2. Introducción a *Cryojab*[2]

Cryojab es el nombre que recibe la aplicación desarrollada por *Jorge Aguilar Barrera* en 2002. Dicha aplicación estaba diseñada para realizar una serie de simulaciones que muestran la evolución del campo de temperatura en el músculo cardiaco humano, al que se le habría perfusionado una solución crioprotectora.

2.1. Fundamentos de *Cryojab*

Cryojab realiza un estudio de la evolución del campo de temperatura en órganos a partir de una fotografía digitalizada de una parte del órgano que sea representativa del mismo, pudiéndose aproximar el comportamiento del mismo a partir de lo que se estudie en esa porción seleccionada.

Para ello ***Cryojab*** hace uso de una serie de conceptos y aproximaciones:

- Conducción Vs. Convección: cuando un órgano es enfriado haciendo uso de su sistema vascular, la convección es el método de transferencia de calor predominante en los grandes vasos sanguíneos. Sin embargo, nuestro caso de estudio es el de los capilares. En estos, el agente crioprotector fluye lentamente, por lo que podemos aproximar por conducción.
- Geometría 2D: trabajaremos solo en el plano XY, suponiendo que los resultados son similares en el eje Z.

[2] Ref.: [1], [2], [5], [6], [7], [8], [19], [26], [27]

- Se aproxima por los materiales más representativos. En el caso del miocardio, los cardiomiocitos, el tejido conjuntivo y los capilares. El resto de células diferentes se aproximan a uno de estos materiales.
- Las propiedades físicas de los tejidos (densidad, conductividad térmica y capacidad calorífica) no varían con el tiempo.
- Durante el proceso no hay transferencia de masa. Asumimos que no existe formación de hielo, que es uno de los mecanismos de transferencia de masa más importantes. El segundo mecanismo es la difusión de un agente crioprotector. De todas formas, por simplicidad, asumiremos que usaremos un agente que no varíe la distribución de masas en el órgano perfusionado.

2.1.1. Modelización

Cryojab hace uso de un concepto denominado **subestructuración** para poder resolver el problema de la transferencia de calor. La imagen que sirve como entrada a Cryojab es modelada en forma de matriz (un elemento por cada pixel de la misma). Esta matriz puede alcanzar unas dimensiones demasiado elevadas, llegando a requerir un tiempo de computación excesivamente extenso.

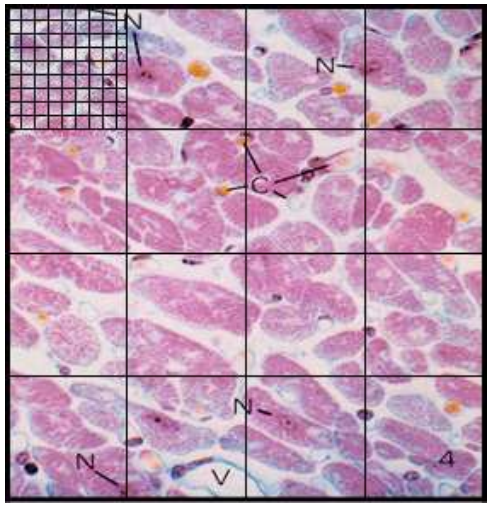


Fig. 2.1 - Imagen Histológica subestructurada

La **subestructuración** consiste en subdividir la imagen en secciones más pequeñas (subestructuras), y por tanto la matriz que representa a dicha imagen, de manera que cada subestructura (cada matriz) es resuelta de manera independiente. Posteriormente, se establecen las condiciones que relacionan las subestructuras entre si para ofrecer un resultado global. La ventaja de usar la subestructuracion reside en que, al disminuir el orden de la matriz con la que se trabaja (cada subestructura), el tiempo invertido en computar todas las matrices correspondientes a cada subestructura es mucho menor al requerido para resolver la matriz que representaría a la imagen completa.

2.1.2. Fundamentos Físicos y Matemáticos

Los órganos están compuestos por distintas sustancias, que confieren al sistema completo de heterogeneidad y de inercia térmica. Esta inercia, unido al hecho de que el órgano se enfría de manera progresiva, da lugar a un régimen, en esencia, no estacionario, por lo que se requiere un tratamiento que aporte versatilidad, rapidez y no implique pérdida de precisión, fundamental para una simulación efectiva.

Así pues, el método elegido no es otro que el método de las diferencias finitas, volúmenes finitos o elementos finitos, en el cual la precisión obtenida en los resultados depende del grado de discretización que se adopte. Una explicación breve del método de las diferencias finitas: el método de las diferencias finitas es un método de carácter general que permite la resolución aproximada de ecuaciones diferenciales en derivadas parciales definidas en recintos finitos.

Para la resolución del problema de transferencia de calor se adopta el método de los autovalores. Este método resuelve el problema de conducción discretizando en el espacio el dominio de cálculo (unidimensional o multidimensional) en diferencias, volúmenes o elementos finitos, manteniendo la variación del campo de temperaturas con el tiempo en forma diferencial.

Este método presenta como principal ventaja el tratamiento de condiciones de contorno no lineales y variables con el tiempo, así como la posibilidad de resolver problemas con excitaciones diversas y la posibilidad de considerar la dependencia de las propiedades termofísicas con la temperatura. Además es capaz de tratar problemas con paso de tiempo variable, de manera que se ajusten a la realidad de los fenómenos físicos que toman parte en el proceso de enfriamiento, con las consecuentes ventajas de ahorro de tiempo de cálculo y precisión en los resultados.

No entraremos a profundizar en el uso de estos métodos pues no es el objeto de este proyecto, aunque en la documentación del proyecto en el que éste se basa se encuentra un desarrollo mucho más exhaustivo.

2.2. Implementación de *Cryojab*

La primera versión de ***Cryojab*** consta de dos aplicaciones implementadas en el lenguaje de programación ***C++***, haciendo uso de las librerías ***Fortran*** para el cálculo matemático. La primera de ellas es un paso previo que se realiza antes de comenzar la simulación para el tratamiento del parámetro principal de entrada de la aplicación central.

2.2.1. *Cryojab-color*

Cryojab-color tiene como principal y única función adecuar al formato esperado por ***Cryojab*** la imagen histológica que nos sirve como entrada del sistema. ***Cryojab-color*** realiza una identificación de los nodos de las subestructuras con los diferentes materiales que aparecen en la fotografía. ***Cryojab-color*** realiza la conversión de una imagen recogida en el laboratorio de histología a un archivo con código que el programa principal necesita para realizar la simulación.

Pensemos en una imagen de 50 x 50 pixeles. Este tamaño es irrisorio si comparamos con las fotografías que podemos tomar con una cámara digital convencional hoy en día. Para hacernos una idea del tamaño, 50 x 50 pixeles es medio icono de escritorio. Si esta pequeña fotografía la subdividimos en secciones cuadradas con un lado de 10 pixeles, tendremos 5 x 5 subestructuras de 10 x 10 pixeles cada una. Como se verá más adelante, ***Cryojab*** requiere un fichero por cada subestructura que contenga el tipo de material que se encuentra en cada nodo. Es decir, que tendríamos que generar 25 ficheros de 100 líneas de código manualmente (independientemente del resto de parámetros que necesita ***Cryojab***). Esto convertiría a la aplicación original en inviable. He aquí el motivo de la implementación de la pequeña aplicación denominada ***Cryojab-color***.

Para realizar todo este proceso, ***Cryojab-color*** requiere una serie de parámetros de entrada. En primer lugar necesita la imagen que queremos estudiar, pero reconvertida a formato textual indicando las componentes **RGB** de cada pixel. Esta conversión no la realiza ***Cryojab-color***, así que es necesario que el usuario haga uso de algún software capaz de realizar este paso. La aplicación que se ha estado usando hasta ahora ha sido ***GIMP*** (*GNU Image Manipulation Program*).

GIMP es un programa de edición de imágenes, y es la alternativa más firme del software libre a ***Photoshop***. ***Cryojab-color*** fue diseñado para que esta conversión la realizase la aplicación ***GIMP***, convirtiéndose en imperativo el uso de esta aplicación, pues ***Cryojab-color*** espera la imagen en un formato muy concreto (el que genera ***GIMP***).

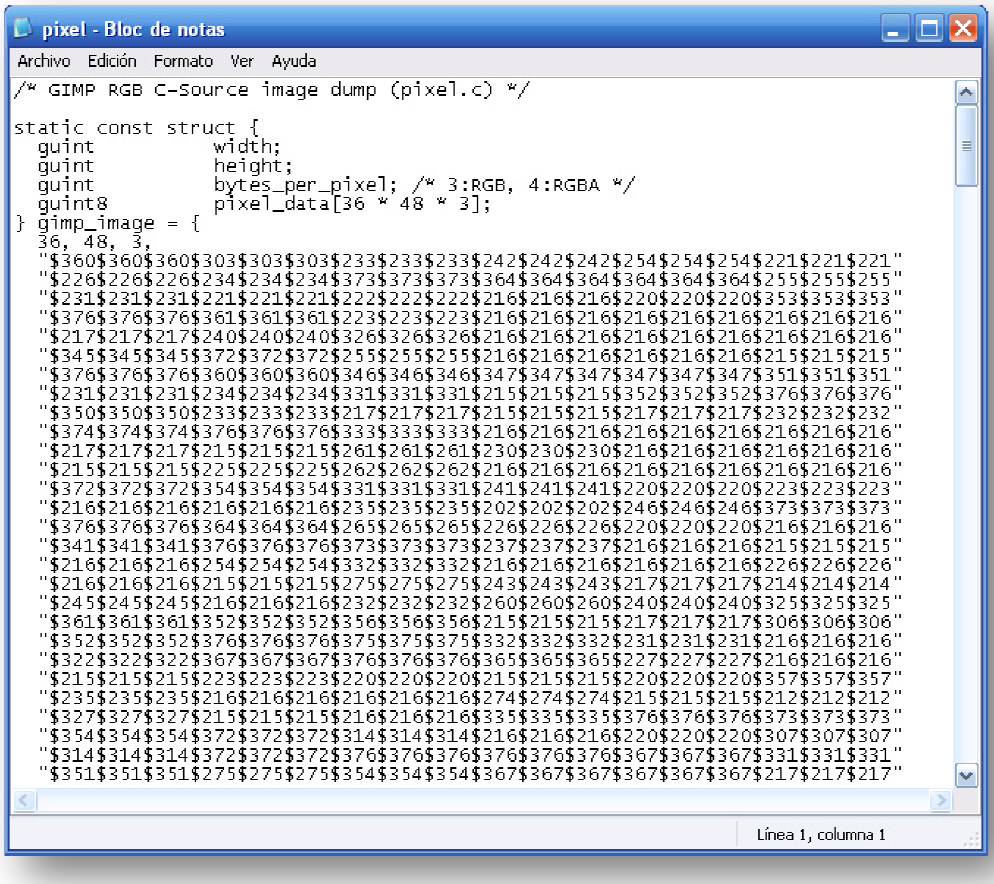


Fig. 2.2 – Fichero de entrada *Pixel.txt* para *Cryojab-color* generado por *GIMP*

Además de este fichero con la imagen en formato textual, ***Cryojab-color*** precisa de un segundo fichero con los diversos materiales que aparecen en la imagen que

queremos estudiar, el número de subestructuras total en la que vamos a subdividir la imagen y el número de nodos en el que vamos a subdividir la subestructura.

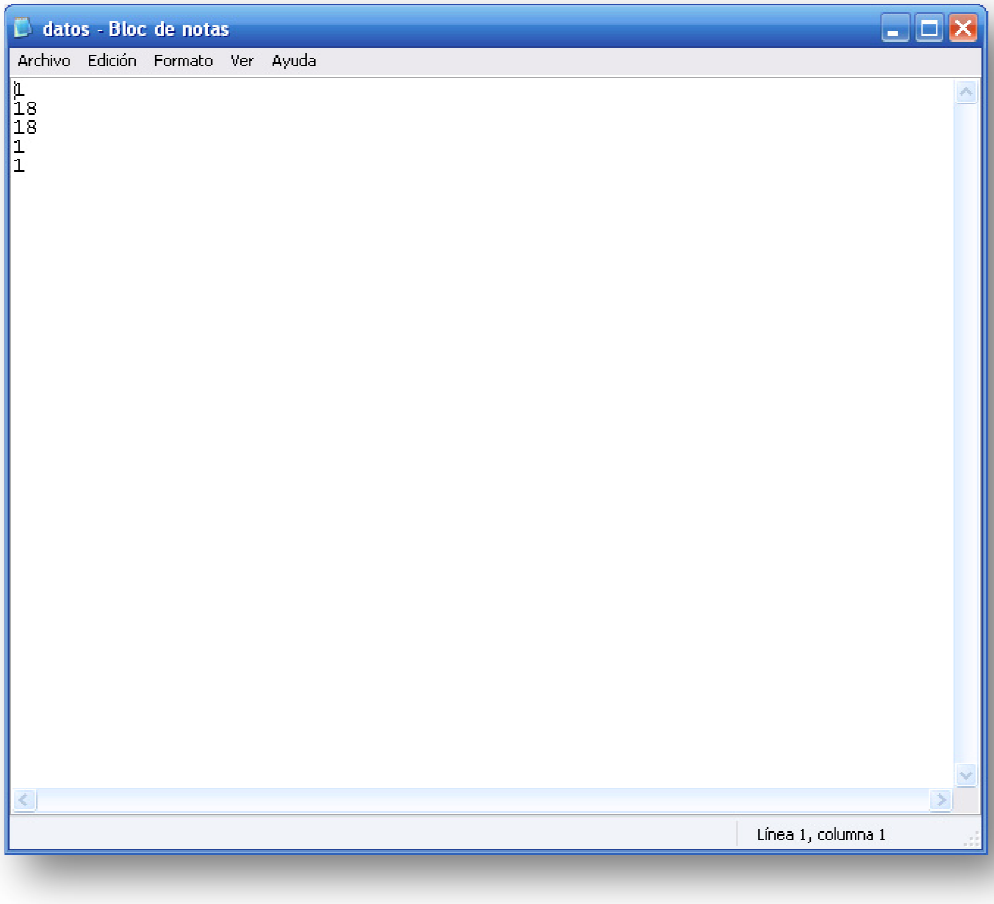


Fig. 2.3 – Fichero *Datos.txt* de entrada para ***Cryojab-color***

El procedimiento que sigue ***Cryojab-color*** es sencillo. Extraemos del fichero el color en sus componentes **RGB** correspondientes a un determinado pixel, comparamos con los colores indicados como materiales y se asigna a dicho nodo el material concreto. Así se prosigue hasta finalizar con el fichero que contiene la imagen en formato textual.

El fallo radica en que el programa tiene que realizar una comparación de cualquier color con unos concretos que nosotros hayamos introducido en el fichero de

datos. A la hora de comparar, puede encontrarse colores que no coinciden para nada con el extraído, por lo que el programa retornaría una salida errónea y aleatoria. No controlaríamos adecuadamente el funcionamiento del programa. Sin embargo, la idea de base es la adecuada, así que para la nueva implementación se aplicara un algoritmo similar.

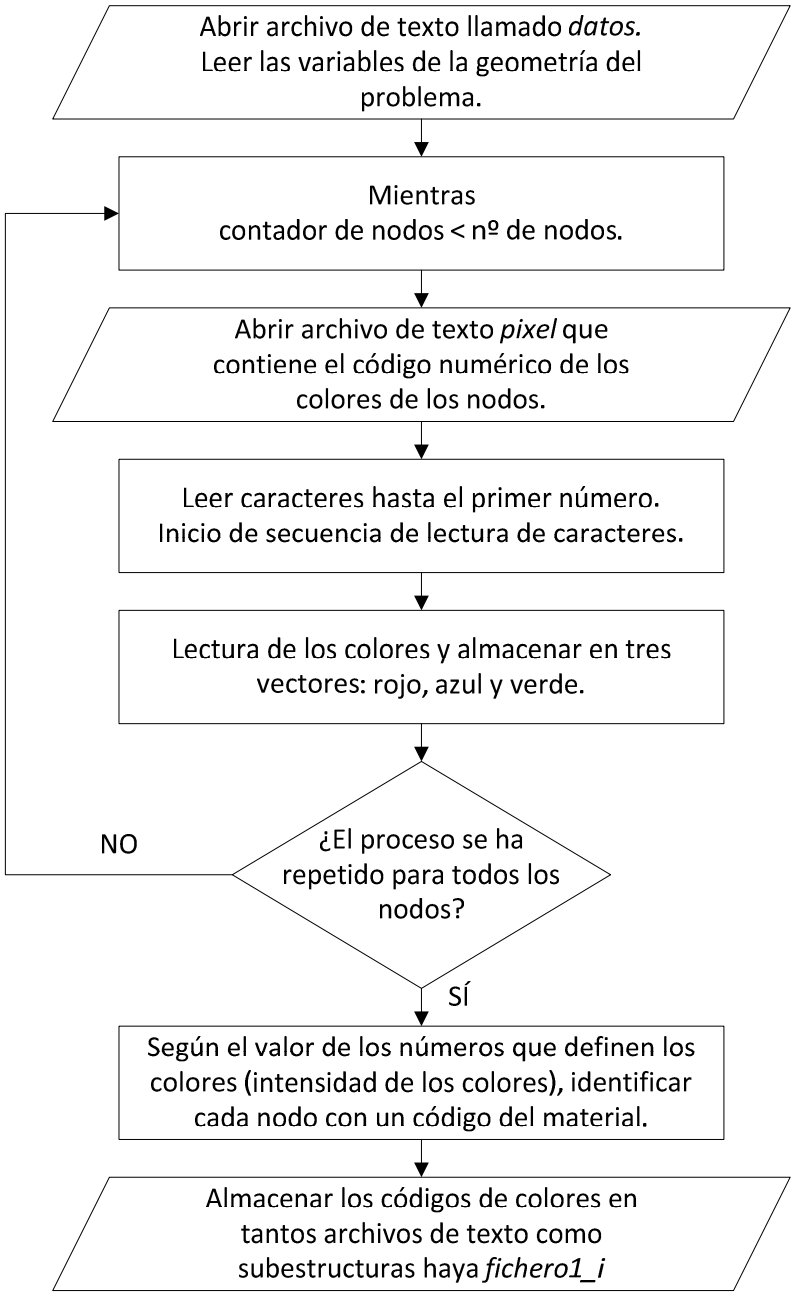
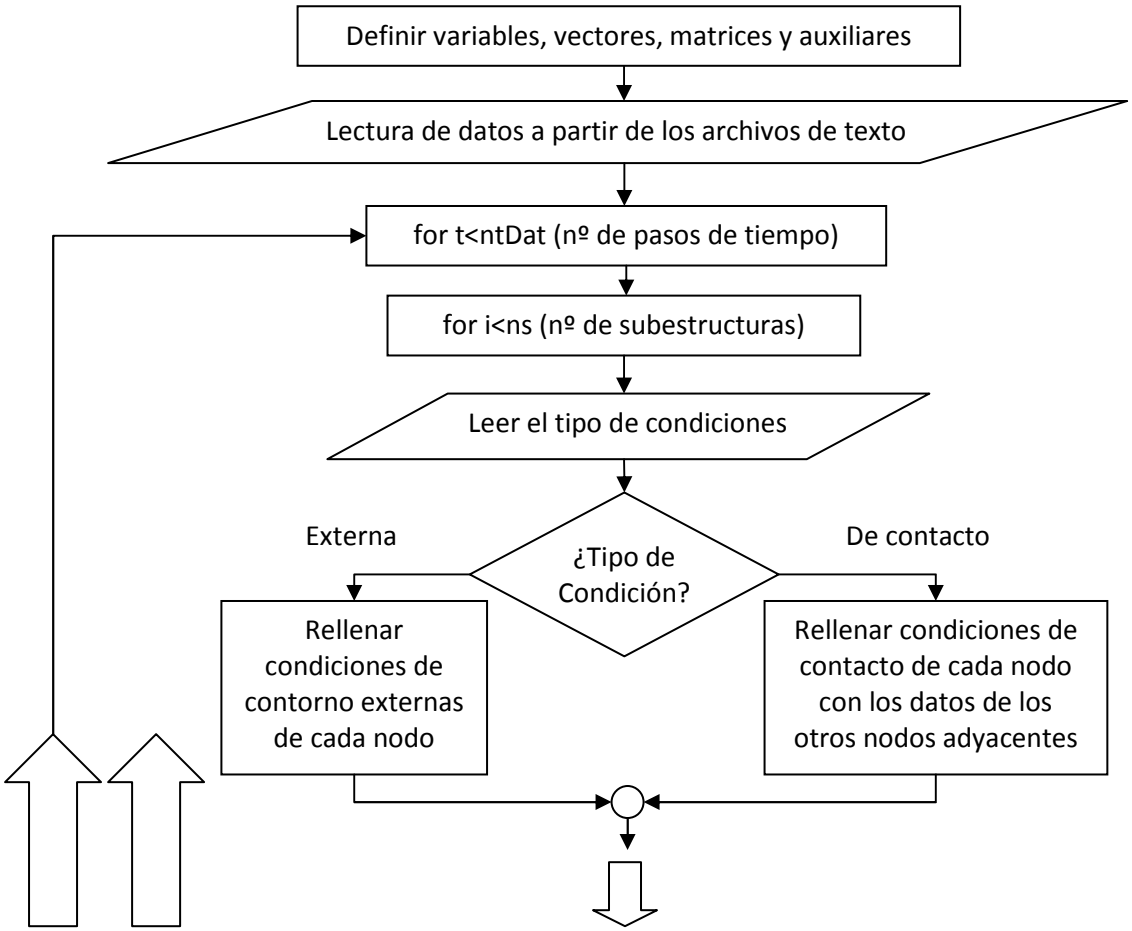


Fig. 2.4 – Diagrama de flujo de la aplicación ***Cryojab-color***

2.2.2. *Cryojab*

Por otro lado tenemos lo que sería el núcleo central de la aplicación, ***Cryojab***. Es el programa que realiza la simulación, tomando como entrada una serie de ficheros que el usuario deberá haber generado antes de ejecutar la aplicación.

El núcleo principal de ***Cryojab*** está compuesto por dos módulos. En primer lugar tenemos el módulo ***subestructuras***, donde se implementan las funciones de tratamiento de las subestructuras. Y por otro lado tenemos el modulo ***main***, donde se desarrolla la ejecución principal de la aplicación, realizando las llamadas a las funciones contenidas en el módulo de ***subestructuras*** tantas veces como sea necesario en función del número de subestructuras contenidas en la imagen. A continuación vemos el diagrama de flujo.



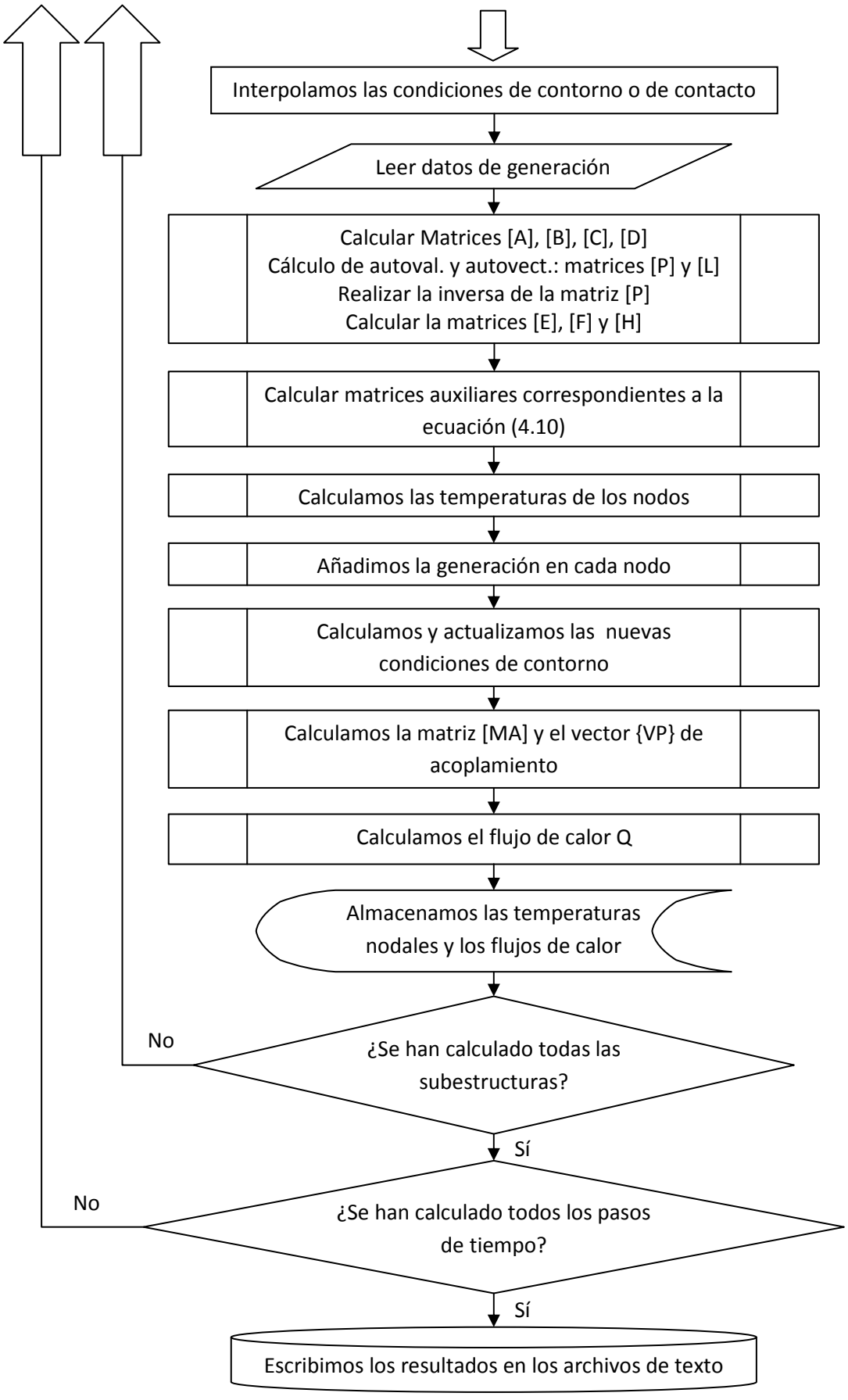


Fig. 2.5 – Diagrama de flujo de la aplicación *Cryojab*

Cryojab requiere de 6 tipos de ficheros para realizar la simulación:

- Fichero tipo 0: fichero que contiene el número de nodos en horizontal y vertical de cada subestructura. Por tanto, se requiere un fichero tipo 0 por cada subestructura. También se incluye el tamaño real que posee la subestructura (es decir el tamaño en metros de esa pequeña sección de órgano) y la temperatura inicial de la subestructura a la que corresponde el fichero.

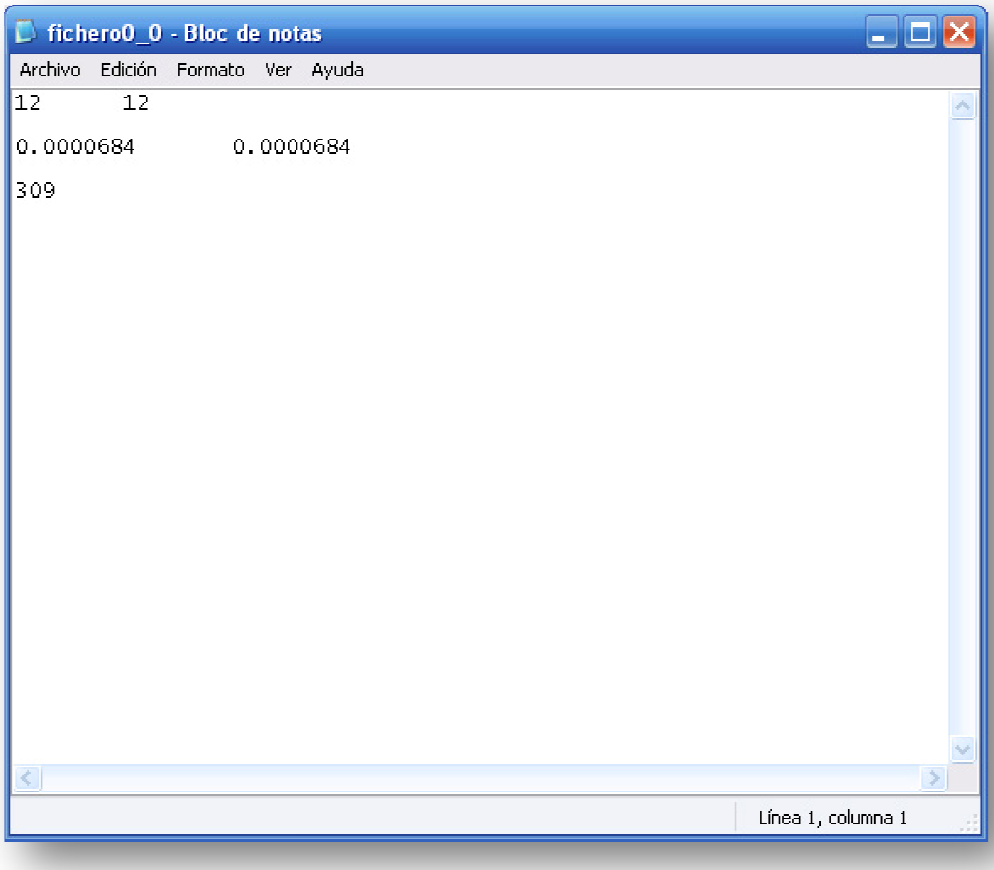


Fig. 2.6 – Entrada de *Cryojab*. Fichero de tipo 0

- Fichero tipo 1: procede de ***Cryojab-color***. Contiene el tipo de material que hay en cada nodo de cada subestructura. Existirá un fichero tipo 1 por cada subestructura existente. Podemos apreciar la magnitud de estos ficheros. Con subestructuras de 10x10 nodos tendremos 100 líneas en el fichero de tipo 1. Y esto por cada subestructura. Las dos primeras columnas indican la coordenada del nodo dentro de la subestructura, mientras que la tercera indica el número asignado al material contenido en ese nodo.

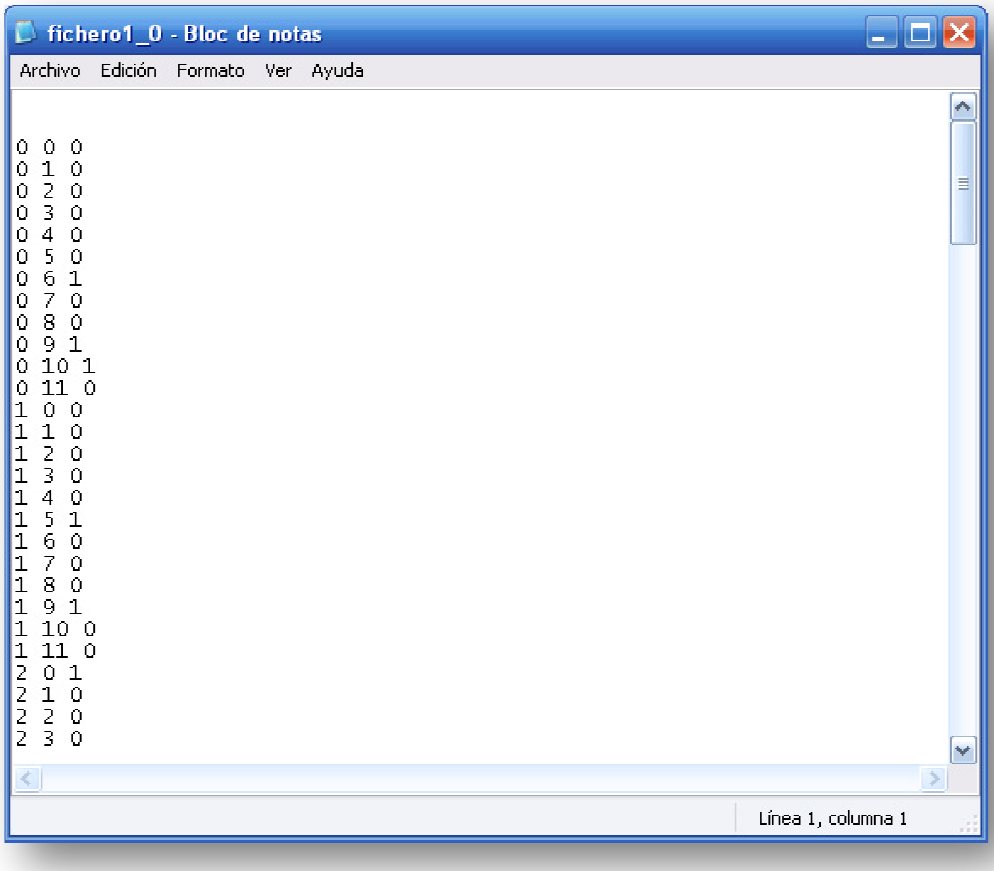


Fig. 2.7 - Entrada de *Cryojab*. Fichero de tipo 1. Generado por *Cryojab-color*

- Fichero tipo 2: aquí se recogen los distintos materiales que aparecen en la imagen a estudiar. En primer lugar se indica el número de materiales que vamos a considerar a la hora de realizar la simulación. Posteriormente, para cada material se indican una serie de parámetros físicos. La primera columna se corresponde con la conductividad térmica, la segunda columna se corresponde con la densidad, la tercera columna se corresponde con la capacidad calorífica y la última columna se corresponde con la velocidad. Este último campo valdrá 1 o 0 en función de si en el material es o no es capilar.

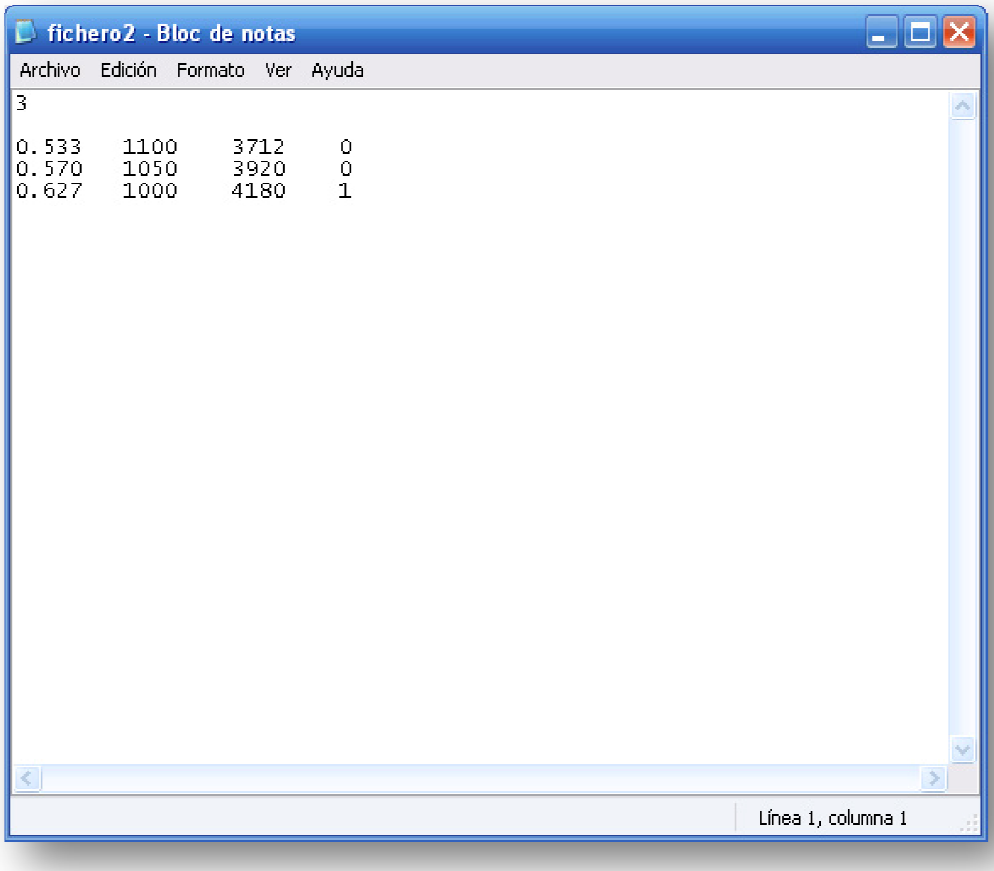


Fig. 2.8 - Entrada de *Cryojab*. Fichero de tipo 2

- Fichero tipo 3: indica la posición de la subestructura en el sistema total. ***Cryojab*** ordena las subestructuras dentro del sistema mediante 4 coordenadas: abajo, arriba, derecha e izquierda. En función del signo del valor de la coordenada se indica un dato u otro. Si la coordenada es positiva indica el número de la subestructura con la que limita por ese lado. Si la coordenada posee signo negativo, indica que ese lado de la subestructura coincide con uno de los lados de la imagen, y, por tanto, se corresponde con una condición de contorno (la correspondiente a ese lado de la imagen).

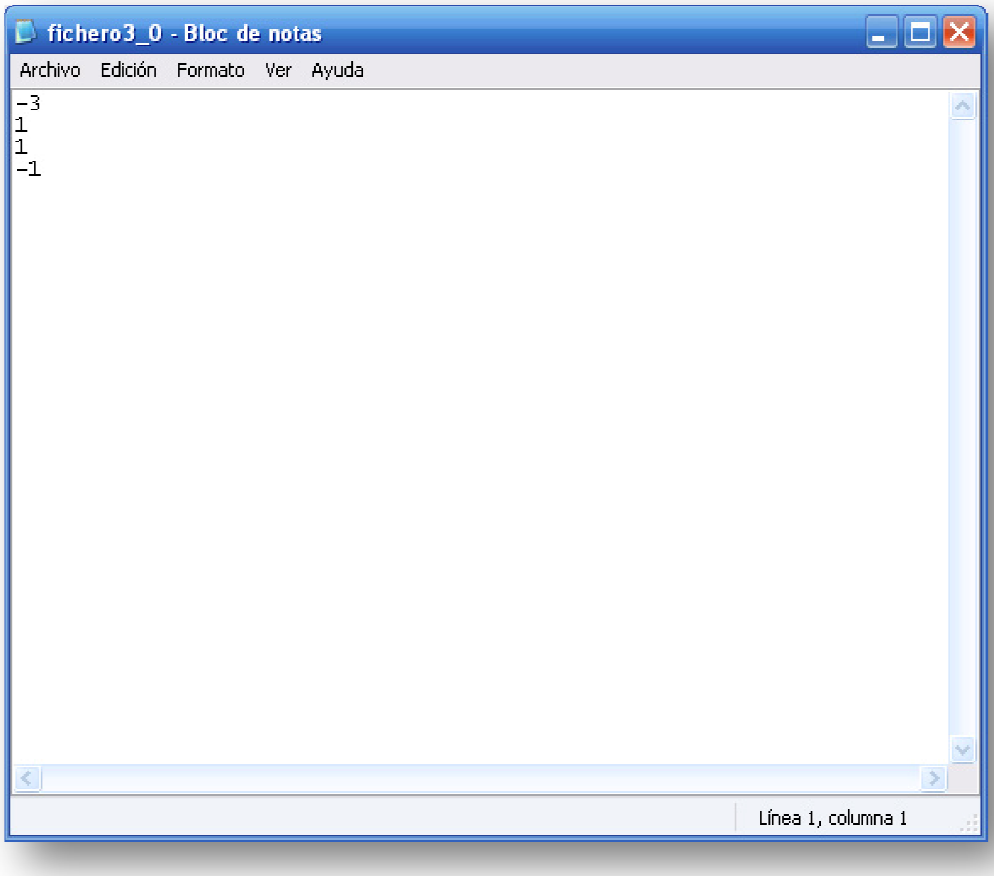


Fig. 2.9 - Entrada de *Cryojab*. Fichero de tipo 3

- Fichero tipo 4: indica el número de condiciones de contorno diferentes que asignaremos a los bordes de la fotografía. Se expresan mediante 3 coeficientes: ***a***, ***b*** y ***r***. ***a*** y ***b*** indican los coeficientes de la función lineal que utilizaremos para establecer un valor para la temperatura (puede ser fija o variable). La función es . Para temperaturas constantes ***b*** = 0. Para el caso ***adiabático***, es decir, que suponemos que en ese lado de la fotografía existirá otra simétrica, ***r*** = 1000000000000 (aproximadamente infinito), y en el caso de temperatura impuesta (si la fijamos mediante ***a*** y ***b***) ***r*** = 0.

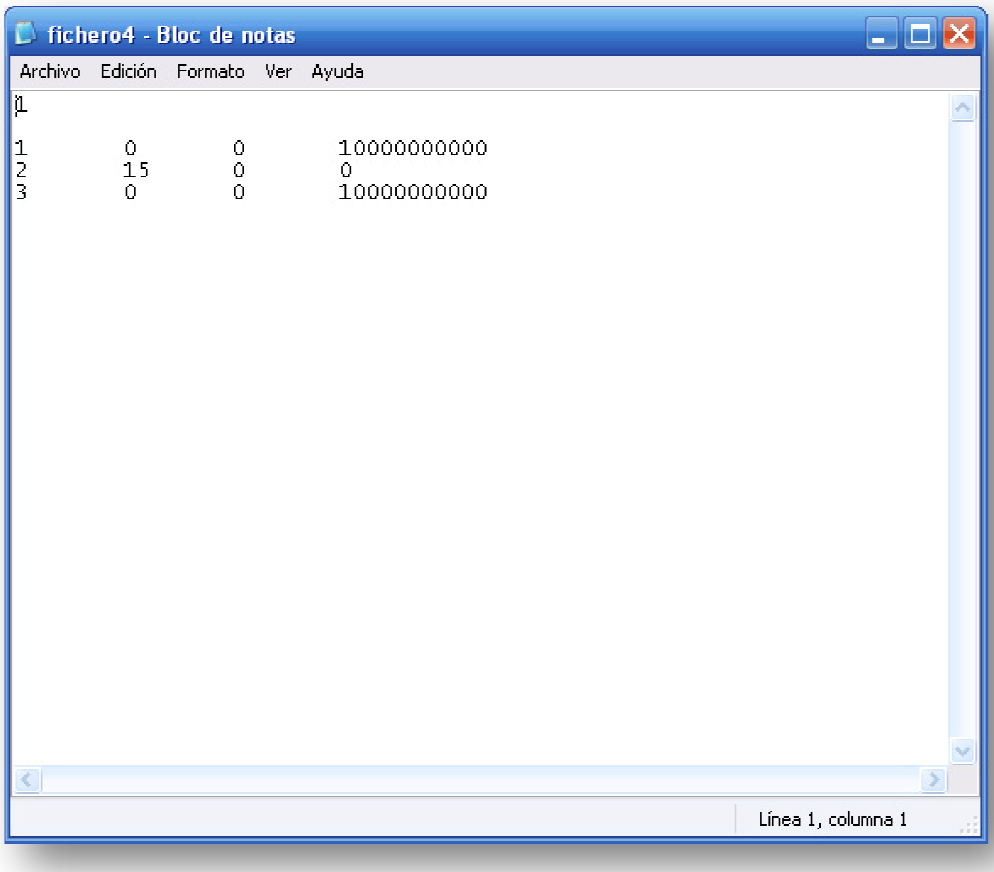


Fig. 2.10 - Entrada de *Cryojab*. Fichero de tipo 4

- Fichero tipo 5: es el fichero que proporciona los parámetros de generación, es decir, los parámetros relacionados con la solución crioprotectora que perfusionamos en el órgano. Se basa en la siguiente función lineal: . ***M*** es la temperatura inicial del fluido crioprotector y ***N*** la velocidad a la que enfría/calienta la solución. El signo de ***N*** es importante. Signo positivo indica que calienta y signo negativo indica que enfría. Es necesaria una línea por cada subestructura que hayamos definido.

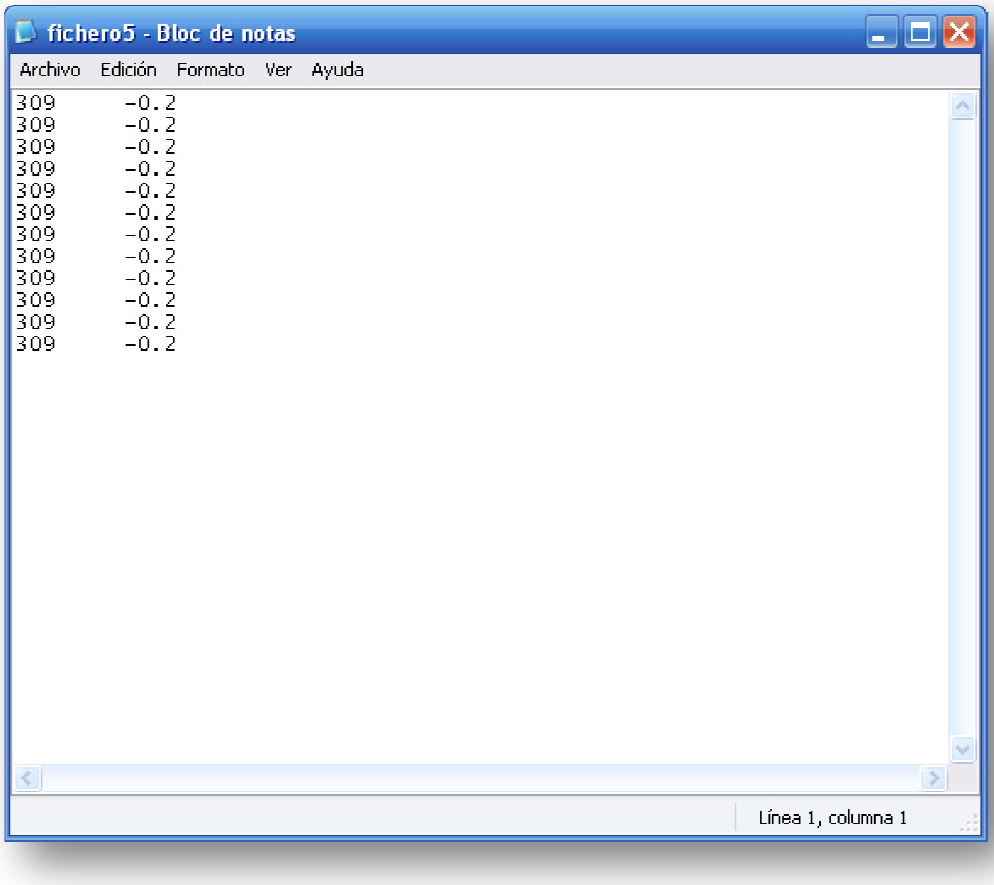


Fig. 2.11 - Entrada de *Cryojab*. Fichero de tipo 5

- Fichero tipo 6: incluye los parámetros del sistema completo. Se indica el número total de subestructuras, el tamaño del paso para la simulación (el tiempo de simulación en segundos), así como el número de subestructuras en horizontal y el número de subestructuras en vertical.

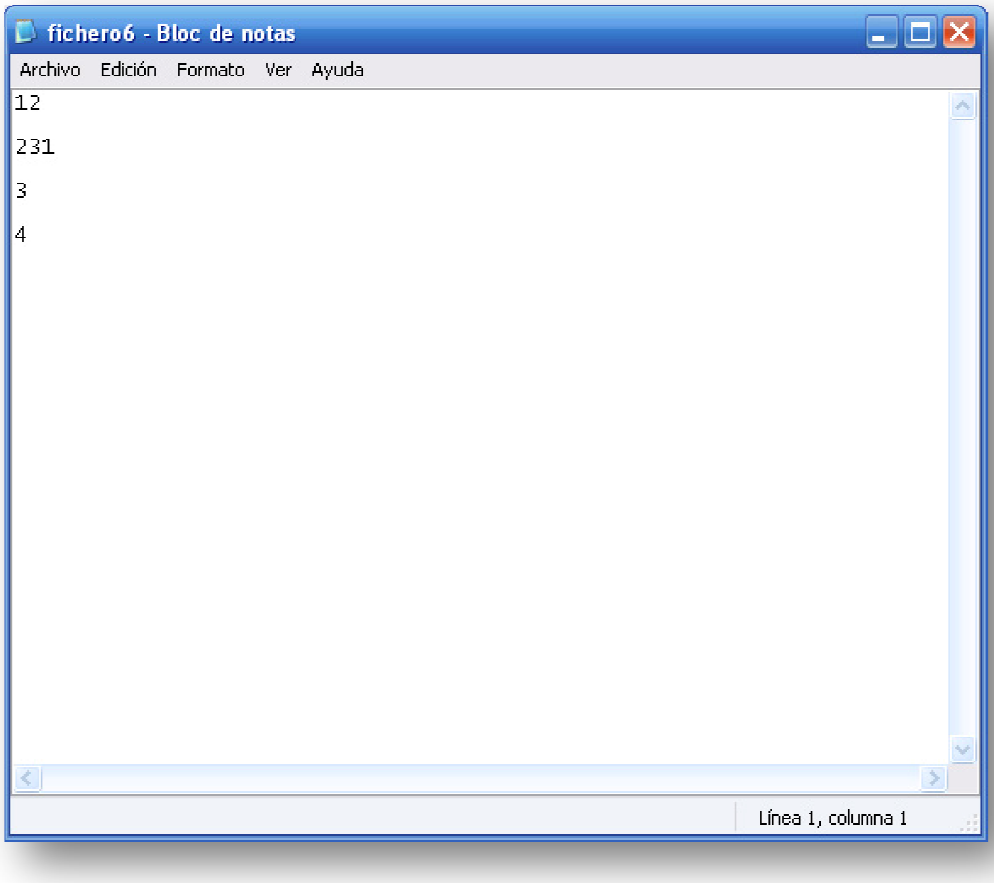


Fig. 2.12 - Entrada de *Cryojab*. Fichero de tipo 6

En este apartado no se incluirá ningún tipo de código, ya que el nuevo proyecto toma los ficheros de código fuente de la aplicación original para desarrollar un nuevo entorno, aunque el núcleo del sistema queda casi intacto.

2.3. Problemas e inconvenientes de la primera implementación de *Cryojab*

2.3.1. Creación manual de los ficheros de entrada.

Anteriormente se puso un pequeño ejemplo de la cantidad de ficheros que habría que generar para realizar una simulación concreta. Éste es sin duda el gran inconveniente que nos encontramos al trabajar con ***Cryojab***.

Cada simulación que queramos realizar implica la creación manual de una gran cantidad de ficheros. Ficheros que además guardan una relación muy fuerte entre sí, complicando aun más la generación de los mismos. Basta con imaginar los ficheros de tipo 3 (los que indican las subestructuras limítrofes).

Supongamos que queremos realizar una simulación con una imagen de 100 x 100 pixeles con subestructuras de 10 x 10 pixeles. Esto nos resultarían 10 x 10 subestructuras, es decir, 100 subestructuras. 100 ficheros tipo 3 que tendríamos que generar sin cometer ningún error. 100 ficheros tipo 0, indicando las propiedades de cada subestructura.

En total, para una simulación de este tipo sería necesario generar unos 300 ficheros, 200 de los cuales habría que crearlos manualmente (los de tipo 1 los genera ***Cryojab-color***).

Así pues el gran escollo que hay que salvar es la creación de los ficheros con los parámetros de entrada.

2.3.2. Funcionamiento incorrecto de *Cryojab-color*

Tras ejecutar la aplicación, se ha comprobado que ***Cryojab-color*** no está correctamente implementado. Es decir, no genera adecuadamente los ficheros de tipo 1.

Esto se convierte en el segundo problema a afrontar. Y de mayor importancia, si cabe, que el anterior. Pues los ficheros de tipo 1 son los más complejos de generar. Si tuviésemos que crearlos manualmente, directamente se desecharía la posibilidad de realizar simulaciones con esta aplicación, pues generar los ficheros de tipo 1 de una imagen con un tamaño no excesivamente grande, 100 x 100 pixeles por ejemplo, implicaría generar unas 10000 líneas de texto identificando claramente cada pixel con cada material.

Es necesario solventar los problemas de implementación de ***Cryojab-color***, pues es uno de los pilares básicos de la aplicación.

2.3.3. Errores en liberación de memoria en *Cryojab*

Aquí nos encontramos con un error de menor magnitud. ***Cryojab*** genera un error al finalizar la simulación. ***Cryojab*** genera correctamente todos los ficheros de resultados, por lo que el funcionamiento es el correcto, y el problema se detecta claramente en la zona de liberación de memoria.

Para pulir el funcionamiento de la aplicación, será necesario localizar el punto concreto en el que se libera memoria y asegurarse de que se realiza esta operación de forma correcta y ordenada. Para ello habrá que tener en cuenta todas las posibilidades

que pueden surgir a raíz del desarrollo de la nueva aplicación, inviables hasta el momento.

2.3.4. Entrada/Salida en formato textual

Finalmente tenemos que considerar la posibilidad de introducir los datos y obtener los resultados de manera no textual.

Es muy poco cómoda e intuitiva la forma en la que ***Cryojab*** recibe los parámetros de entrada para la simulación. Lo ideal sería poder introducir los datos de una manera ordenada y por pasos, guiando al usuario de manera que no se pierda en el proceso de preparar la simulación.

Tras ésta, ***Cryojab*** ofrece los resultados en forma de matrices almacenadas en ficheros textuales. Si bien es necesario tener estos datos almacenados para realizar comparativas exhaustivas, es mucho más representativo un resultado grafico.

Por ello se hace necesaria la creación de una interfaz para el usuario, que nos permita realizar simulaciones de manera rápida, efectiva e intuitiva.

2.3.5. Uso excesivo de la memoria virtual

Cryojab, a la hora de realizar una simulación, almacena todos los datos resultantes en memoria, y al final del proceso los vuelca a los ficheros de salida. El objetivo de esta primera implementación era muy concreto: estudiar el comportamiento del musculo cardiaco con una imagen muy concreta. No se tuvo en cuenta los posibles

usos de la aplicación y las posibles nuevas entradas que se podrían introducir en el programa.

Así, cuando el número de subestructuras es elevado y el tiempo de simulación es un tiempo lo suficientemente interesante para realizar una simulación, la cantidad de memoria que ***Cryojab*** reserva para su uso es excesiva.

En alguna de las pruebas realizadas, ***Cryojab*** ha llegado a ocupar aproximadamente dos gigabytes de memoria (entre memoria virtual y memoria física). Esto puede llegar a ser un problema crítico, llegando a provocar errores de funcionamiento de la aplicación si ésta no encuentra suficiente memoria para almacenar los datos resultantes. Especialmente en computadoras que tengan limitada la memoria virtual o, sin llegar a tenerla limitada, poseen poco espacio en el disco duro donde se encuentra el fichero de paginación del sistema.

Como pequeña explicación, podríamos decir que la memoria virtual es un concepto que permite a una aplicación utilizar más memoria principal (**RAM**, acrónimo de *Random Access Memory*) de la que realmente posee el ordenador. Para ello se hace uso del disco duro, que, como bien sabemos, posee unos tiempos de acceso a lectura y escritura mucho mayores que la memoria **RAM**, pero que para almacenar datos temporales que no van a ser tratados en demasía por la aplicación (como es nuestro caso), es muy útil. Así, si esta memoria virtual está limitada por algún tipo de margen o el disco duro que está configurado para ello no posee demasiado espacio libre, puede llegar a desbordar, provocando fallos de funcionamiento.

Este problema es de difícil solución, ya que nuestro objetivo es no modificar la aplicación original. Y el problema se encuentra precisamente en cómo está ideada la

misma. El proceso que sigue ***Cryojab*** es sencillo (como se ha visto en los diagramas de flujo anteriores). Extrae lo datos de entrada, los procesa, y los vuelca en los ficheros de salida. Entonces, la única solución viable, sin modificar completamente la aplicación, es asegurarse de que el desbordamiento de la memoria virtual no va a suceder. Para ello hay que modificar la configuración de la memoria virtual, permitir al sistema que sea él quien gestiones el fichero de paginación y asegurarse de que siempre hay suficiente espacio en el disco duro configurado para ser usado también como memoria virtual.

2.3.6. Dependencia de la arquitectura

Este es el problema de menor envergadura de todos los encontrados. ***Cryojab*** está implementado en un lenguaje compilado, es decir, requiere que un compilador genere una aplicación que se ajuste a una determinada arquitectura. Esto priva de portabilidad a la aplicación.

Si nosotros quisiésemos migrar la aplicación a una arquitectura distinta de la que estemos usando (por ejemplo queramos pasar a ***Linux***, habiendo estado trabajando en ***Windows***), se requeriría una nueva compilación del código para que se adapte a las exigencias del nuevo entorno.

Esto no supondría ningún problema si no fuese porque la nueva implementación que se lleva a cabo en este proyecto añade la posibilidad de la portabilidad, ya que el lenguaje que se ha usado es interpretado.