcionamiento del mismo, pero nos proporcionan valores relativos de tiempo y número de evaluaciones interesantes en diferentes condiciones.

En EcosimPro se demuestra como para sistemas DAE el método DASSL es el más eficaz, pudiendo incluso desestabilizarse el sistema si usas el método de Runge-Kutta con intervalo de comunicación muy alto.

En Simulink hemos visto cómo interfieren los lazos algebraicos en el aumento de números de métodos de subsistemas no virtuales, ralentizando el sistema. Estos lazos algebraicos fueron creados al tratar algunos de los subsistemas como no virtuales de forma que los lazos fueran resolubles.

Para estudiar la influencia de los eventos en el tiempo de simulación y el consumo de recursos creamos un modelo nuevo con un evento y estudiamos sus tiempos sin hacer saltar el evento y haciéndolo saltar varias veces. En Simulink había un retraso entre ambas situaciones del 28.2 % mientras en EcosimPro era mayor, del 36 %. Este peor tiempo es debido a que EcosimPro reinicia el sistema completo al tratarse de un evento que implica cambio de una las ecuaciones.

En la siguiente parte del proyecto, realizado dentro de un equipo de investigación, se desarrolló una librería farmacocinética, LibPK donde hemos buscado maximizar las ventajas de modelado y extender los procesos representados. Por último hemos comprobado su validez mediante pruebas de concepto: Sobre el modelo tricompartimental de hemodiálisis de Urea y sobre un modelo LADME del Diazepam.

6. Conclusiones.

En este proyecto se ha propuesto una comparación entre programas de modelado y simulación. Hemos escogidos dos programas ampliamente extendidos en ingeniería pero con distinta filosofía de modelado: Modelado mediante Diagrama de Bloques y el Modelado Orientado a Objeto, hemos desarrollado el mismo modelo farmacocinético con las mismas condiciones en ambos programas y hemos analizado tanto el desarrollo como los resultados.

Para analizar el desarrollo de los modelos hemos buscado una serie de ventajas de modelado generales para el sistema fisico-químico que nos ocupa, como la reusabilidad, la descripción multinivel y la acausalidad matemática. Comparando entre ambos modelos cual aprovecha mejor esas ventajas.

Estas ventajas de modelado generales son los pilares para la construcción de un "lenguaje céntrico" con código reusable, creciente y con métodos computacionales intercambiables [18] que faciliten la creación, fijación y adhesión de conocimiento y a la interoperabilidad entre investigadores [18]. Ya que la difusión de los conocimiento de modelado fisiológico y farmacocinético es obstaculizada por la falta de enfoques comunes útiles para intercambiar información [19].

En el análisis metodológico ha quedado claro que el programa de modelado Simulink que usa una técnica de modelado mediante Diagrama de Bloques no es lo más apropiado para maximizar las ventajas de modelado que hemos resaltado. Sin embargo EcosimPro si reúne todas las condiciones para poder maximizar estas caracteristicas, aunque en la librería Kinetic no han sido desarrolladas aún a su máximo nivel, si lo son en la librería LibPK.

En el futuro se incorporarán a esta librería aspectos fisiológicos, el sistema cardiovascular y circulatorio. La derivación hacia una librería fisiológica tendrá en cuenta aspectos anatómicos y físico-químicos, el cual tiene en cuenta aspectos energéticos, junto con mecanismos biológicos importantes. También se desarrollará una Librería de instrumentos para dar información farmacocinética de los procesos en curso, tal como AUC[6], tiempo de actividad, concentración media, etc.

[7] Área bajo la curva.

7. Publicaciones.

Artículos.

Tomé Matos, Manuel Prado-Velasco, Cristina Vallez y Juan Navarro:
On a Reusable and Multilevel Methodology for Modeling and Simulation of Pharmacokinetic-Physiological Systems: a preliminary study.
- Artículo en revisión para su publicación en la revista *Computer Methods and Programs in Biomedicine.*

Tomé Matos, Manuel Prado-Velasco, Juan Navarro y Cristina Vallez.
Modeling and Simulation of Multilevel Pharmacokinetic Systems with fully Reusable parts (I): Methodological and Technological design.
- Artículo en fase de preparación.

Manuel Prado-Velasco, Tomé Matos, Juan Navarro y Cristina Vallez.
Modeling and Simulation of Multilevel Pharmacokinetic Systems with fully Reusable parts (II): Computional study and benchmarking.
- Artículo en fase de preparación.

Bibliografia.

[1] Cellier, F.E. Kofman, E. *Continuous System Modelling.* Springer. 1991.

[2] Hunter, P.J. *Modeling living systems: the IUPS/EMBS Physiome project.* Proceedings IEEE, 94, 678-991. 2006.

[3] Aderem, A. *Systems biology: Its practice and challenges.* Cell, vol. 121(4), pag 511-513. 2005.

[4] The Mathworks Inc. *Simulink User´s guide. R 2012b.* Marzo 2012.

[5] Empresarios Agrupados S.A. *Manuales EcosimPro.*

[6] Francisco Vázquez, Jorge Jiménez, Juan Garrido y Antonio Belmonte. *Introducción al modelado y simulación con EcosimPro.* Pearson, 2010.

[7] Manuel Prado Velasco. *Tesis Doctoral: Aportaciones a la tele-asistencia de pacientes renales crónicos.* Escuela Superior de Ingenieros de Sevilla, 2003.

[8] Canaud et al. *Urea as a marker of adequacy in hemodialysis: Lesson from in vivo urea dynamics monitoring.* Kidney International, Vol. 58, 2000.

[9] Prado M, Roa L, Palma A, Milán JA. *A novel mathematical method based on urea kinetic modeling for computing the dialysis dose*. Computer Methods and Programs in Biomedicine, 74(2), 2004.

[10] Fernandez de Canete, J. Del Saz Huang, P. *First-principles modeling of fluid and solute exchange in the human during normal and hemodialysis conditions.*
Computers in biology and medicine, Vol. 40, 2010.

[11] Prado, Manuel. Roa, Laura. Palma, Alfonso y Milan, José A. Space. *Discretization in Multipool Kinetic Models Used for Telehealthcare*
4th International workshop in Biosignals Interpretation. 363-366. 2002.

[12] Prado M, Roa LM (2005). Virtual modeling of compartimental pharmacokinetic systems. 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society; Sept.1-4; Shanghai, China.

[13] Cellier, F.E. Kofman, E. *Continuous System Simulation.* Springer. 2006.

[14] Meyer, B. *Reusability: The Case for Object-Oriented Design.*
IEEE Software. Vol. 4. pag. 50-64. 1987.

[15] Roa LM, Prado M. *Simulation lenguajes.*
Wiley Encyclopedia of Biomedical Engineering. nº 8, pag. 1-13. 2006

[16] Zoltan Szallasi, Jörg Stelling y Vipul Periwal. *System Modelling in Cellular Biology: from concepts to nuts and bolts.*
Massachusetts Institute of Technology, 2006.

[17] Ingolf Meineke *, Jürgen Brockmöller. *Simulation fo complex pharmacokinetic models in Microsoft Excel.*
Computer Methods and Programs in Biomedicine, Vol. 88, 2007.

[18] Daniel Stoffler, Sophie I Coon, Ruth Huey, Arthur J Olson, Michel F Sanner, North Torrey, Pines Road and La Jolla. *Integrating Biomolecular Analysis and Visual Programming: Flexibility and Interactivity in the Design of Bioinformatics Tools.* 2002.

[19] Karl Thomaseth. *Multidisciplinary modelling of biomedical systems.* Computer Methods and Programs in Biomedicine. 71, pag. 189-201. 2003.