Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Trabajo Fin de Máster

Modelado del metabolismo de células tumorales en la formación y desarrollo de esferoides tumorales

Modelling of tumoral cells metabolism in tumoral spheroids formation and development.

Autor:

Alejandro Modrego Bravo

Directores:

José Manuel García Aznar

Inês G. Goncalves

Escuela de Ingeniería y Arquitectura

Curso 2022/2023

Índice

1. Introducción

El cáncer es una de las principales causas de muerte en el mundo, estimándose que un quinto de la población mundial sufrirá de él en algún momento de su vida [1]. Aunque cada tipo de cáncer tiene sus particularidades, todos ellos comparten ciertas características comunes entre las que se incluyen un crecimiento sostenido y no controlado de la población de las células cancerígenas, una resistencia anormal a la necrosis, una alta capacidad de metástasis y la capacidad de inducir angiogénesis, proceso por el cual las células activan mecanismos fisiológicos para la creación de nuevos vasos sanguíneos a partir de los ya existentes en el entorno, con los que son capaces de absorber una mayor cantidad de nutrientes.

A pesar de que las causas por las que las células del organismo se transforman en células tumorales aún no se conocen, está aceptado que estas se originan debido a que las células sufren una serie de mutaciones que provocan que varíen sus mecanismos de proliferación y necrosis y que, a medida que el tumor se desarrolla, causan alteraciones malignas como las citadas anteriormente [2].

A lo largo de su desarrollo, el tumor sigue 3 fases diferenciadas. En la primera fase se da un crecimiento exponencial avascular, donde las células obtienen los nutrientes y el oxígeno necesario mediante de la difusión de estos a través de los tejidos próximos. En este entorno, el tumor prolifera hasta un límite en el que entra en la segunda fase donde, debido a que no es capaz de adquirir nutrientes suficientes por medio de la difusión, se satura y mantiene su volumen celular constante.

Finalmente, el tumor entra en la tercera fase, donde este activa los mecanismos fisiológicos con los que se produce metástasis de las células y se induce angiogénesis. De esta forma, ya en un entorno vascular, las células tumorales entran de nuevo en un estado en el que disponen de nutrientes suficientes para continuar su proliferación.

En este trabajo se estudia el desarrollo del tumor en la primera fase de crecimiento, donde las células obtienen los nutrientes mediante la difusión de estos a través de los tejidos circundantes y en la cual los agentes principales que determinan el crecimiento y proliferación de las células son los que se muestran en la Figura 1. También aparecen los dos fenotipos que aparecen en esta fase, que son la proliferación y la necrosis celular.

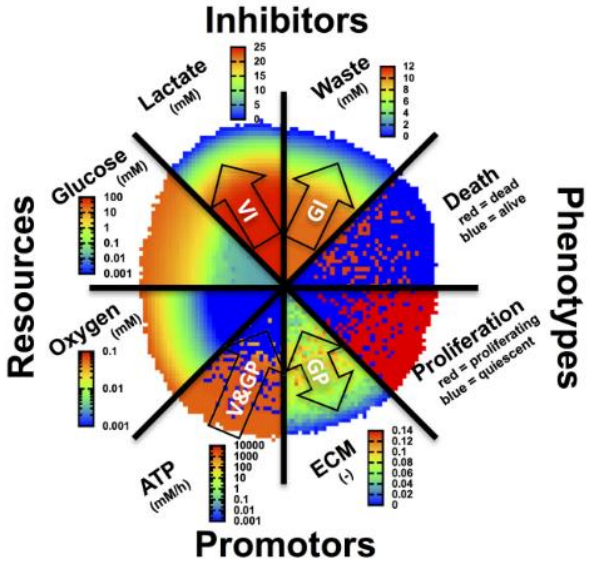

Figura 1. Sustancias determinantes del ciclo celular. [3]

Se elige el software de simulación PhysiCell como herramienta para este trabajo, ya que es un modelo denominado GSA (Global Sensitivity Analysis). Estos modelos se caracterizan, a diferencia de los LSA (Local Sensitivity Analysis), por tener en cuenta la variación y efecto de distintos parámetros simultáneamente en el sistema, utilizados especialmente en el estudio de biosistemas complejos como el cáncer.

PhysiCell realiza simulaciones en las que el desarrollo o necrosis de las células tumorales se determina en función de distintos parámetros, algunos de ellos se introducen como valores iniciales y otros se computan durante los ensayos.

De las sustancias presentadas en la Figura 1, la única que inicialmente PhysiCell considera es el oxígeno. El objetivo de este trabajo es, mediante la implementación de métodos vistos en diferentes artículos relacionados con el tema, crear un modelo que permita simular el metabolismo de las células cancerígenas en la primera etapa, de crecimiento avascular, por la que pasa el tumor.

La combinación de los diferentes agentes mostrados en la Figura 1 supone la modificación tanto del fenotipo de las células como de sus ciclos metabólicos, siendo vital para determinar estos la cantidad de ATP que cada célula es capaz de producir y el tipo de ciclo metabólico en el que se encuentra, pudiendo ser este aeróbico o anaeróbico.

El ATP es la 'energía' de la célula, la sustancia que una célula sintetiza y utiliza para poder realizar su ciclo celular. Para esta síntesis la célula requiere de oxígeno y glucosa y, en función de la cantidad disponible de ambas sustancias en el medio, la producción de ATP es una u otra. Más adelante, en el Apartado 2.2.1 se profundiza más en este concepto.

Por otro lado, el ciclo metabólico es el que determina el estado de una célula e indica la vía por la que la célula sintetiza el ATP. Ese ciclo depende del oxígeno disponible en el entorno de la célula, en el caso de haber suficiente, esta realizará su metabolismo de forma aeróbica mientras que, si la cantidad de oxígeno es baja y se encuentra por debajo de un valor umbral, el metabolismo celular será anaeróbico, aumentando el consumo de glucosa y no dependiendo del oxígeno. En este caso, además, se genera una sustancia de deshecho como es el lactato, que acidifica el medio.

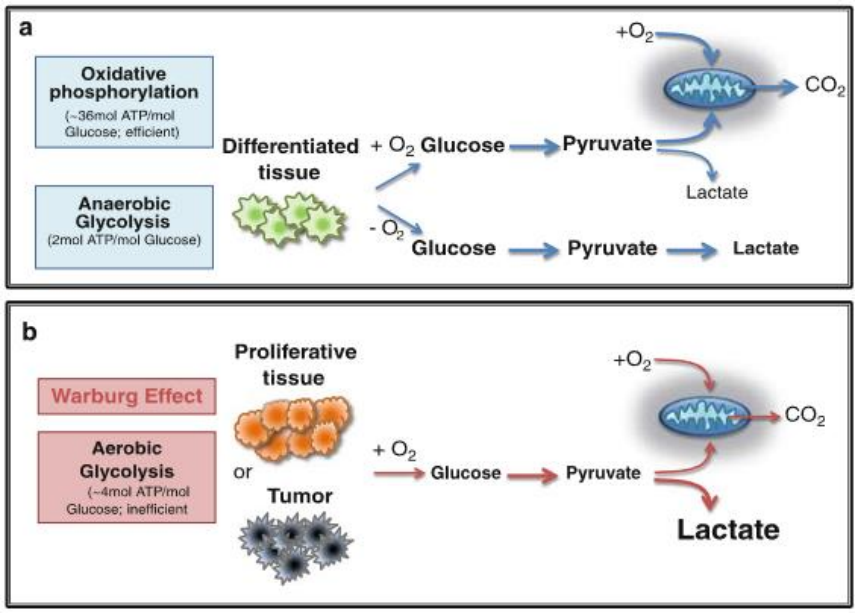

Figura 2. Esquema de oxidación aeróbica y anaeróbica. [4]

2. Modelo de Análisis Global: PhysiCell

PhysiCell [5] es un software de simulación que modela el entorno celular mediante una combinación de una aproximación continua, que describe el entorno químico y mecánico y un modelo basado en agentes que simula las células.

Para la aproximación continua se utiliza una parte del código del programa, BioFMV, el cual es el encargado de solucionar las ecuaciones de reacción-difusión, teniendo en cuenta los parámetros de difusión y decaimiento de la sustancia, así como los que determinan la secreción o el consumo de esta, pudiendo estas ser definidas por el usuario.

Se selecciona este programa para la realización de este trabajo debido, además de que las características mencionadas anteriormente se adaptan perfectamente al objetivo de este trabajo, por su avanzado estado de desarrollo y a la facilidad que brinda para introducir módulos programados por el usuario, pudiendo personalizar, en función del trabajo que se busca realizar, el funcionamiento de algunos aspectos del programa. De esta forma es posible adaptar, mediante nuevos módulos de código, los casos simulados para tener en cuenta un mayor número de parámetros y, tras comprobar la fiabilidad de los resultados obtenidos comparativamente con artículos, hacer un estudio cualitativo de los efectos en el modelo que supone la variación de alguno de estos parámetros o la combinación de ellos.

Dentro de los parámetros que es posible modificar, existen submodelos aplicables a procesos como el ciclo celular, cambios en mecánicas celulares, difusión, secreción u otros muchos factores que determinan el comportamiento del sistema general y que dan lugar a un extenso abanico de sistemas bilógicos que poder simular.

PhysiCell funciona mediante un archivo de entrada, de formato XML, en el cual figuran datos iniciales como el tiempo de simulación, introducido mediante iteraciones que corresponden a minutos, los parámetros iniciales que el modelo tiene en cuenta y su cantidad inicial o cantidad de iteraciones que deben pasar para que el modelo guarde las condiciones en ese instante, entre otros parámetros. En el Anexo 7.1 se muestra un ejemplo de fichero de entrada del programa.

Como indica en el Apartado 1, inicialmente PhysiCell tiene en cuenta únicamente la densidad de oxígeno presente en cada uno de los instantes de tiempo de la simulación. Partiendo de la concentración inicial y considerando el oxígeno como una sustancia que se degrada debido a difusión y es consumida por las células, se obtienen los cambios en su concentración mediante la ecuación de reacción-difusión, presentada en forma general ya que, teniendo en cuenta que a lo largo de este trabajo se simula el metabolismo celular, las sustancias que se tendrán en cuenta y que funcionarán mediante esta ecuación son, además del oxígeno, la glucosa y el lactato (Eq. 1).

$$\frac{\delta \rho_a}{\delta t} = D_a \nabla^2 \rho_a - \lambda_a \rho_a - \sum_{cells,k} \delta(x - x_k) W_k \left(U_{k,a} \rho_a\right) \; in \; \Omega \qquad (1)$$

Con condiciones de contorno de Dirichlet en $\delta\Omega$, donde:

- D_a es el coeficiente de difusión del agente [μ^2/min]
- λ_a es la ratio de decaimiento del agente [$1/min$]
- $\delta(x)$ es la delta de Dirac [-]
- x_k es la posición de la célula [μm]
- W_k es el volumen celular [μm^3]
- $U_{k,a}$ es la ratio de consumo celular del agente [$1/min$]

Las condiciones de contorno de Dirichlet que el modelo considera implican que los límites del volumen estudiado tienen el valor introducido, en este caso el mismo que el valor inicial de cada una de las sustancias, y que este no varía a lo largo de la simulación.

Con esto se pretenden simular unos ensayos en los cuales se considera que el volumen estudiado es una pequeña parte de un sistema mayor, que no ve afectada su composición debido a los cambios en este volumen estudiado. En este caso las concentraciones de las diferentes sustancias introducidas en el modelo tienen el valor inicial de cada sustancia, a pesar de que en el volumen en el que se introduzcan las células y en las unidades de volumen de su alrededor este valor cambie debido al gradiente de concentraciones que estas provocan.

Es posible configurar el modelo para que trabaje de dos formas distintas. La primera. de una forma determinística, en la que se trabaja con probabilidades de valor 0 o 1 tanto para la proliferación como para la necrosis, donde 0 es un resultado negativo y 1 es un resultado positivo, siendo esta la forma de trabajo del modelo inicialmente.

La segunda forma y la que se adopta más adelante en los diferentes ensayos es una forma estocástica, donde se trabaja no con valores de 0 o 1 sino con probabilidades entre estos dos valores, las cuales posteriormente se computan para determinar si la célula hace una cosa u otra.

En el modelo se toma como densidad de saturación de oxígeno 38.0 mm Hg [6]. Este parámetro determina el valor de oxígeno a partir del cual la proliferación celular no aumenta, sino que se mantiene constante. La cantidad computada de oxígeno, medida en $\%O_2$ en volumen, es la que determina estas probabilidades que una célula tiene para proliferar o necrosar, aumentando las probabilidades de este último caso si la densidad de oxígeno en el entorno cae por debajo de 15 mm Hg [5], valor que es considerado límite de proliferación y comienzo de necrosis.

Esta probabilidad se calcula mediante un factor que puede tomar únicamente dos valores, 0 o 1. Este factor le indica al modelo si la célula es capaz de proliferar (1) o no (0). En el caso de que los valores de los parámetros permitan a la célula proliferar, la probabilidad de proliferación se obtiene mediante un número generado aleatoriamente entre 0 y 1. De forma opuesta se procede para el valor de la probabilidad de necrosis.

Para tener en cuenta el efecto que una célula tiene sobre las demás, se actualizan las posiciones computando el efecto de las distintas fuerzas que actúan, considerando la Eq. 2 como la ecuación de equilibrio.

$$0 \approx \sum_{j \epsilon N(i)} \left(F_{cca}^{ij} + F_{ccr}^{ij}\right) + F_{loc}^{i} \qquad (2)$$

Donde:

- $N(i)$ son cada una de las células vecinas
- F_{cca}^{ij} y F_{ccr}^{ij} son fuerzas adhesivas y repulsivas célula-célula, respectivamente
- F_{loc}^{i} es la fuerza locomotriz neta

En el punto inicial de trabajo, el modelo solo tiene en cuenta las interacciones célula-célula a la hora de computar la velocidad y las trayectorias de migración celular, sin tener en cuenta el efecto que la ECM tiene sobre estas.

El oxígeno presente en cada iteración es la única sustancia que el modelo tiene en cuenta a la hora de evaluar la proliferación de cada una de las células en las distintas iteraciones.

Tal y como se puede ver en [5], a medida que la concentración inicial de oxígeno sea mayor, las células podrán proliferar durante un periodo de tiempo mayor, aunque a la hora de la proliferación en sí misma, esta cantidad no es determinante, ya que cada célula consume en cada iteración una cantidad de oxígeno concreta.

A mayor número de células, por tanto, la cantidad de oxígeno introducido inicialmente se agota antes, lo cual hace que estas sean capaces de proliferar durante un tiempo menor al que son capaces si el número fuese menor.

Por otro lado, es importante también el procesado de los datos por parte del programa tanto durante el tiempo de simulación como tras este. Inicialmente en el programa se introducen 4 células móviles en unas posiciones fijas, las cuales se localizan mediante sus coordenadas dentro de la ECM. El hecho de que sean células móviles implica un tipo específico de célula para el programa, en el que estas son capaces de moverse por la matriz, siempre bajo las condiciones de viscosidad dinámica dadas. Esto permitirá la migración celular y la formación, tras un tiempo, de clústers más o menos diferenciados en función de las posiciones finales de cada célula.

Uno de los algoritmos utilizados en este trabajo y que permite el post procesado de datos, programado en Python, permite diferenciar, en función de una distancia de búsqueda dada por el usuario, clústers celulares (algoritmo DBSCAN en C++). Este programa, cuyo código se encuentra en el Anexo 7.3, agrupa y separa por colores las células de una iteración determinada, indicando además las distancias en los ejes de coordenadas.

Este programa no se realiza partiendo de cero, sino que se utiliza un algoritmo previo capaz de diferenciar grupos de células mediante la asignación de un plano en Z, que plotea las células en 2D. De este programa se reutiliza la forma de agrupar las células en función de la distancia y de diferenciar por colores cada uno de los grupos, adaptándolo para poder obtener una representación en 3D, replicando las células como esferas.

El programa sigue la secuencia explicada en el gráfico de la Figura 3.

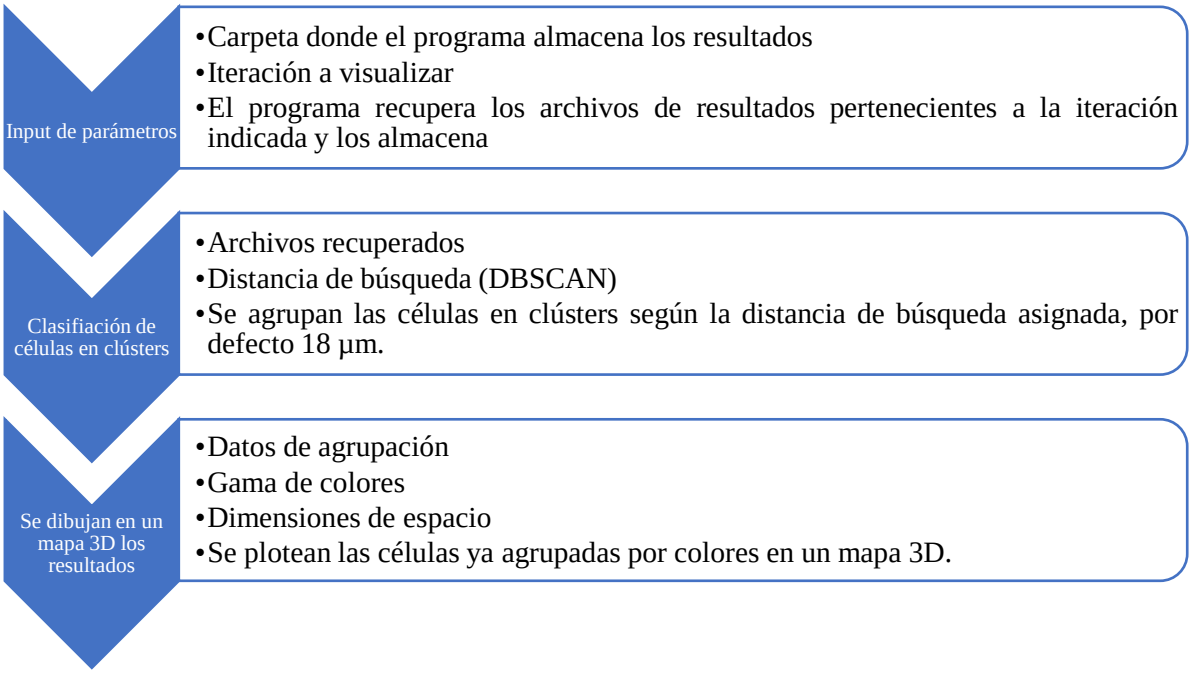

Figura 3. Secuencia de trabajo del modelo.

Tal como se indica anteriormente, el objetivo es la implementación en el modelo de una función en la que tenga en cuenta todas las variables que tienen un papel importante en la proliferación celular.

Según [3], existen sustancias que favorecen que el ciclo celular sea proliferativo mientras que hay otras que dificultan este ciclo y encaminan a la célula a un fenotipo de carácter necrótico, en el que no es capaz de proliferar y, poco a poco, pierde la capacidad de realizar sus funciones vitales hasta que, finalmente, muere.

El orden de los parámetros que este trabajo tiene como objetivo implementar en el modelo son, por un lado, la matriz extracelular, en la que se tiene en cuenta la densidad de colágeno que esta posee y que, si bien no está directamente relacionada con la proliferación y necrosis celular, sí que tiene un papel importante en la capacidad migratoria de las células, dependiendo de la viscosidad dinámica.

La siguiente sustancia que introducir en el modelo es la glucosa, sustancia cuya adición junto al oxígeno implica la introducción del efecto del ATP (Adenosín Trifosfato).

La última sustancia añadida al modelo es el lactato, el cual se genera al producirse una falta de oxígeno disponible en el entorno para la célula. Esto supone que la célula entre en un ciclo anaeróbico, en el cual obtiene el ATP necesario para realizar sus funciones únicamente de la glucosa, siendo un proceso menos efectivo y que tiene como resultado la producción de esta sustancia. La aparición y posterior aumento de esta sustancia en la célula implica una acidificación del entorno celular, que, en el caso de sobrepasar cierto nivel, puede desembocar en necrosis. La introducción de cada uno de los agentes indicados se explica más en detalle en los Apartados 2.1, 2.2 y 2.3.

Los parámetros de referencia utilizados en el modelo son los mostrados en la tabla de la Figura 3.

Symbol	Parameter	Value	Unit
$\rho^0_{O_2}$	Oxygen Initial Density	38.0	mmHg
D_{O_2}	Oxygen Diffusion Coefficient	1.0e5	μm^2/min
λ_{O_2}	Oxygen Decay Rate	0.1	1/min
U_{k,O_2}	Oxygen Uptake Rate (per cell, k)	10.0	1/min
$\rho^0_{Collagen}$	Collagen Initial Concentration	[2.5, 4.0, 6.0]	mg/mL
μ	Drag Coefficient	[7.96, 18.42, 39.15]	Pa · s
R	Cell Radius	8.4	μm
R_A	Maximum Cell Adhesion Distance	1.25R	μm
c_{cca}	Cell-Cell Adhesion Coefficient	7.2	-
c_{ccr}	Cell-Cell Repulsion Coefficient	380	-
T_{Ki67-}	Cell Quiescence Time	6.5	h
T_{Ki67+}	Cell Proliferation Time	15.5	h
r_D	Cell Death Rate	0.00319	1/h

Figura 4. Parámetros de referencia del modelo [6].

Estos valores son únicamente datos de partida, con lo que la mayoría se mantiene constante por no ser un objetivo de estudio en este trabajo (ratio de consumo de oxígeno, coeficiente de difusión…) y otros sufren variaciones debido a ser parámetros clave para el estudio que se realiza a lo largo de este trabajo (densidad inicial de oxígeno).

1.1. Análisis previo

Los resultados que se muestran en este apartado son obtenidos realizando varias simulaciones con distintas densidades iniciales de oxígeno, sin realizar ninguna modificación sobre el programa, para poder tener una primera visión de cómo funciona el modelo y comprobar las diferencias esperadas.

En la Figura 5 se muestra la comparativa entre tres simulaciones realizadas con distinta concentración inicial de oxígeno (17, 38 y 60 mm Hg) y distintas células iniciales (1 y 5), observándose que, al únicamente ser el oxígeno el que marca una diferencia en la simulación, a mayor concentración inicial, las células son capaces de proliferar durante mayor tiempo, sin desembocar en la proliferación en sí.

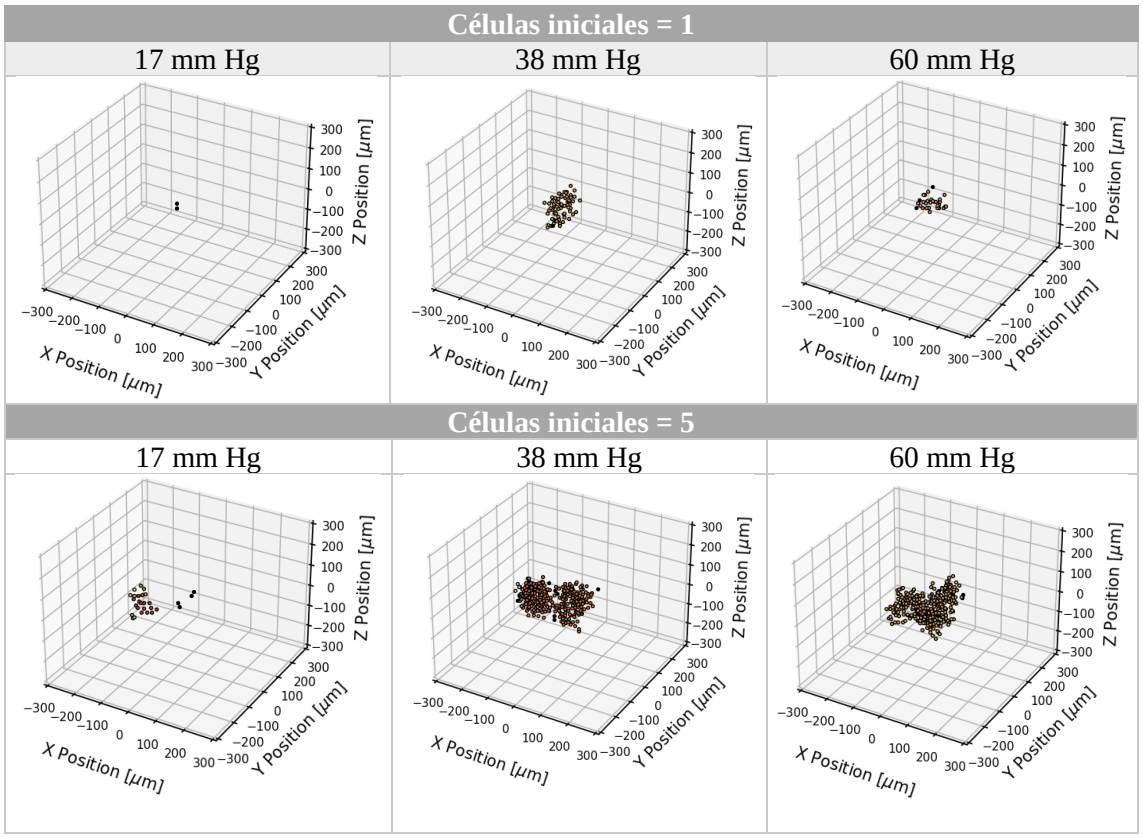

Figura 5. Resultados obtenidos por PhysiCell mediante la variación de la cantidad de células inicial de la simulación.

En las dos situaciones para 17 mm Hg de densidad inicial de oxígeno, se observa que el número final de células es bastante inferior respecto a los otros casos. Esto es debido a que el umbral mínimo de oxígeno que las células del modelo requieren para proliferar es de 15 mm Hg, por lo que, con este valor de densidad inicial, se llega a un valor por debajo de este umbral durante la simulación, lo que hace que las células dejen de proliferar y mueran.

2. Agentes Modelados

2.1. Matriz extracelular

2.1.1. Implementación en el modelo

Una vez realizadas las comprobaciones iniciales para verificar el comportamiento esperado del modelo, el siguiente paso es la modificación del código para conseguir que este tenga en cuenta la acción de la matriz extracelular en la migración celular y no considere este parámetro como un valor fijo que no afecta al ensayo.

Esto se consigue mediante la adición al modelo de una función que tiene en cuenta, no solo el oxígeno presente y disponible para cada célula en cada step simulado, sino la densidad de colágeno en la matriz extracelular, la cual se introduce como parámetro inicial en el archivo XML y que permanecerá constante en toda la simulación.

Para poder comparar los resultados obtenidos en el modelo se tienen en cuenta las densidades que se citan en [6]. De esta forma se simplifica la comprobación respecto a que el código introducido al modelo funciona correctamente, pudiendo realizar una comparación directa entre las distribuciones finales obtenidas en el artículo y las obtenidas en el ensayo mediante PhysiCell.

Para poder tener en cuenta el efecto de la densidad de la matriz extracelular en la migración y proliferación celular, la nueva función sustituye a la que PhysiCell tiene por defecto. Esta lee el valor de densidad de colágeno que la matriz extracelular tiene y, en función de este, asigna un valor a la viscosidad dinámica de la matriz para el ensayo. Esta viscosidad dinámica es la que el programa computa para obtener la velocidad de migración de cada una de las células.

La nueva función que se implementa en PhysiCell únicamente lee los tres valores de densidad de colágeno indicados en [6]: 2.5, 4.0 y 6.0 mg/mL. Cada uno de estos valores de densidad tiene asignado un valor de viscosidad dinámica indicados en la Tabla 1 [6].

Densidad (mg/ml)	μ (-)
2.5	7.96
4.0	18.42
6.0	39.15

Tabla 1. Resumen de las viscosidades dinámicas (μ) aplicadas para cada densidad de colágeno [6].

Para cuantificar numéricamente la capacidad de migración de las células, con cada uno de los valores anteriores se obtiene, mediante la Ecuación (*3*) el valor actualizado de la velocidad:

$$v_i = \frac{v_{celular\ estandar}}{\mu} \tag{3}$$

Donde la velocidad celular estándar es el valor obtenido mediante la función que PhysiCell posee por defecto para computar este dato, el cual depende de la motilidad de las células. Para los ensayos realizados a lo largo de este trabajo, todas las células se consideran motiles, esto quiere decir que se considera que las células tienen la capacidad no solo de proliferar sino de moverse por el dominio dependiendo de diversos factores relacionados tanto con la matriz extracelular como con la propia célula.

El programa determina la dirección preferente de migración de la célula, considerándola completamente aleatoria y, una vez obtenida, la multiplica por el valor de velocidad de migración. Este valor no considera fuerzas adhesivas o repulsivas, donde el valor dado a esta variable es:

$$velocidad_{migración} = 1.0\ \mu m/min$$

La v_i es la que PhysiCell tiene en cuenta para computar la migración de cada una de las células en el modelo.

Con la adición de estas fuerzas al modelo, se añade un nuevo término a la ecuación de equilibrio de fuerzas, F^i_{drag}, correspondiente a las fuerzas disipativas originadas por la matriz. La Ecuación (*2*) queda de la siguiente manera:

$$0 \approx \sum_{j \epsilon N(i)} \left(F^{ij}_{cca} + F^{ij}_{ccr}\right) + F^i_{drag} + F^i_{loc} \tag{4}$$

Se puede calcular el valor de estas fuerzas disipativas [6] como:

$$F^i_{drag} = -\mu v_i \tag{5}$$

Donde v_i es la velocidad celular, que se puede calcular simplificando y reordenando la Ecuación (*2*):

$$v_i = \frac{1}{\mu}\left(\sum_{j \epsilon N(i)} \left(F^{ij}_{cca} + F^{ij}_{ccr}\right) + F^i_{loc}\right) \tag{6}$$

Como se puede observar en la Ecuación (*5*), la velocidad de las células depende de la viscosidad de la matriz, la cual, a su vez, depende de la densidad de colágeno presente.

2.1.2. Resultados celulares

Una vez aplicados los cambios explicados en el apartado anterior al modelo para tener en consideración la densidad de la matriz extracelular en el desplazamiento de las células, se realizan varios ensayos para comprobar que los resultados obtenidos mediante el modelo coinciden con los mostrados en [6].

Tanto en [6] como en los ensayos realizados con PhysiCell, el punto de partida es una única célula y se considera un tiempo de simulado de 5 días, los cuales se deben introducir en el modelo en minutos (7200 minutos). La comparativa entre los resultados del artículo y los obtenidos con la nueva aproximación añadida a PhysiCell se muestra en la Figura 6.

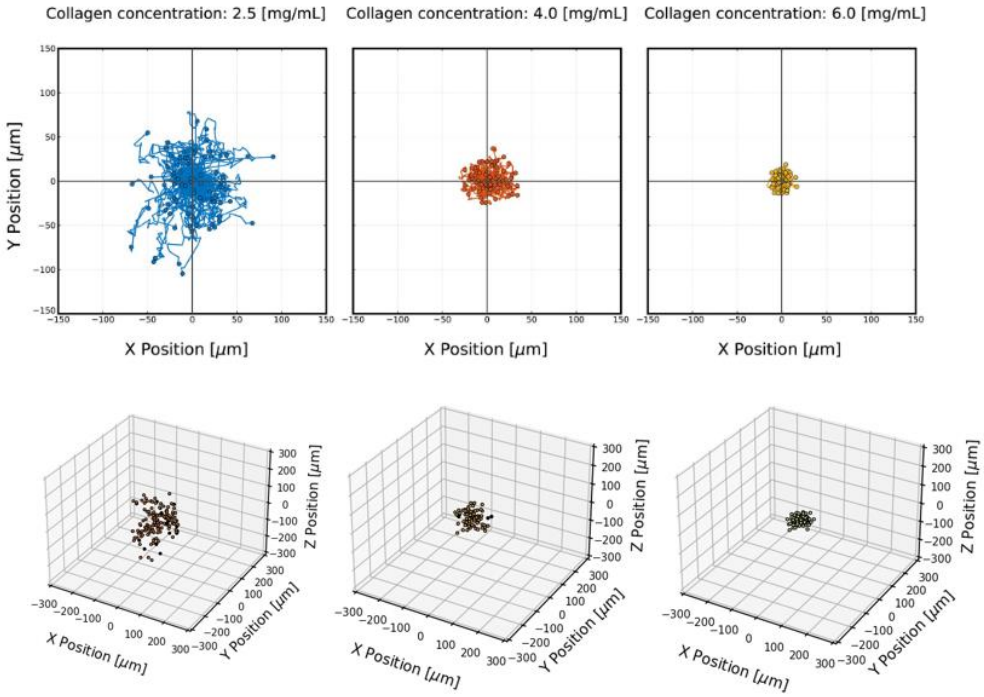

Figura 6. Comparativa de resultados para una célula tras 5 días de simulación entre los resultados obtenidos en [6] (arriba) y los resultados obtenidos mediante PhysiCell (abajo).

Los resultados obtenidos mediante los ensayos en PhysiCell son visualmente bastante similares a los obtenidos en el artículo. Para el caso de una densidad de colágeno de 2.5 mg/ml en la matriz, tanto en [6] como en la simulación realizada en PhysiCell, la dispersión de las células es bastante similar, siendo difícil agruparlas en el step final en clústers diferenciados.

En el otro extremo, en una matriz con una densidad de colágeno de 6 mg/ml, en la que la migración se hace más complicada por las fuerzas que se oponen al movimiento que la propia matriz ejerce sobre las células, la agrupación en un clúster diferenciado es mucho mayor, obteniendo un radio de clúster bastante similar en ambos casos. En el caso intermedio, de densidad de colágeno de 4.0 mg/mL, la separación entre las células en el último step de simulación es más evidente que en el caso de 6.0 mg/mL, aunque el clúster formado, menos compactado, sigue siendo claramente visible.

Dado que se observa que los resultados obtenidos mediante la adaptación programada sobre PhysiCell se asemejan bastante a los obtenidos en el artículo de [6], se considera que la aproximación del modelo es válida y, por tanto, los resultados que se obtienen posteriormente son válidos en cuanto a la acción de la matriz sobre las células posicionadas.

2.2. Glucosa

2.2.1. Implementación en el modelo

Tener en cuenta únicamente el oxígeno disponible en cada instante no es suficiente para un modelado preciso del ciclo celular, ya que este no solo es determinado por la cantidad de oxígeno disponible sino por la combinación de varias sustancias. Estas sustancias principales que tienen un papel importante en el ciclo celular se muestran en la Figura 1, y su distribución espacial en la que aparecen dentro de un clúster celular. De todas ellas, este apartado se enfoca en la glucosa y en cómo introducirla para que sea tenida en cuenta de forma coherente por el modelo.

El ciclo por el cual la célula es capaz de producir este ATP es el llamado ciclo de Krebs, el cual forma parte de la respiración aeróbica de la célula y que engloba una serie de reacciones químicas originadas en la mitocondria de las cuales, por simplificación, únicamente se tiene en cuenta la producción de ATP a partir de la glucosa y el oxígeno.

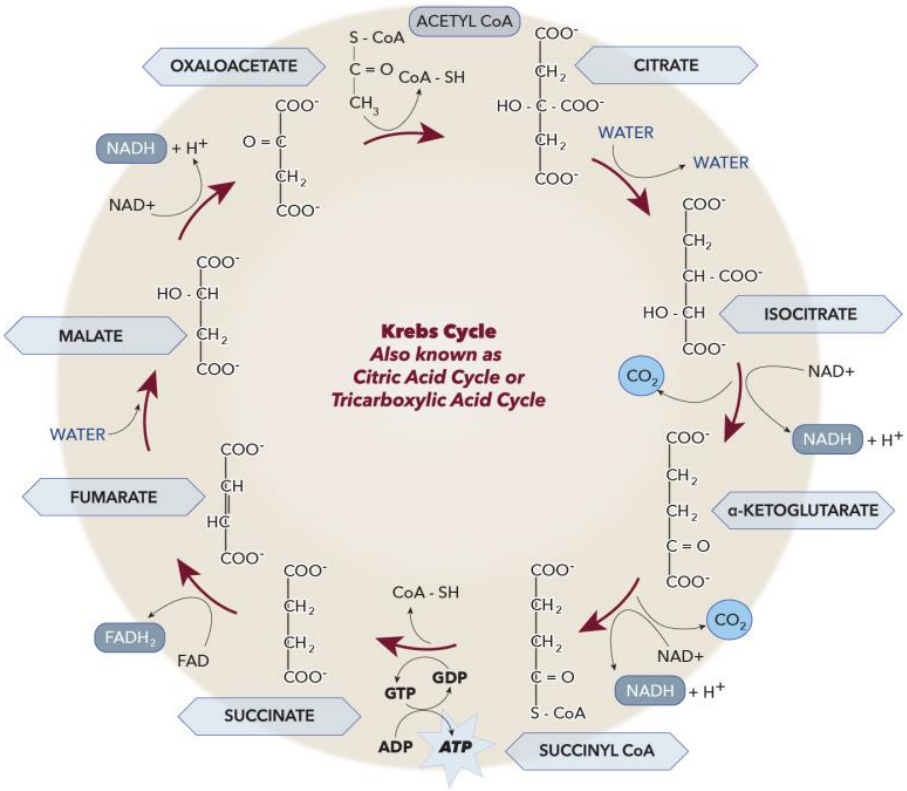

Figura 7. Ciclo de Krebs detallado [7].

En esta nueva aproximación del modelo, por tanto, ya no se tiene en cuenta el nivel de oxígeno para determinar si una célula prolifera o necrosa, sino que se determina en base a la cantidad disponible en cada instante de oxígeno y glucosa, que a su vez determinan la cantidad de ATP que la célula puede producir. Este valor, posteriormente se compara con el valor límite referido en la literatura [3] para determinar las probabilidades de proliferación o necrosis de la célula.

El funcionamiento del módulo consiste en la aplicación del método y las ecuaciones utilizados en [3] para la adecuación de un modelo basado en el crecimiento y supervivencia basado en ATP. PhysiCell trabaja con probabilidades, más en concreto las probabilidades de proliferación o necrosis de cada una de las células. Estas probabilidades vienen dadas por las Ecuaciones (*7*) y (*8*).

$$k_{div} = k_{div}^{max} H(\rho_{ATP} - \rho_{ATP}^{min}) \quad (7)$$

$$k_{nec} = k_{nec}^{max} H(\rho_{ATP}^{min} - \rho_{ATP}) \quad (8)$$

Donde ρ_{ATP} es la producción de ATP en [mM/h] y que se determina por la Ecuación (*9*).

$$\frac{d[ATP]}{dt} = 2q_G + \frac{17}{3} q_{O2} = \rho_{ATP} \quad (9)$$

Y donde ρ_{ATP}^{min} es el valor mínimo de producción de ATP que delimita el umbral entre la proliferación y la necrosis, siendo este valor, según [3]:

$$\rho_{ATP}^{min} = 900\ mM/h$$

Por otro lado, la función $H(x)$ es la función de Heaviside, o función escalón, donde esta vale 1 si $x > 0$ y 0 si $x \leq 0$.

Además, los valores para k_{div}^{max} y k_{nec}^{max} se computan interiormente en el modelo en función del ciclo celular en el momento de la obtención de este valor.

Para el modelo en este caso se considera este valor como el mínimo necesario para que la célula sea capaz de proliferar, aunque, según el artículo, el valor óptimo de producción de ATP para que una célula pueda realizar sus funciones con normalidad está en torno a $1100\ mM/h$. En la Figura 8 se muestra la dependencia en la producción de ATP descrita en [3] en función de la concentración de oxígeno y glucosa. La línea negra indica este valor de producción mínimo de ATP (ρ_{ATP}^{min})

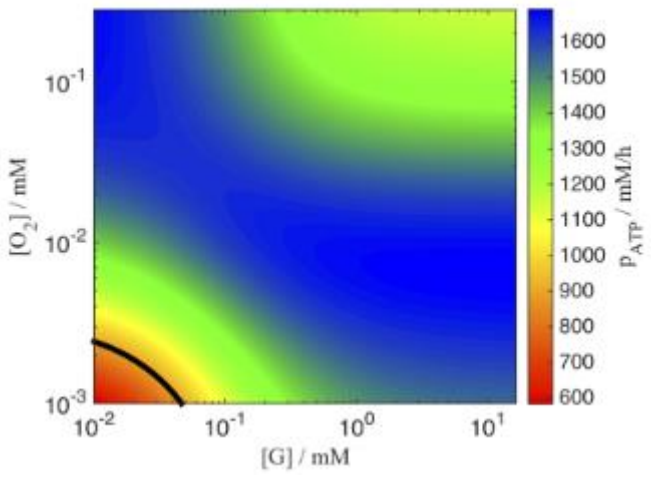

Figura 8. Ratio de producción de ATP en función de la concentración de oxígeno y glucosa [3].

Para obtener los valores de q_G y q_{O2}, en [3] se obtiene la distribución en base a unos experimentos realizados en células de la línea EMT6/Ro variando las concentraciones medias, habiéndose obtenido las Ecuaciones (*10*) y (*11*).

$$q_G = V_G^{max} \frac{[G]}{[G]k_G^G} \tag{10}$$

$$q_{O2} = V_{O_2}^{max} \frac{[O_2]}{[O_2]k_{O_2}^{O_2}} \tag{11}$$

En estos experimentos se ve, además, que estas ratios de consumo de oxígeno y glucosa son interdependientes de ambas sustancias. En la Figura 9 se muestra la dependencia de una y otra en función de ambas.

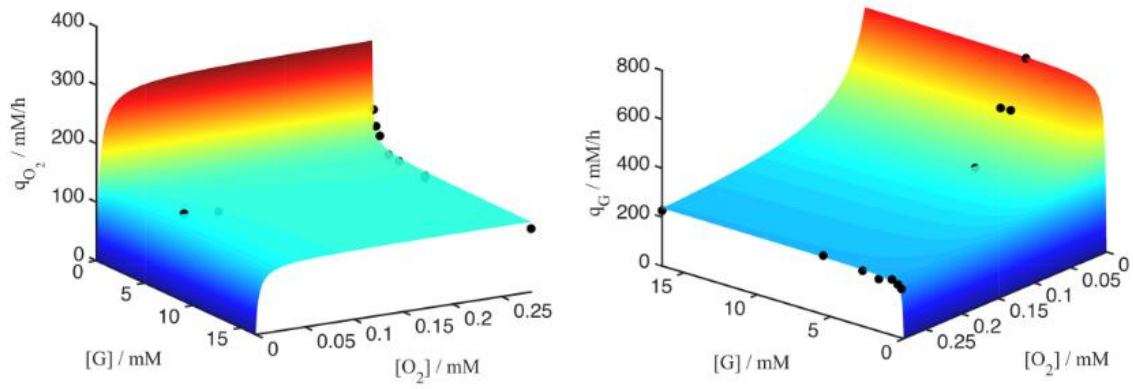

Figura 9. Dependencia de las ratios de consumo con ambas sustancias [3].

Y, por comparación directa de estos resultados, se encuentra el valor para los términos V_G^{max} y $V_{O_2}^{max}$ con los que tener en cuenta la interdependencia observada con los consumos obtenidos experimentalmente. Esta relación se muestra en la Ecuación (*12*) y la Ecuación (*13*).

$$V_G^{max} = q_G^{max}\left(1 - \left(1 - \frac{q_G^{min}}{q_G^{max}}\right)\frac{[O_2]}{[O_2] + k_G^{O_2}}\right) \tag{12}$$

$$V_{O_2}^{max} = q_{O_2}^{max}\left(1 - \left(1 - \frac{q_{O_2}^{min}}{q_{O_2}^{max}}\right)\frac{[G]}{[G] + k_{O_2}^{G}}\right) \tag{13}$$

Utilizando como constantes para estas ecuaciones las que aparecen en la tabla de la Figura 10.

Parameter	Unit	Value
k_G^G	*mM*	0.068
$k_G^{O_2}$	*mM*	0.031
q_G^{min}	*mM/h*	186.67
q_G^{max}	*mM/h*	706.67
$k_{O_2}^{O_2}$	*mM*	0.031
$k_{O_2}^{G}$	*mM*	0.100
$q_{O_2}^{min}$	*mM/h*	120.00
$q_{O_2}^{max}$	*mM/h*	306.66

Figura 10. Parámetros de modelo para glucosa y oxígeno. [3]

Además, los parámetros $[O_2]$ y $[G]$ son las concentraciones de oxígeno y glucosa en cada iteración, medidos en las mismas unidades que las constantes que aparecen en la tabla de la Figura 10, donde:

$$mM = \frac{mmol}{l}$$

Para ello hay que hacer un cambio de unidades ya que PhysiCell trabaja con gramos y minutos. Para modificar las unidades del oxígeno, se debe realizar una regla de tres, mediante los datos indicados en [3]. En primer lugar, el oxígeno se introduce al modelo en $mm\,Hg$. Para la conversión inicial de estas unidades a $\%O_2$, se tiene la relación:

$$38\,mm\,Hg = 5\%\,O_2$$

Con ella es posible obtener la cantidad de oxígeno en cada iteración en $\%\,O_2$.

$$\%O_2 = O_2[mm\,Hg] * \frac{5}{38} \tag{14}$$

A su vez, en el mismo artículo se da una relación directa entre el $\%O_2$, mM. Esta relación dada es la indicada en la Tabla 2.

$\%O_2$	Molaridad (mM)
5	0.07
20	0.28

Tabla 2. Relación entre $\%O_2$ y mM [3].

Con esta relación, es posible hacer una conversión directa a partir de la cantidad de oxígeno en cada iteración.

$$O_2[mM] = \%O_2 * \frac{0.28}{20} \tag{15}$$

El valor en mM de la glucosa se obtiene de una forma más simple ya que se consigue únicamente mediante su peso molecular ($180\,\frac{g}{mol}$):

$$glucosa\,[mM] = glucosa\frac{\left[\frac{mg}{ml}\right]}{180} * 1000 \tag{16}$$

Con estas conversiones realizadas a ambas variables, ya en las unidades que las ecuaciones implementadas utilizan, es posible aplicar las Ecuaciones (*9*) - (*13*) para obtener el valor de ATP que se genera en cada iteración del modelo.

Con esta cantidad de ATP obtenida, mediante las Ecuaciones (*7*) y (*8*) es posible calcular las probabilidades de proliferación y necrosis para cada una de las células presentes en cada iteración del modelo.

Por la propia definición de ambas variables, cuando $k_{div} > 0$ entonces la probabilidad de necrosis deberá ser necesariamente $k_{nec} = 0$ y viceversa, ya que una célula que sea capaz de proliferar implica que tiene el suficiente oxígeno y glucosa para realizar su ciclo y, por tanto, para evitar la necrosis.

Esto ocurre por la simplificación aplicada a lo largo del trabajo, en la que el límite inferior del oxígeno para que una célula sea capaz de proliferar es el mismo con el cual comienza la necrosis. En el caso de elegir un valor distinto para el comienzo de la necrosis, habría un intervalo de densidades de oxígeno en los que la célula no proliferaría ($k_{div} = 0$) pero tampoco entraría en necrosis ($k_{nec} = 0$) este sería el estado de quiescencia, donde la célula únicamente es capaz de realizar su ciclo celular sin proliferación y donde se aumenta su capacidad migratoria.

2.2.2. Resultados celulares

Se muestran los resultados obtenidos tras las modificaciones realizadas para que el modelo considere el efecto de la glucosa sobre la proliferación celular considerando 5 días de simulación. Teniendo en cuenta tanto la cantidad de oxígeno como los niveles de glucosa para la determinación de los niveles de ATP que cada célula es capaz de producir y que determina su capacidad de proliferación.

En la Figura 11 se muestran los resultados obtenidos para cada uno de los ensayos, variando la densidad inicial de oxígeno de la simulación y la densidad de la matriz.

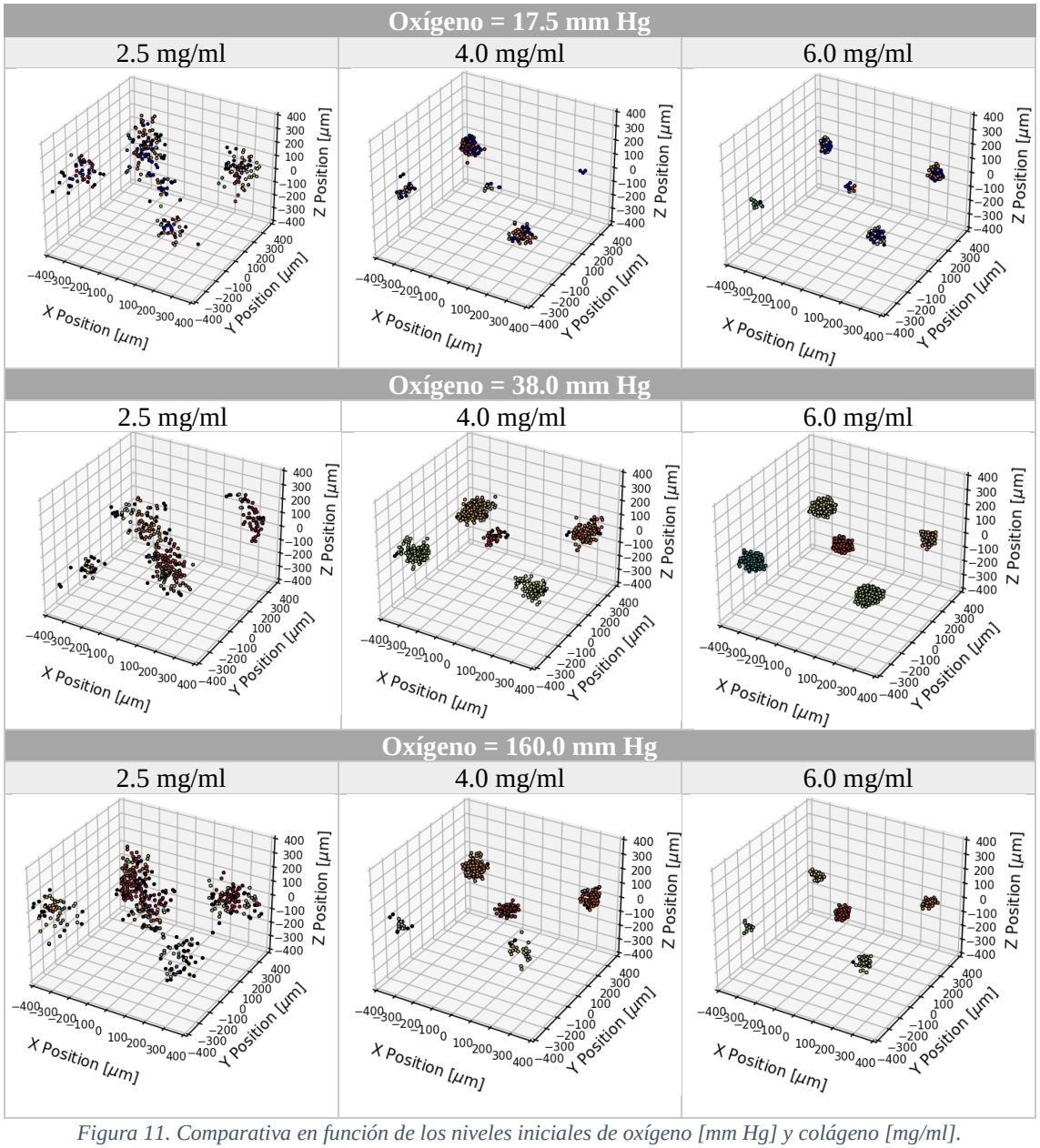

Figura 11. Comparativa en función de los niveles iniciales de oxígeno [mm Hg] y colágeno [mg/ml].

Si la concentración de oxígeno está por encima del valor límite de saturación de oxígeno (casos de 38.0 y 160.0 mg/ml), los resultados obtenidos son bastante parecidos, aunque sí que existen ciertas diferencias entre ellos ya que por debajo de 38.0 mm Hg las células no consumen la misma cantidad de oxígeno y su proliferación varía. Es por ello por lo que se observa que, a pesar de tener suficiente oxígeno para poder realizar el ciclo de Krebs sin problemas y producir ATP, la proliferación celular difiere para esos dos valores.

Según [6] el valor límite a partir del cual las células cesan su labor de proliferación está en 15 mm Hg, es por ello por lo que, en el caso de 17.0 mg/ml de densidad inicial de oxígeno, donde el modelo tiene tiempo de proliferar y, en cierto momento, el oxígeno disponible cae por debajo de este valor, se observa que las células cesan su proliferación y entran en estado de necrosis, con lo que el volumen celular final es mucho menor en comparación con los otros dos casos.

También se realizan simulaciones para comprobar los resultados que se obtienen con el modelo al mantener la densidad de colágeno y la concentración inicial de oxígeno y realizar la variación en la cantidad de glucosa inicial introducida en el modelo.

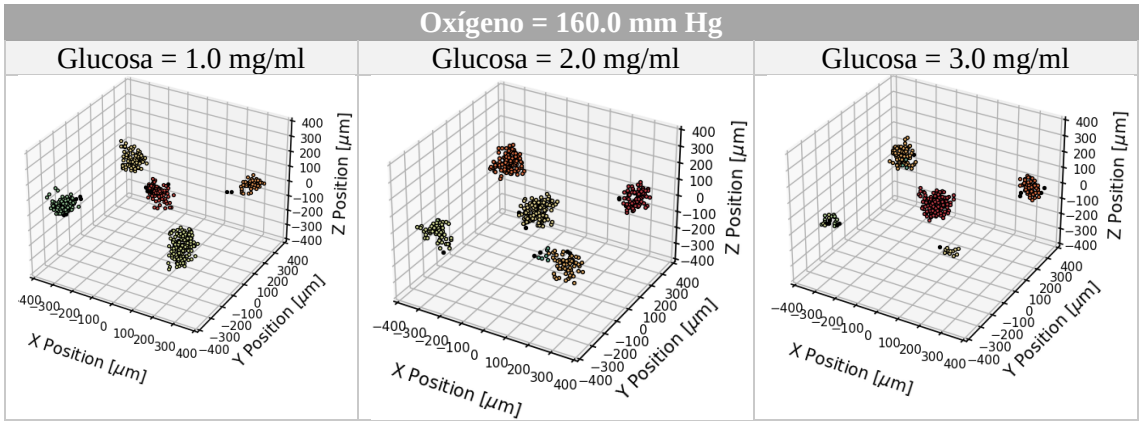

Figura 12. Comparativa en función de la densidad inicial de glucosa.

Los niveles de glucosa no tienen un papel importante en la proliferación celular en comparación con la cantidad de oxígeno presente en la simulación. Este fenómeno se puede explicar mediante la Ecuación (9) aplicada para el cálculo del ATP producido por una célula en base a la glucosa y el oxígeno presentes. En ella se ve que, siempre que los niveles de glucosa sean suficientes para permitir la proliferación, el consumo de oxígeno tiene más peso en la ecuación. Es por ello por lo que los resultados obtenidos son similares entre sí a pesar de las diferencias en cuanto a la concentración inicial de glucosa.

En la Figura 13 se muestra la comparativa entre los resultados obtenidos en los ensayos realizados mediante PhysiCell y los resultados obtenidos en [6]. La clara similitud entre los resultados obtenidos en el modelo y los mostrados en el artículo lleva a concluir que la aproximación programada en PhysiCell es válida y concuerda con lo esperado.

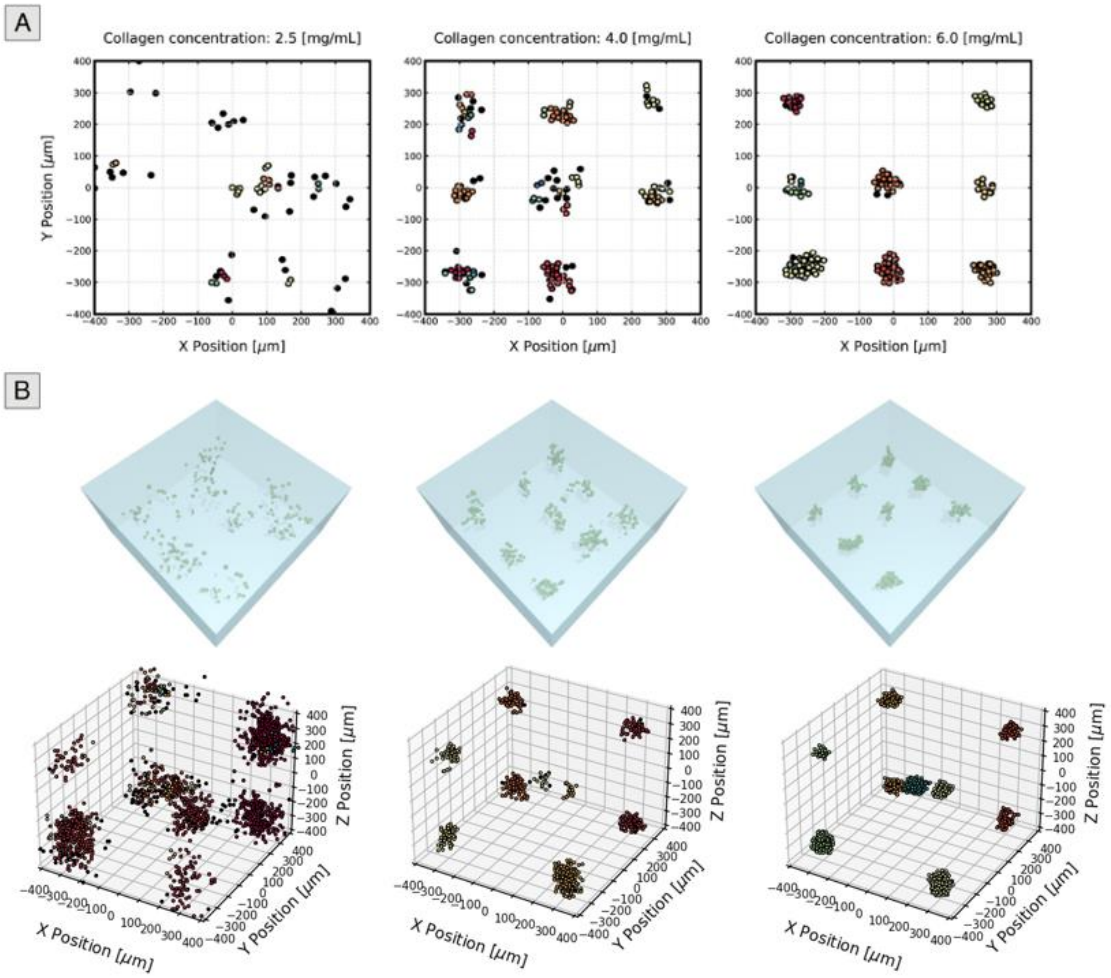

Figura 13. Comparativa entre ensayos con PhysiCell (figuras inferiores) y resultados de [6] (figuras A y B).

2.3. Lactato

2.3.1. Implementación en el modelo

El lactato es una sustancia generada por las células al entrar en el ciclo anaeróbico. Este ciclo se presenta como respuesta de la célula a una disminución por debajo de un valor mínimo en la cantidad de oxígeno presente en la matriz extracelular disponible para realizar el ciclo celular de forma aeróbica.

Las células en este caso son capaces de generar el ATP necesario para realizar sus funciones únicamente mediante un cambio de ciclo celular en el que se aumenta el consumo de la glucosa presente en la matriz como único medio de producción de este, sin embargo, esto conlleva la acumulación del piruvato generado en la glicolisis, que no puede ser descompuesto sin la presencia de oxígeno y que, cuando se acumula en los alrededores de la célula por encima de un valor límite, resulta en una acidificación del medio que da lugar, en última instancia, a la necrosis de la célula. En la Figura 2 se muestra el proceso del metabolismo anaeróbico de la célula.

Esta situación se da en los volúmenes interiores de los clústers celulares con anterioridad a cualquier otro lugar de la matriz donde el oxígeno es mucho más reducido y hay una mayor cantidad de células que demandan esta sustancia.

El efecto del lactato en el modelo como sustancia con un papel fundamental se introduce mediante la modificación de la probabilidad de necrosis de las células en función del valor que adopte la cantidad de esta sustancia para cada célula y de si este supera un valor límite determinado. Este valor mínimo estimado para la consideración de la entrada por parte de las células en un ciclo anaeróbico se obtiene de [6], donde este valor se toma como límite de proliferación.

$$\rho_{hipoxia} = 15\ mm\ Hg$$

De la misma forma que la ratio de producción de lactato se aproxima en [3], se considera que, por cada molécula de glucosa que la célula utiliza para la generación de ATP, produce dos moléculas de lactato, de forma que, en términos del modelo, la cantidad de lactato generada por una célula en cada instante de tiempo se puede calcular mediante su ratio de consumo de glucosa según la Ecuación (*17*).

$$\rho_L = 2 * q_G \tag{17}$$

En [3] se muestra la producción de lactato en un sistema en función del oxígeno y de la glucosa que las células tengan disponible en ese momento

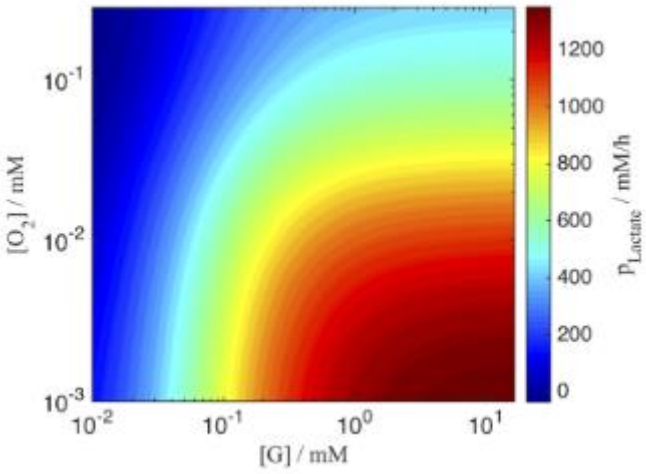

Figura 14. Comparativa entre simulaciones y resultados de [3].

Liberándose esta cantidad de lactato en el dominio alrededor de la célula. Una vez añadida esta variable, el valor límite tenido en cuenta en este modelo para la consideración del lactato es el que se indica en el artículo:

$$\rho_{L,max} = 20\ mM$$

En cuanto el valor de lactato presente para una célula en un instante determinado supera este valor límite, la probabilidad de necrosis de la célula se hace máxima, suponiendo la necrosis de la célula para todo lo que resta de modelo. Además, no es necesario que el nivel de lactato en la célula llegue a estos niveles, sino que tan pronto como comienza a aparecer lactato la probabilidad de necrosis aumenta según la Ecuación (*18*)

$$k_{nec} = k_{nec}^{max} * H(\rho_{ATP}^{max} - \rho_{ATP}) * \frac{[L]^n}{([L]^{max})^n - [L]^n} \quad (18)$$

Donde los parámetros ya vistos se obtienen de la misma forma que en el Apartado 2.2.1, [L] es el valor de concentración de lactato en cada instante computado para cada una de las células y n es el coeficiente de Hill, que determina la forma de transición en la muerte celular. Para valores bajos del coeficiente de Hill se obtendrán transiciones más suaves mientras que para valores altos la transición será mucho más brusca. En este caso, siguiendo la línea de [3] se aplica un coeficiente de Hill n = 2.

En las simulaciones realizadas en este apartado, donde se considera no sólo el lactato introducido sino todas las sustancias anteriores, en el instante inicial del modelo se considera que los niveles de lactato están a cero. De esta forma es posible evaluar que la producción se asemeje a la descrita en [3].

Tal y como están definidas las condiciones de producción de lactato, la cantidad final de esta sustancia está fuertemente relacionada con la cantidad inicial de oxígeno, ya que de ella dependerá el instante en el que el oxígeno del modelo se sitúe por debajo del nivel mínimo que determina el ciclo aeróbico y se comience a acumular lactato en las inmediaciones.

Para forzar al modelo a la acumulación de lactato, tras varias iteraciones se identifica un valor de 17.0 mm Hg de densidad de oxígeno inicial como un valor que permite realizar estas simulaciones, ya que, de esta forma, las células del modelo son capaces de reproducirse al comienzo para luego entrar en su ciclo anaeróbico por el déficit de oxígeno en el entorno y que tengan la posibilidad de generar lactato, pudiendo ver unos resultados significativos al final de la simulación.

Con otros valores superiores el ensayo no dispone de suficiente tiempo para entrar en la fase de generación de lactato y que se pueda observar con claridad esta producción. Por otro lado, en el caso de introducir valores inferiores, las células obtenidas al final de la simulación quedan todas en necrosis.

Dado que el modelo tiene problemas a la hora de simular ensayos con tiempos muy altos, se mantiene el tiempo de la simulación de 5 días, aunque en [3] se indica que los valores de lactato comienzan a ser relevantes entre 17 y 24 días de ensayo. Este problema viene debido a la forma que PhysiCell posee para actualizar las características de cada una de las células presentes en el modelo en cada una de las iteraciones (fenotipo, ciclo celular, cantidad de sustancias disponible, etc.) y se suple modificando el valor inicial del oxígeno de cada ensayo tal y como se ha indicado.

2.3.2. Resultados celulares

Los resultados obtenidos bajo estas condiciones explicadas en el apartado anterior, para una densidad inicial de oxígeno de 17.5 mm Hg, son los que se muestran a continuación.

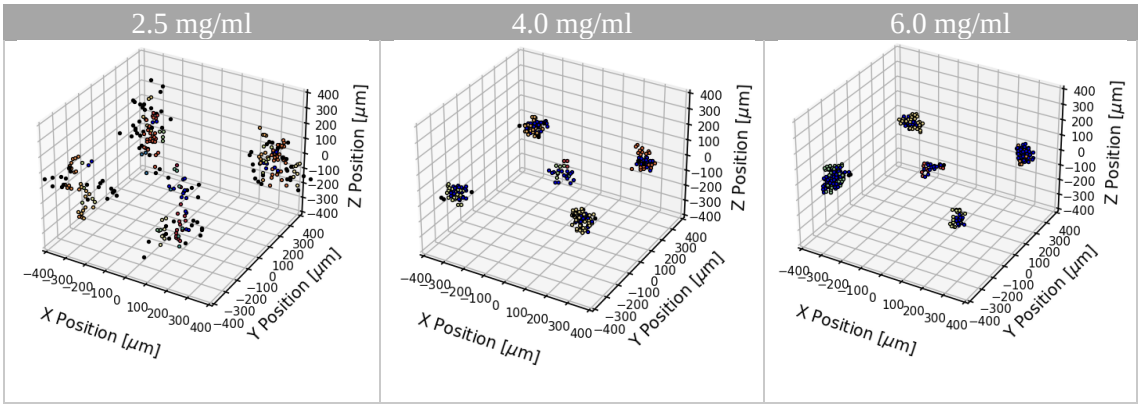

Figura 15. Resultados obtenidos para densidad de colágeno 2.5 mm Hg (izquierda), 4.0 mg/ml (centro) y 6.0 mm Hg (derecha). Densidad inicial de oxígeno de 17.5 mm Hg.

Para densidades de colágeno bajas, donde las células no forman clústers definidos, algunas mueren (coloreadas en azul) debido a la acumulación de lactato en las zonas cercanas al volumen donde se halla el grueso de las células. Sin embargo, estas muertes no siguen un patrón claro, sino que aparecen en ciertas células que pertenecen a estos volúmenes que no se pueden distinguir como clústers. Estas células, aunque no pertenecen a un clúster delimitado, están afectadas por la difusión de las sustancias y del lactato que las células adyacentes producen.

Bajo concentraciones de colágeno mayores estas células necrosadas comienzan a aparecer en las zonas interiores de los clústers formados [2] [8], ya que en esta zona más poblada la cantidad de oxígeno se reducirá de forma más rápida que en las zonas exteriores al clúster, zonas a las que solo las células que forman la corteza exterior del clúster tienen acceso.

Además, al reducir el oxígeno disponible desde el inicio en el entorno, las células son forzadas a entrar rápidamente en un ciclo anaeróbico para mantener la producción de ATP que les permite realizar sus funciones vitales y reproducirse. Al entrar en este ciclo, estas comienzan a acumular lactato en su entorno, que al final desemboca en la necrosis de alguna de las células.

Esto genera que, en el instante final de la simulación, el número de células en el dominio sea menor que en ensayos anteriores, algunas de las cuales se obtienen en estado de necrosis.

Además, se observa que las células en estado de necrosis tienden en general a encontrarse en la zona más interior de los clústers, en los casos de densidades en los que estos son diferenciables, ya que el oxígeno en estas zonas se consume más rápido y las células situadas en el exterior de los clústers tienen también acceso al oxígeno de zonas exteriores.

Por último, a mayor número de células iniciales, el oxígeno del entorno se reduce a un ritmo mayor, derivando en un mayor número de células en el instante final de la simulación, aunque también a un mayor número de células necrosadas.

3. Estudio de concentraciones

La densidad final para cada una de las sustancias estudiadas en apartados anteriores varía en cada uno de los ensayos, dependiendo, principalmente, del número de células que forman los clústers, ya que, en función de este valor, el consumo será mayor o menor en cada una de las localizaciones del entorno y, por tanto, la concentración final de cada sustancia variará de forma opuesta.

Otro factor para tener en cuenta incluso más importante es la concentración inicial de oxígeno que se introduce al modelo a la hora de comenzar la simulación ya que de ello depende que las células entren en ciclo anaeróbico y comiencen a consumir únicamente glucosa, lo cual supone una reducción en la cantidad de glucosa mucho más rápida que la que se daría en un caso en el que el oxígeno fuese predominante.

Para ello se utiliza un algoritmo creado para recuperar la información referente a la concentración de cada una de las sustancias y su localización en el entorno al final de la simulación. Con estos datos es posible generar, con la ayuda del algoritmo mencionado, un mapa de concentraciones a lo largo de cualquier plano entorno, aunque en este caso se escoge el plano medio para obtener las imágenes ya que es el plano donde se colocan inicialmente las células. De esta forma es posible ver los resultados en un plano significativo, que reproduzca los resultados de forma fiel en función al tamaño de los clústers.

El algoritmo que se utiliza en este caso se detalla en el Anexo 7.4. Este algoritmo funciona de forma que, una vez realizada la simulación y con todos los archivos que PhysiCell devuelve, recupera los datos referentes a las concentraciones de la sustancia indicada junto con las coordenadas dentro del entorno de cada valor. Se extrae el plano medio del entorno, ya que la posición inicial de todas las células introducidas para ver estos resultados se localiza en este plano, de forma que se puede ver un resultado más claro que si se escogiese cualquier otro plano.

Se estudian distintos planos de la iteración final para cada simulación, comenzando los ensayos con 5 células iniciales repartidas por el plano central del dominio (0, 0, 0). Uno de los planos estudiados en todas las simulaciones es este, el central, debido a que las células parten de este y, por tanto, habrá una mayor cantidad de células en este plano o adyacentes, pudiéndose ver con mayor claridad el efecto de estas sobre las sustancias.

En este apartado, se estudian todas las sustancias a excepción de la concentración de colágeno, la cual no tiene mucho sentido considerar, ya que, como se ha indicado al comienzo de este trabajo, esta concentración se considera constante y no decae debido a la actividad de las células, por lo que al final de la simulación esta concentración no habrá variado.

3.1. Oxígeno

La concentración inicial del oxígeno a lo largo de este trabajo depende de la situación que se quiera forzar en cuanto a las células presentes.

Se introduce un valor de densidad en condiciones normales, 160 mm Hg (21% O_2), en condiciones por debajo del límite de saturación de oxígeno, 38.0 mm Hg (5% O_2), y, para estudiar la producción de lactato por parte de las células, 17.0 mm Hg, cercana al valor límite de hipoxia. El consumo de oxígeno debería ser menor en este último caso ya que, una vez la densidad de esta sustancia pase el límite de los 15 mm Hg, las células reducen su consumo de oxígeno, reduciéndose la cantidad de esta sustancia casi exclusivamente por su difusión en el medio. Además, únicamente consumirán glucosa debido a su ciclo anaeróbico.

Los resultados obtenidos para el caso en el que la densidad inicial de oxígeno es de 17.0 mm Hg se muestran en la Figura 16.

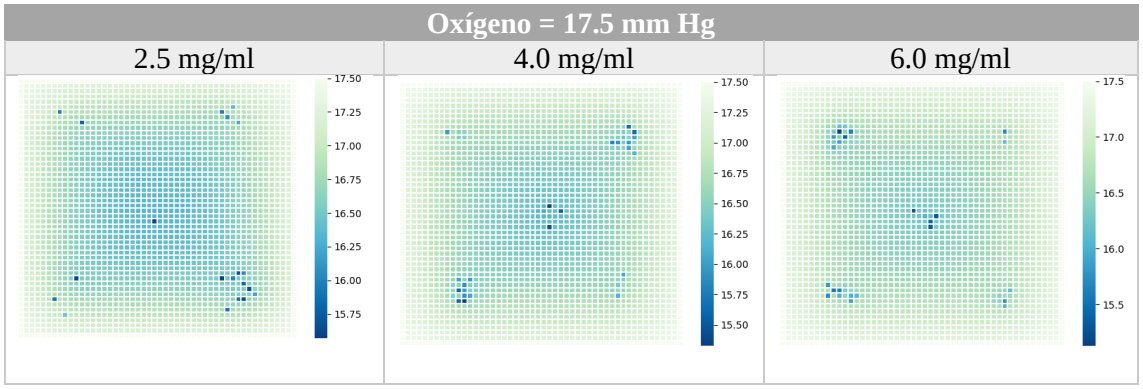

Figura 16. Resultados obtenidos para concentración inicial de oxígeno de 38.0 mm Hg.

Por otro lado, los resultados obtenidos para el segundo caso, en el que la densidad inicial de oxígeno es de 38.0 mm Hg, en el que las células consumen oxígeno durante los 5 días que dura la simulación, son los que se muestran en la Figura 17.

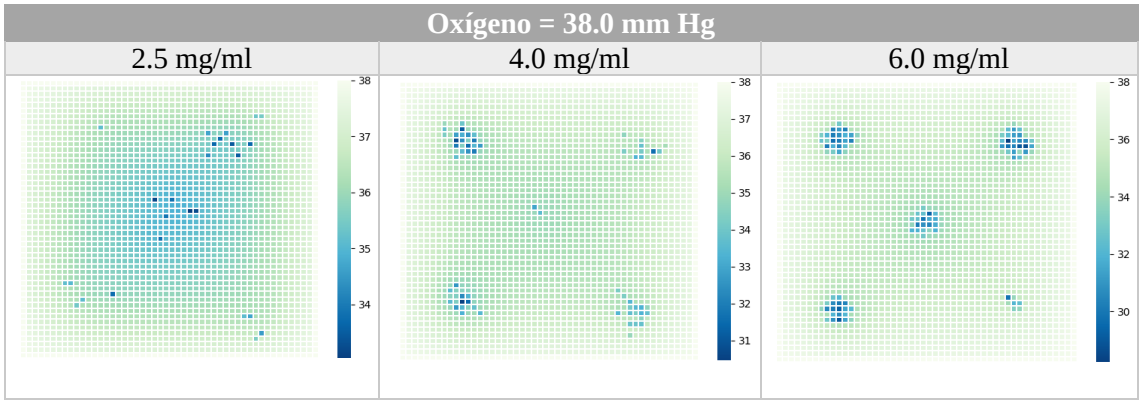

Figura 17. Resultados obtenidos para concentración inicial de oxígeno de 38.0 mm Hg.

Se observa que, a medida que la densidad de la matriz aumenta y, por tanto, la clusterización de las células, las áreas donde estos clústers se forman coinciden con las zonas en las que la disminución de oxígeno es mayor. En simulaciones en las que las células están más repartidas por el dominio, el gasto de oxígeno está más repartido en el área del plano que se estudia.

Por último, se muestran los resultados obtenidos para una concentración de oxígeno de 160.0 mm Hg, correspondiente a 21% O_2. Esta cantidad se corresponde con el estado de normoxia del sistema.

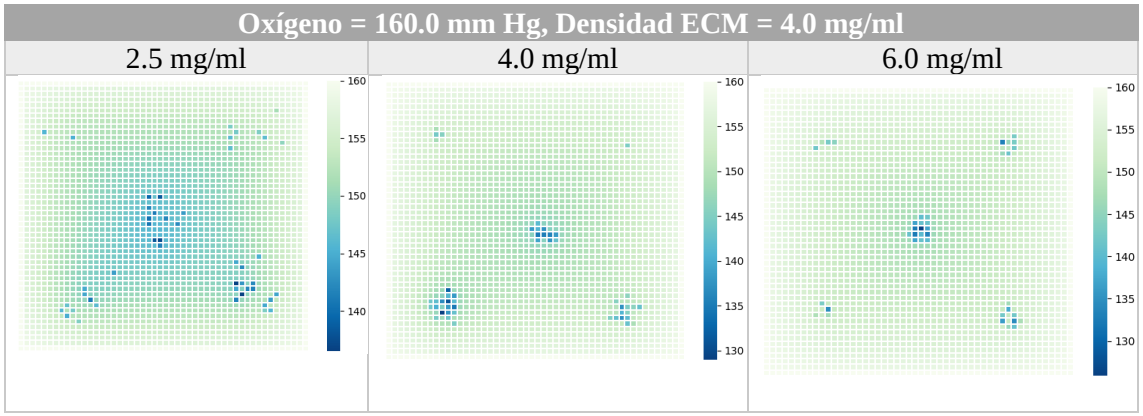

Figura 18. Resultados obtenidos para concentración inicial de oxígeno de 160.0 mm Hg.

Tras mostrar los resultados para los tres casos simulados, comparativamente se puede observar que el gradiente de la concentración de oxígeno aumenta con la densidad de colágeno de la matriz. Esto se debe a concentrar un mayor número de células en un volumen más reducido.

3.2. Glucosa

Respecto a los resultados obtenidos para la glucosa, únicamente se estudian dos concentraciones iniciales de oxígeno. Esto es debido a que, en lo que refiere al consumo de glucosa, no hay diferencia entre una concentración inferior o superior al límite de saturación de oxígeno (38.0 mm Hg) por lo que en ambos casos se obtienen resultados similares. Los casos mostrados son para densidades iniciales de oxígeno de 17.0 y 160.0 mm Hg.

Los resultados para el primer caso (concentración de oxígeno de 17.0 mm Hg) se muestran en la Figura 19. En ellos se puede ver que, a medida que aumenta la densidad de colágeno de la matriz, las zonas en las que la cantidad de glucosa es menor se concentran más, siendo además mayor el gradiente.

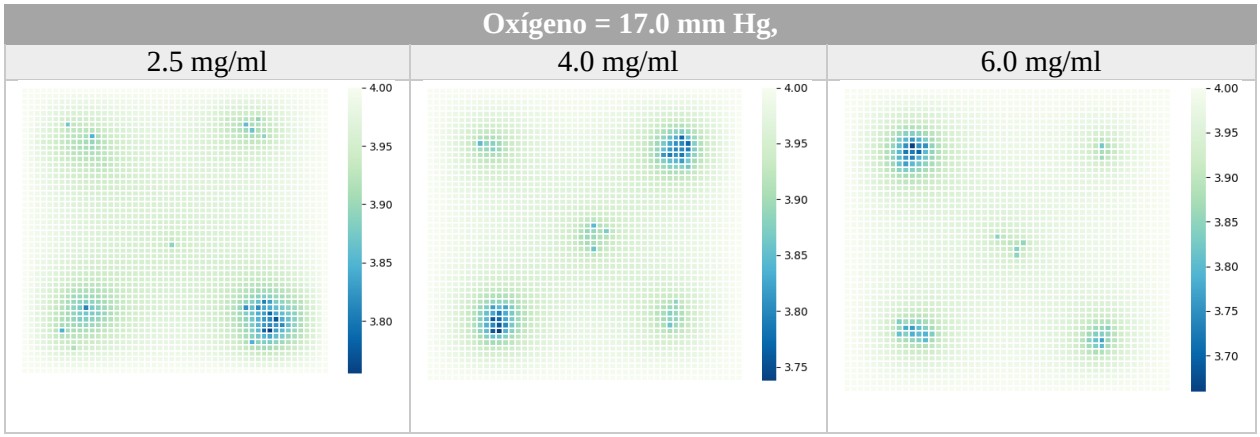

Figura 19. Resultados obtenidos con una concentración inicial de oxígeno de 17.0 mm Hg.

El segundo caso es el correspondiente a 160.0 mm Hg como densidad inicial de oxígeno, donde el consumo de glucosa se mantiene constante en todo el tiempo simulado. El gradiente de glucosa es mayor que en el caso anterior debido a que se consiguen un mayor número de células vivas al final de la simulación, que son las que producen la reducción en la cantidad de glucosa. Los resultados son los que aparecen en la Figura 20.

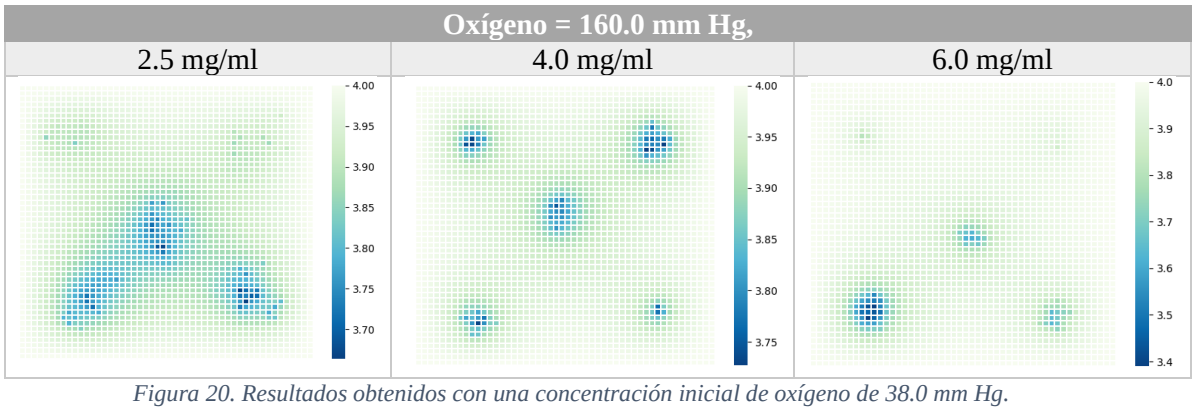

Figura 20. Resultados obtenidos con una concentración inicial de oxígeno de 38.0 mm Hg.

En este apartado se estudian también los resultados obtenidos para una concentración de glucosa baja. Según [3] la cantidad mínima de glucosa para que las células sean capaces de producir el ATP necesario está en torno a 1.0 mg/ml. En la adaptación que se ha realizado en PhysiCell este dato no se tiene en cuenta a la hora de producir ATP, sino que las células producen la cantidad de esta sustancia que sea posible y esta se compara con el valor mínimo de ATP que la célula necesita para realizar sus funciones. Es por ello por lo que el consumo de glucosa apenas varía en función de su cantidad inicial, las diferencias observables en ellas se deben a la estocasticidad del modelo y a la forma en la que proliferan las células en cada simulación.

Las distribuciones de la Figura 21 se han obtenido para una concentración de colágeno en la matriz de 4.0 mg/ml.

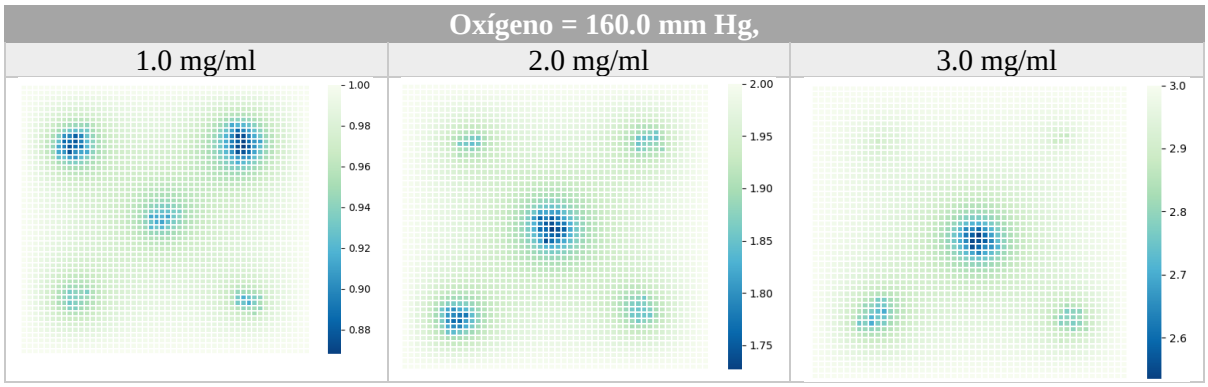

Figura 21. Densidad de oxígeno baja y alto contenido en glucosa.

3.3. Lactato

El caso de la concentración del lactato es algo distinto a las otras dos sustancias vistas anteriormente. Este comienza con una concentración inicial nula y, por tanto, en lugar de disminuir a lo largo de la simulación, se acumula en los volúmenes ocupados por los clústers celulares.

La concentración de esta sustancia únicamente comienza a crecer cuando cada una de las células entra en ciclo anaeróbico por el déficit de oxígeno disponible. Es por ello por lo que, para esta sustancia, como se ha indicado en el Apartado 2.3.1, se comprueba en cada instante la densidad de oxígeno disponible para cada una de las células presentes en el volumen de dominio. Si esta es inferior al valor límite, la célula comienza a producir lactato con la ratio explicada en el Apartado 2.3.1.

Los resultados obtenidos para la densidad de lactato al final de la simulación son los mostrados en la Figura 22.

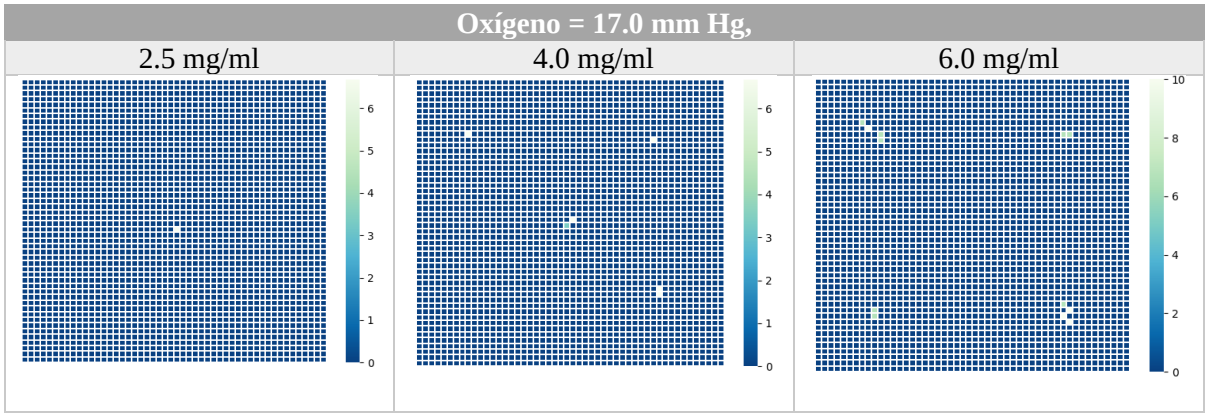

Figura 22. Densidad de oxígeno baja y alto contenido en glucosa.

Algo similar a lo que ocurre con las sustancias anteriores ocurre con el lactato. Este se concentra en las regiones del plano en las que se posicionan los clústers a medida que la densidad de colágeno aumenta, ya que en esas zonas es donde mayor densidad de población celular hay. Además, por esta misma razón, se observa que la producción de lactato es mayor conforme el colágeno en la matriz aumenta. Así, en el caso de una matriz con densidad de colágeno de 2.5 mg/ml el lactato generado es muy bajo y se concentra en el centro del plano, zona donde se sitúa uno de los clústers.

A medida que la densidad aumenta, se distinguen más zonas en las que esta sustancia se acumula, siempre, como se ha dicho anteriormente, en las zonas más centrales donde se sitúa cada uno de los clústers celulares.

4. Conclusiones

Los agentes vistos a lo largo de este trabajo (matriz extracelular, oxígeno, glucosa y lactato), tienen un papel fundamental en la proliferación y la necrosis celular. Como conclusiones extraídas de los resultados obtenidos:

1. La densidad de oxígeno y glucosa presentes disponibles para cada célula del modelo son los dos parámetros que determinan la cantidad de ATP que la célula es capaz de producir [9]. Esta producción es comparada con el valor mínimo de producción de ATP necesario para determinar si las células son capaces de proliferar [3] [10], viendo en los resultados que, tal como indica la Ecuación (*9*), el agente predominante en esta reacción es el oxígeno.

2. La densidad de colágeno de la matriz extracelular afecta a la capacidad migratoria de las células y, por tanto, a la formación de clústers [6], siendo estos más evidentes conforme este parámetro aumenta. Esto se debe a las fuerzas que la matriz ejerce sobre las células, Ecuación (*5*), y que es computada durante el modelo en función del valor de densidad inicial de colágeno introducido [11].

3. El lactato es una sustancia que inhibe el crecimiento y la proliferación celular, acidificando el medio y haciendo difícil la supervivencia de las células si este llega a una cantidad determinada [3].

4. Se han aplicado distintas aproximaciones vistas en los artículos de la bibliografía para cada uno de los agentes [3] [6] mediante funciones programadas en PhysiCell. Los resultados obtenidos mediante estas aproximaciones son coherentes con los mostrados en estos artículos, por lo que estas aproximaciones funcionan de forma satisfactoria.

5. Como líneas futuras de este trabajo, se podría implementar alguna sustancia más vista en los artículos, como los desechos que se generan tras la muerte celular [3] [8]. Otra opción podría ser la obtención de resultados estadísticos como la variación de velocidades o trayectorias bajo las distintas condiciones vistas, así como la forma de los clústers formados, que puede variar en función de la composición de la matriz. Además, a lo largo de todo el trabajo se ha supuesto que la glicolisis celular únicamente se da en el ciclo anaeróbico, sin embargo, el efecto Warburg, el cual determina la glicólisis en las células cancerígenas, indica que ésta se puede dar incluso en presencia de oxígeno.

6. Referencias

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts y P. Walter, «Molecular Biology of the Cell. 5th ed. New York: Garland Science, Taylor & Francis Group. 9780815341055,» 2007.

[2] M. Antonopoulos, D. Dionysiou, G. Stamatakos y N. Uzunoglu, «Three-dimensional tumor growth in timevarying chemical fields: a modeling framework and theoretical study. BMC Bioinformatics 20:442 https://doi.org/10.1186/s12859-019-2997-9,» 2019.

[3] N. Jagiella, B. Müller, M. Müller, I. E. Vignon-Clementel y D. Drasdo, «Inferring Growth Control Mechanisms in Growing Multi-cellular Spheroids of NSCLC Cells from Spatial-Temporal Image Data. PLoS Comput Biol 12(2): e1004412. doi:10.1371/journal.pcbi.1004412,» 2016.

[4] Y. Kim, H. Kang y S. Lawler, «The role of the miR-451-AMPK signaling pathway in regulation of cell migration and proliferation in glioblastoma.Mathematical Models of Tumor-Immune System Dynamics, Springer Proceeding in Mathematics & Stadistics 107. DOI 10.1007/978-1-4939-1793-8_6».

[5] A. Ghaffarizadeh, R. Heiland, S. H. Friedman, S. M. Mumenthaler y P. Macklin, «PhysiCell: an Open Source Physics-Based Cell Simulator for 3-D Multicellular Systems, PLoS Comput. Biol. 14(2): e1005991. DOI: 10.1371/journal.pcbi.1005991.,» 2018.

[6] I. Goncalves y J. M. García-Aznar, «Extracellular matrix density regulates the formation of tumour spheroids through cell migration. PLoS Comput Biol17(2): e1008764. https://doi.org/ 10.1371/journal.pcbi.1008764,» 2021.

[7] T. Alabduladhem y B. Bordoni, «Physiology, Krebs Cycle. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2022 Jan-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK556032/,» 2021.

[8] Y. Jiang, J. Pjesivac-Grbovic, C. Cantrell y J. P. Freyer, «A Multiscale Model for Avascular Tumor Growth. doi: 10.1529/biophysj.105.060640,» 2005.

[9] M. J. Piotrowska y S. D. Angus, «A quantitative cellular automaton model of in vitro multicellular spheroid tumour growth. Journal of Theoretical Biology 258 165–178,» 2009.

[10] F. Cleri, «Agent-based model of multicellular tumor spheroid evolution including cell metabolism. Eur. Phys. J. E (2019) 42: 112. DOI 10.1140/epje/i2019-11878-7,» 2019.

[11] J. Casciari, S. Sotirchos y R. Sutherland, «Variations in tumor cell growth rates and metabolism with oxygen concentration, glucose concentration, and extracellular pH. https://doi.org/10.1002/jcp.1041510220».

7. Anexos

7.1. Anexo 1. Fichero input XML

En este anexo se muestra un ejemplo de fichero inicial, con unos valores estándar para cada una de las variables. A lo largo de este trabajo los parámetros que se varían son los referentes a los agentes.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
/*
###############################################################################
# If you use PhysiCell in your project, please cite PhysiCell and the version #
# number, such as below:                                                      #
#                                                                             #
# We implemented and solved the model using PhysiCell (Version x.y.z) [1].    #
#                                                                             #
# [1] A Ghaffarizadeh, R Heiland, SH Friedman, SM Mumenthaler, and P Macklin, #
#     PhysiCell: an Open Source Physics-Based Cell Simulator for Multicellu-  #
#     lar Systems, PLoS Comput. Biol. 14(2): e1005991, 2018                   #
#     DOI: 10.1371/journal.pcbi.1005991                                       #
#                                                                             #
# See VERSION.txt or call get_PhysiCell_version() to get the current version  #
#     x.y.z. Call display_citations() to get detailed information on all cite-#
#     able software used in your PhysiCell application.                       #
#                                                                             #
# Because PhysiCell extensively uses BioFVM, we suggest you also cite BioFVM  #
#     as below:                                                               #
#                                                                             #
# We implemented and solved the model using PhysiCell (Version x.y.z) [1],    #
# with BioFVM [2] to solve the transport equations.                           #
#                                                                             #
# [1] A Ghaffarizadeh, R Heiland, SH Friedman, SM Mumenthaler, and P Macklin, #
#     PhysiCell: an Open Source Physics-Based Cell Simulator for Multicellu-  #
#     lar Systems, PLoS Comput. Biol. 14(2): e1005991, 2018                   #
#     DOI: 10.1371/journal.pcbi.1005991                                       #
#                                                                             #
# [2] A Ghaffarizadeh, SH Friedman, and P Macklin, BioFVM: an efficient para- #
#     llelized diffusive transport solver for 3-D biological simulations,     #
#     Bioinformatics 32(8): 1256-8, 2016. DOI: 10.1093/bioinformatics/btv730  #
#                                                                             #
###############################################################################
#                                                                             #
# BSD 3-Clause License (see https://opensource.org/licenses/BSD-3-Clause)     #
#                                                                             #
# Copyright (c) 2015-2018, Paul Macklin and the PhysiCell Project             #
# All rights reserved.                                                        #
#                                                                             #
# Redistribution and use in source and binary forms, with or without          #
# modification, are permitted provided that the following conditions are met: #
#                                                                             #
# 1. Redistributions of source code must retain the above copyright notice,   #
# this list of conditions and the following disclaimer.                       #
#                                                                             #
# 2. Redistributions in binary form must reproduce the above copyright        #
# notice, this list of conditions and the following disclaimer in the         #
# documentation and/or other materials provided with the distribution.        #
#                                                                             #
# 3. Neither the name of the copyright holder nor the names of its            #
# contributors may be used to endorse or promote products derived from this   #
# software without specific prior written permission.                         #
#                                                                             #
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" #
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE   #
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE  #
# ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE   #
# LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR         #
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF        #
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS    #
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN     #
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)     #
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE  #
# POSSIBILITY OF SUCH DAMAGE.                                                 #
#                                                                             #
###############################################################################
*/
-->

<!--
<user_details />
-->

<PhysiCell_settings version="devel-version">
        <domain>
                <x_min>-500</x_min>
                <x_max>500</x_max>
                <y_min>-500</y_min>
                <y_max>500</y_max>
                <z_min>-500</z_min>
                <z_max>500</z_max>
                <dx>20</dx>
                <dy>20</dy>
                <dz>20</dz>
                <use_2D>false</use_2D>
        </domain>

```

```
89.         <overall>
90.                 <max_time units="min">7200</max_time> <!-- 5 days * 24 h * 60 min -->
91.                 <time_units>min</time_units>
92.                 <space_units>micron</space_units>
93.
94.                 <dt_diffusion units="min">0.01</dt_diffusion>
95.                 <dt_mechanics units="min">0.1</dt_mechanics>
96.                 <dt_phenotype units="min">6</dt_phenotype>
97.         </overall>
98.
99.         <parallel>
100.                 <omp_num_threads>6</omp_num_threads>
101.         </parallel>
102.
103.         <save>
104.                 <folder>output</folder> <!-- use . for root -->
105.
106.                 <full_data>
107.                         <interval units="min">60</interval>
108.                         <enable>true</enable>
109.                 </full_data>
110.
111.                 <SVG>
112.                         <interval units="min">60</interval>
113.                         <enable>true</enable>
114.                 </SVG>
115.
116.                 <legacy_data>
117.                         <enable>false</enable>
118.                 </legacy_data>
119.         </save>
120.
121.         <options>
122.                 <legacy_random_points_on_sphere_in_divide>false</legacy_random_points_on_sphere_in_divide>
123.         </options>
124.
125.         <microenvironment_setup>
126.                 <variable name="oxygen" units="mmHg" ID="0">
127.                         <physical_parameter_set>
128.                                 <diffusion_coefficient units="micron^2/min">100000.0</diffusion_coefficient>
129.                                 <decay_rate units="1/min">.1</decay_rate>
130.                         </physical_parameter_set>
131.                         <initial_condition units="mmHg">38.0</initial_condition>
132.                         <Dirichlet_boundary_condition units="mmHg" enabled="true">38.0</Dirichlet_boundary_condition>
133.                 </variable>
134.
135.                 <variable name="ECM" units="mg/mL" ID="1">
136.                 <physical_parameter_set>
137.                         <diffusion_coefficient units="micron^2/min">0.0</diffusion_coefficient>
138.                         <decay_rate units="1/min">0.0</decay_rate>
139.                 </physical_parameter_set>
140.                 <initial_condition units="mg/mL">2.5</initial_condition>
141.                 <Dirichlet_boundary_condition units="mg/mL" enabled="true">2.5</Dirichlet_boundary_condition>
142.                 </variable>
143.                 <variable name="glucose" units="mg/mL" ID="2">
144.                         <physical_parameter_set>
145.                                 <diffusion_coefficient units="micron^2/min">8000.0</diffusion_coefficient>
146.                                 <decay_rate units="1/min">0.00001</decay_rate>
147.                         </physical_parameter_set>
148.                         <initial_condition units="mg/mL">3.0</initial_condition>
149.                         <Dirichlet_boundary_condition units="mg/mL" enabled="true">3.0</Dirichlet_boundary_condition>
150.                 </variable>
151.                 <variable name="lactate" units="mg/mL" ID="3">
152.                         <physical_parameter_set>
153.                                 <diffusion_coefficient units="micron^2/min">0.0</diffusion_coefficient>
154.                                 <decay_rate units="1/min">0.0</decay_rate>
155.                         </physical_parameter_set>
156.                         <initial_condition units="mg/mL">0.0</initial_condition>
157.                         <Dirichlet_boundary_condition units="mg/mL" enabled="true">0.0</Dirichlet_boundary_condition>
158.                 </variable>
159.
160.                 <options>
161.                         <calculate_gradients>true</calculate_gradients>
162.                         <track_internalized_substrates_in_each_agent>true</track_internalized_substrates_in_each_agent>
163.                         <!-- not yet supported -->
164.                         <initial_condition type="matlab" enabled="false">
165.                                 <filename>./config/initial.mat</filename>
166.                         </initial_condition>
167.                         <!-- not yet supported -->
168.                         <dirichlet_nodes type="matlab" enabled="false">
169.                                 <filename>./config/dirichlet.mat</filename>
170.                         </dirichlet_nodes>
171.                 </options>
172.         </microenvironment_setup>
173.
174.         <user_parameters>
175.                 <random_seed type="int" units="dimensionless">0</random_seed>
176.                 <!-- example parameters from the template -->
177.
178.                 <!-- motile cell type parameters -->
179.                 <motile_cell_persistence_time type="double" units="min">15</motile_cell_persistence_time>
180.                 <motile_cell_migration_speed type="double" units="micron/min">0.25</motile_cell_migration_speed>
181.                 <motile_cell_relative_adhesion type="double" units="dimensionless">0.05</motile_cell_relative_adhesion>
182.                 <motile_cell_apoptosis_rate type="double" units="1/min">0.0</motile_cell_apoptosis_rate>
183.                 <motile_cell_relative_cycle_entry_rate type="double" units="dimensionless">0.1</motile_cell_relative_cycle
184.
185.         </user_parameters>
186.
187.
188. </PhysiCell_settings>
```

Figura 23. Fichero de introducción de datos XML.

7.2. Anexo 2. Introducción de datos iniciales.

Para esta funcionalidad se necesitan tres ficheros para que la introducción de datos sea rápida y sea posible lanzar varios cálculos simultáneamente en lote. Esto es posible mediante la herramienta HTCondor, la cual permite correr el programa varias veces con datos iniciales distintos. El primer fichero necesario es propiamente el de HTCondor, fichero '.sub', con el que se le pasan a la función la cadena de valores que se desea introducir al modelo en cada una de las simulaciones. Un ejemplo de este fichero es el que aparece en la Figura 24.

```
1. # Basic definitions
2. Universe     = vanilla
3. Executable   = run_batch_simulation.sh
4. arguments        = --odens $(ODENS) --cdens $(CDENS) --gdens $(GDENS)
5.
6. # Request memory and CPUs
7. Request_cpus = 10
8. Request_memory = 200M
9.
10. # Select the files that will be needed for the simulation
11. transfer_input_files = run_batch_simulation.sh, config/PhysiCell_settings.xml, data_setup.py, projecti
12.
13. # Transfer outputs to the cluster
14. should_transfer_files = YES
15. WhenToTransferOutput = ON_EXIT_OR_EVICT
16. # Log and error files will be saved in a log folder (do not forget to create it)
17. Log    = log/project.log
18. Error  = log/project_o$REAL(ODENS,%05.2f)_c$REAL(CDENS,%05.2f)_g$REAL(GDENS,%05.2f).err
19. Output = log/project_o$REAL(ODENS,%05.2f)_c$REAL(CDENS,%05.2f)_g$REAL(GDENS,%05.2f).out
20. # Output files will be downloaded to the current directory as a compressed folder
21. transfer_output_files = results_o$REAL(ODENS,%05.2f)_c$REAL(CDENS,%05.2f)_g$REAL(GDENS,%05.2f).tar.bz
22.
23. # ODENS = O2 INITIAL DENSITY
24. # CDENS = COLLAGEN INITIAL DENSITY
25. # GDENS = GLUCOSE INITIAL DENSITY
26.
27. queue ODENS, CDENS, GDENS from (
28. 16.5, 2.5, 4.0
29. 16.5, 4.0, 4.0
30. 16.5, 6.0, 4.0
31. )
```

Figura 24. Fichero HTCondor de ejemplo.

En este fichero se hace llamada a todos los archivos necesarios para realizar la simulación, así como al archivo comprimido que esta devuelve una vez finalizada. Entre ellos, el que recoge cada una de las filas de valores que aparecen en el archivo '.sub' y las pasa una a una al fichero Python que modifica finalmente el archivo XML es un archivo escrito en Bash '.sh'.

Este archivo es esencialmente de preproceso, crea las carpetas necesarias para almacenar los resultados y borra archivos temporales que, una vez finalizada la simulación no son necesarios.

Además, nombra cada una de las carpetas de resultados en función de los valores iniciales introducidos, para tener algo más de organización. El código de este fichero se muestra en la Figura 25.

```
#!/bin/bash
# Put the config file in the location where project3D expects to find it.

mkdir -p config
mv PhysiCell_settings.xml config/
# Create the results folder for project3D.
mkdir -p output


while [ "$1" != "" ]; do
    # All parameters appearing in python and HTCondor scripts
    case $1 in
        --odens)
            odens=$2
            shift 2
        ;;

        --cdens)
            cdens=$2
            shift 2
        ;;

        --gdens)
            gdens=$2
            shift 2
        ;;

        --ldens)
            ldens=$2
            shift 2
        ;;

        *)
            echo "command not recognized: $1, $2"
            exit
           # unknown option
        ;;
    esac
done

suffix=$(printf "o%05.2f_c%05.2f_g%05.2f" ${odens} ${cdens} ${gdens})
results_path=results_${suffix}

python data_setup.py ${odens} ${cdens} ${gdens} ${ldens}


# (change project3D to the name of the PhysiCell project you want to run)
./project3D

mkdir -p ${results_path}

cp -r *.txt ${results_path}
cp config/PhysiCell_settings.xml ${results_path}
cp -r output/. ${results_path}
rm -r output

# Clean and compress results to improve transfer speeds.
tar -cjf ${results_path}.tar.bz2 ${results_path}
mv ${results_path}.tar.bz2 results/

rm -r ${results_path}
```

Figura 25. Fichero Bash de ejemplo.

Por último, el fichero Python que accede al XML de entrada y modifica los datos en función de los pasados con lo anterior descrito se muestra en la Figura 26.

```
1. import sys                                    # To read the terminal inputs
2. from xml.etree import ElementTree as et       # To change the XML file
3.
4. # Variable definition
5. folder_name = "config/"
6. file_name = "PhysiCell_settings.xml"
7.
8.
9. oxygen_node = "microenvironment_setup/variable[@name='oxygen']"
10. collagen_node = "microenvironment_setup/variable[@name='ECM']"
11. glucose_node = "microenvironment_setup/variable[@name='glucose']"
12. #lactate_node = "microenvironment_setup/variable[@name='lactate']"
13.
14. oxygen_dens = oxygen_node + "/initial_condition"
15. collagen_dens = collagen_node + "/initial_condition"
16. glucose_dens = glucose_node + "/initial_condition"
17. #lactate_dens = lactate_node + "/initial_condition"
18.
19. oxygen_dirichlet = oxygen_node + "/Dirichlet_boundary_condition"
20. glucose_dirichlet = glucose_node + "/Dirichlet_boundary_condition"
21. collagen_dirichlet = collagen_node + "/Dirichlet_boundary_condition"
22. #lactate_dirichlet = lactate_node + "/Dirichlet_boundary_condition"
23.
24. # Read and update file
25. tree = et.parse(folder_name + file_name)
26.
27. tree.find(oxygen_dens).text = str(sys.argv[1])
28. tree.write(folder_name + file_name)
29.
30. tree.find(oxygen_dirichlet).text = str(sys.argv[1])
31. tree.write(folder_name + file_name)
32.
33. tree.find(collagen_dens).text = str(sys.argv[2])
34. tree.write(folder_name + file_name)
35.
36. tree.find(collagen_dirichlet).text = str(sys.argv[2])
37. tree.write(folder_name + file_name)
38.
39. tree.find(glucose_dens).text = str(sys.argv[3])
40. tree.write(folder_name + file_name)
41.
42. tree.find(glucose_dirichlet).text = str(sys.argv[3])
43. tree.write(folder_name + file_name)
```

Figura 26. Fichero de modificación XML Python.

7.3. Anexo 3. Visualización de resultados celulares.

Es necesaria también la introducción de un método con el que poder recuperar los datos que PhysiCell devuelve y convertirlos en las gráficas 3D que se muestran a lo largo de este trabajo. Este método convierte las células en esferas de diámetro fijo y las colorea en función del clúster al que pertenecen o de negro si estas están en estado de necrosis. Se ha utilizado como base un fichero ya programado y se ha adaptado para mostrar los resultados en una matriz 3D. El código de este método se muestra en las Figura 30.

```
# Importing libraries
from datetime import time
#from os import times, listdir, startfile, remove
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import mpl_toolkits.mplot3d.art3d as art3d
import pandas as pd
import seaborn as sns
from scipy import io as sio
from pathlib import Path
from scipy.spatial import distance as dist
from sklearn.cluster import DBSCAN
from matplotlib.patches import Ellipse
import sys
import time

start = time.time()

#%matplotlib inline
sns.set_context('notebook')
sns.set_palette(['#0071BC', '#d95319', '#efb320'])



##### METHODS USED HERE #####
def build_path_name(timestep, folder_name, data_type):
    """Returns a Path object with the adequate PhysiCell output name format.

   Uses the standard PhysiCell output filename structure,
   (output{data_type}_{time_id}.mat)

   Parameters
   ----------
   timestep : int
       The time point at which the output was recorded
   folder_path: Path
       The path to the folder where the output (.mat, .xml) files are stored
   data_type: string
       The type of data to be retrieved (cells for cell-based data and
       microenvironment for the continuum variables)
   """

    # All possible file types written by PhysiCell
    data_type_name = {
        'cells': 'cells_physicell'
    }
    # Variable definition
    file_name = data_type_name[data_type]

    time_str = str(timestep).zfill(8)

    file_name = 'output{}_{}.mat'.format(time_str, file_name)



    # Build path name
    path_name = folder_name / file_name

    return path_name
```

Figura 27. Fichero de visualización de resultados celulares (I).

```
62.
63.
64. def get_cell_data(timestep, folder_name, variables='all'):
65.     """Returns a dictionary with the cell output data for the given variables.
66.
67.    Parameters
68.    ----------
69.    timestep : int
70.        The time point at which the output was recorded
71.    folder_path: Path
72.        The path to the folder where the output (.mat, .xml) files are stored
73.    variables : list
74.        The variables to be extracted from the output files. If variables
75.        are not defined, all the available outputs will be saved.
76.    """
77.
78.     # All possible output variables written by PhysiCell
79.     data_labels = [
80.         'ID',
81.         'position_x', 'position_y', 'position_z',
82.         'total_volume',
83.         'cell_type',
84.         'cycle_model', 'current_phase', 'elapsed_time_in_phase',
85.         'nuclear_volume', 'cytoplasmic_volume',
86.         'fluid_fraction', 'calcified_fraction',
87.         'orientation_x', 'orientation_y', 'orientation_z',
88.         'polarity',
89.         'migration_speed',
90.         'motility_vector_x', 'motility_vector_y', 'motility_vector_z',
91.         'migration_bias',
92.         'motility_bias_direction_x', 'motility_bias_direction_y', 'motility_bias_direction_z',
93.         'persistence_time',
94.         'motility_reserved'
95.     ]
96.
97.     # Variable definition
98.     cells = {}
99.     file_type = 'cells'
100.
101.     if variables == 'all':
102.         variables = data_labels
103.
104.     # Build path name
105.     path_name = build_path_name(timestep, folder_name, file_type)
106.
107.
108.     # Read output file
109.     cell_data = sio.loadmat(path_name)['cells']
110.
111.     # Select and save the variables of interest
112.     variables_indexes = [data_labels.index(var) for var in variables]
113.
114.     for index, var, in zip(variables_indexes, variables):
115.         cells[var] = cell_data[index, :]
116.
117.     return cells
118.
119.
120. def cells_in_z_area_of_interest(cells, height_of_interest):
121.     """Selects cells in a defined area and returns them in a DataFrame.
122.
123.
124.    Parameters
125.    ---------
126.    cells : DataFrame
127.        The ax object to be stylized
128.    height_of_interest : int
129.        The threshold that defines which cells to select. Assumed to be
130.        equal for positive and negative z coordinates
131.    """
132.
133.     # Define conditions
134.     cells_above_neg_height_of_interest = cells['position_z'] > -height_of_interest
135.     cells_below_height_of_interest = cells['position_z'] < height_of_interest
136.     area_of_interest = cells_above_neg_height_of_interest & cells_below_height_of_interest
137.
138.     # Select cells
139.     cells_of_interest = cells[area_of_interest]
140.
141.     return cells_of_interest
142.
143.
144. def classify_cells_into_clusters(cells, DBSCAN_radius=18, DBSCAN_min_cells=3):
145.     """Classifies cells into clusters using the DBSCAN algorithm.
146.
147.    Uses the cells' spatial information (x and y coordinates) through sklearn's
148.    DBSCAN algorithm. More information can be found at:
149.    https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
150.    Returns the input DataFrame, with an additional column ('cluster') corrsponding to
151.    the ID of the cluster the cell belongs to. Outlier cells are given an ID of -1.
152.
153.    Parameters
154.    ----------
155.    cells : DataFrame
156.        The DataFrame with cell data to be classified
157.    DBSCAN_radius : int
158.        The radius to be considered by the DBSCAN algorithm
159.        (default is 18 [microns])
160.    DBSCAN_min_cells : int
161.        The minimum number of cells to be used by the DBSCAN algorithm
162.        (default is 3 cells)
163.    """
164.
165.     # Classify cells based on spatial information
166.     cells_positions = cells[['position_x', 'position_y']]
167.     dbscan_clusters = DBSCAN(eps=DBSCAN_radius, min_samples=DBSCAN_min_cells).fit(cells_positions)
168.
169.     # Update the cells DataFrame with the corresponding cluster labels
170.     cells['cluster'] = dbscan_clusters.labels_
171.
172.     return cells
173.
174. def classify_cells_into_death(cells):
175.     cell_cycles = cells['cycle_model']
176.
177.
178.     return cells
179.
180.
```

Figura 28. Fichero de visualización de resultados celulares (II).

```
def draw_cells_as_spheres(ax, cells, y_variable='position_y', z_variable='position_z', clusterized=False, cell_radius=8):
    """Draws cells as circles of a defined radius.

    Parameters
    ----------
    ax : ax
        The ax object to draw in.
    cells : DataFrame
        The DataFrame with the cells to be drawn.
    y_variable : string
        The coordinate to be represented in the y axis (y or z, default
        is the y coordinate)
    clusterized : Boolean
        Defines if cells should be colorized based on the cluster they
        belong to (default is False, all cells get the same color)
    cell_radius : int
        The radius of the cells (default is 8 [microns])
    """

    # When cells are classified into clusters, cells from the same cluster are plotted with the same color.
    # Independent cells are plotted as black.
    if clusterized:
        num_colors = len(cells['cluster'].unique())

        my_palette = sns.color_palette('Spectral', num_colors)
        my_palette.insert(len(my_palette), (0,0,0))

        for x_pos, y_pos, z_pos, color, model in zip(cells['position_x'], cells[y_variable], cells[z_variable], cells['cluster'], cells['cycle_model']):
            #cell_contour = plt.Circle((x_pos, y_pos),
            #                          cell_radius,
            #                          facecolor=my_palette[color], edgecolor='black')

            #ax.add_patch(cell_contour)
            #print(model)
            if model == 101.0 :
                ax.scatter(x_pos, y_pos, z_pos, s=cell_radius, color='white', edgecolors='black')
            else:
                ax.scatter(x_pos, y_pos, z_pos, s=cell_radius, color=my_palette[color], edgecolors='black')

    # When cells are not classified into clusters, all cells are plotted with the same color.
    else:
        for x_pos, y_pos, z_pos, model in zip(cells['position_x'], cells[y_variable], cells[z_variable], cells['cycle_model']):
            #cell_contour = plt.Circle((x_pos, y_pos),
            #                          cell_radius,
            #                          facecolor='C0', edgecolor='black')


            #ax.add_patch(cell_contour)
            if model == 101.0 :
                ax.scatter(x_pos, y_pos, z_pos, s=cell_radius, color='white', edgecolors='black')
            else:
                ax.scatter(x_pos, y_pos, z_pos, s=cell_radius, color='black', edgecolors='black')


def set_cell_view_style_axes(ax):
    """Sets axes style to predefined settings.

    Includes a light gray dashed grid and all spines set as black,
    with ticks pointing inwards.

    Parameters
    ---------
    ax : ax
        The ax object to be stylized.
    """

    # Grid
    ax.yaxis.grid(color='lightgray', linestyle='--', linewidth=0.5, zorder=-1)
    ax.xaxis.grid(color='lightgray', linestyle='--', linewidth=0.5, zorder=-1)
    ax.zaxis.grid(color='lightgray', linestyle='--', linewidth=0.5, zorder=-1)

    # Ticks
    ax.tick_params(axis="y", direction="in", right=True)
    ax.tick_params(axis="x", direction="in", top=True)
    ax.tick_params(axis="z", direction="in", top=True)

    # Spines
    for spine in ax.spines.values():
        spine.set_visible(True)
        spine.set_linewidth(1)
        spine.set_edgecolor('black')

############################

##### INITIAL VARIABLES DECLARATION HERE #####

variables = ['ID', 'position_x', 'position_y', 'position_z', 'cycle_model']
files = []

#if (timestep == ''):
timestep = int(sys.argv[1])

folder = str(sys.argv[2])

if(str(timestep) == ""):
        timestep = 100

if(folder == ""):
    folder = "output"

```

Figura 29. Fichero de visualización de resultados celulares (III).

```
base_folder_name = Path('./' + str(folder) + '/')  # Cluster Path
#base_folder_name = Path('D:/TFM/output/output_o2dens_100/')    # Local Path
cells_folder = base_folder_name / 'cells_plot/'
##############################


# Create fig object
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
fig.set_figwidth(10)
fig.set_figheight(5)

cells = get_cell_data(timestep, base_folder_name, variables)

# Se almacenan los datos seleccionados de las celulas de todo el volumen
cells_df = pd.DataFrame(cells, columns=variables)

# Se eliminan las celulas que no esten en el plano pasado al metodo
#cells_df = cells_in_z_area_of_interest(cells_df, z_height)

# Clusterization
cells_df = classify_cells_into_clusters(cells_df, DBSCAN_min_cells=3)

# Define ax to draw in
#ax = axes[timestep]

# Plot cells with an accurate representation of the assumed geometry
draw_cells_as_spheres(ax, cells_df, y_variable='position_y', z_variable='position_z', clusterized=True)

# Figure aesthetics
ax.set_ylim(-300,300)
ax.set_xlim(-300,300)
ax.set_zlim(-300,300)

# Use the standard axes style
set_cell_view_style_axes(ax)

#ax.set_title("Collagen concentration: {} [mg/mL]".format(concentrations[index]), y=1.05, fontsize=15)

ax.set_xlabel("X Position [$\mu$m]", labelpad=15, fontsize=15)
ax.set_ylabel("Y Position [$\mu$m]", labelpad=10, fontsize=15)
ax.set_zlabel("Z Position [$\mu$m]", labelpad=10, fontsize=15)
ax.set_title("Step number " + str(timestep), loc='center')

end = time.time()
plt.show()

print('Elapsed time (s): '+ str(round(end-start, 1)))
```

Figura 30. Fichero de visualización de resultados celulares (IV).

7.4. Anexo 4. Visualización de concentraciones.

Para la visualización de concentraciones se parte de un archivo previo el cual es adaptado para mostrar las concentraciones de las sustancias añadidas al modelo a lo largo de este trabajo. El fichero únicamente necesita como datos de entrada la carpeta en la que están almacenados los resultados de PhysiCell y la sustancia que se desea mostrar. El código de este archivo se muestra en la Figura 31.

```
1. import os
2. from pathlib import Path
3. import matplotlib.pyplot as plt
4. import numpy as np
5. import pandas as pd
6. import seaborn as sns
7. from os import listdir, remove
8. from sys import argv
9.
10. import phisypy as phypy
11. import imageio
12.
13.
14. FILE = argv[1]
15. SUBSTANCE = str(argv[2])
16.
17. BASE_FILE = Path(FILE)
18. #BASE_FILE = Path('./output/')
19.
20. # Takes the path given and runs the program with the files included in the folder
21. # BASE_FILE = Path(argv)
22. gif_name = 'output.gif' #BASE_FILE.name
23. counter = 0
24. ax = []
25. timestep = 120
26.
27. substance_data = phypy.get_me_data(timestep, SUBSTANCE, BASE_FILE)
28. #colorbar = fig.add_axes([.92, .15, .012, .7])
29.
30. sns.heatmap(substance_data, cmap="GnBu_r", linewidths=0.01, xticklabels=False, yticklabels=Fa
31. filename = 'step_{}.png'.format(timestep)
32. ax.append(filename)
33. plt.savefig(BASE_FILE / filename)
34. plt.show()
```

Figura 31. Fichero de visualización de concentraciones.

Donde en la variable ‘substance_data’ se almacena la información de cada una de las células en un vector. Se almacenan las posiciones en X, Y y Z, así como la concentración de cada una de las sustancias introducidas al modelo por el fichero XML.

```python
def get_me_data(timestep, substance, base_file, Z=200):
    """Returns an array with the substance concentrations at the middle plane of the domain.

    Parameters
    ----------
    timestep : int
        The time point at which the output was recorded.
    substance : string
        The substance to be quantified.
    base_file: Path
        The path to the folder where the output (.mat/.xml) files are stored.
    """

    # Create path
    if len(str(timestep)) == 1:
        timestep = '00' + str(timestep)
    elif len(str(timestep)) == 2:
        timestep = '0' + str(timestep)

    if str(timestep) == 'initial':
        xml_file = base_file / 'initial.xml'
        me_file = base_file / 'initial_microenvironment0.mat'
    elif str(timestep) == 'final':
        xml_file = base_file / 'final.xml'
        me_file = base_file / 'final_microenvironment0.mat'
    else:
        xml_file = base_file / 'output00000{}.xml'.format(timestep)
        me_file = base_file / 'output00000{}_microenvironment0.mat'.format(timestep)

    # Open XML file to get list of variables
    tree = ET.parse(xml_file)
    root = tree.getroot()
    me_node = root.find('microenvironment')
    me_node = me_node.find('domain')
    mesh_node = me_node.find('mesh')
    variables_node = me_node.find('variables')
    var_children = variables_node.findall('variable')
    variables = [var.get('name') for var in var_children]

    # Get x, y and z coordinates
    # X coordinates
    coord_str = mesh_node.find('x_coordinates').text
    delimiter = mesh_node.find('x_coordinates').get('delimiter')
    x_coords = np.array(coord_str.split(delimiter), dtype=np.float)
    # Y coordinates
    coord_str = mesh_node.find('y_coordinates').text
    delimiter = mesh_node.find('y_coordinates').get('delimiter')
    y_coords = np.array(coord_str.split(delimiter), dtype=np.float)
    # Z coordinates
    coord_str = mesh_node.find('z_coordinates').text
    delimiter = mesh_node.find('z_coordinates').get('delimiter')
    z_coords = np.array(coord_str.split(delimiter), dtype=np.float)
    z_middle_point = int(len(z_coords) / 2)

    # Define the shape of the output array
    data_shape = (len(x_coords), len(y_coords))

    # Load substance data
    me_data = sio.loadmat(me_file)['multiscale_microenvironment']

    # Select the data corresponding to the chosen substance
    substance_index = variables.index(substance)
    substance_data = me_data[substance_index + 4, me_data[2, :] == z_coords[z_middle_point]]

    # Reshape output array to match the x and y coordinates
    substance_data = np.reshape(substance_data, data_shape)

    return substance_data
```

Figura 32. Método para recuperar los datos de la simulación.

7.5. Anexo 5. Adición del efecto del colágeno.

En este anexo se muestra el código implementado para la adición del efecto de la densidad de colágeno en el modelo, donde se ve el valor de viscosidad dinámica seleccionado en función de la densidad de colágeno. Este se muestra en la Figura 33.

```
1. void drag_update_cell_velocity( Cell* pCell, Phenotype& phenotype, double dt ) {
2.     // sample ECM
3.     int ECM_density_index = microenvironment.find_density_index( "ECM" );
4.     double ECM_density = pCell->nearest_density_vector()[ECM_density_index];
5.     double dyn_viscosity;
6.     // get viscosity based on concentration
7.     if(ECM_density == 2.5) {
8.         dyn_viscosity = 7.96;
9.     }
10.     else if(ECM_density == 4.0){
11.         dyn_viscosity = 18.42;
12.     }
13.     else if(ECM_density == 6.0) {
14.         dyn_viscosity = 39.15;
15.     }
16.     // update the speed value
17.     pCell->phenotype.motility.migration_speed =
18.         locomotive_forces_generator();
19.     // update velocity
20.     standard_update_cell_velocity(pCell, phenotype, dt);
21.     // include the 1/vu (1/ECM density) term to consider friction
22.     pCell->velocity /= dyn_viscosity;
23.     return;
24. }
```

Figura 33. Fichero de visualización de resultados.

7.6. Anexo 6. Adición del efecto de la glucosa.

En este anexo se muestra el código implementado para la adición del efecto de la glucosa en el modelo y todo el proceso explicado anteriormente por el que se computa, junto con el oxígeno, la producción de ATP. Este se muestra en la Figura 34.

```
1. void get_proliferation_necrosis_rates(Cell* pCell, Phenotype& phenotype, double dt, int start_phase_index, int end_phase_index) {
2.
3.         int oxygen_substrate_index = pCell->get_microenvironment()->find_density_index("oxygen");
4.         int glucose_substrate_index = pCell->get_microenvironment()->find_density_index("glucose");
5.         int lactate_substrate_index = pCell->get_microenvironment()->find_density_index("lactate");
6.
7.         //COMBUSTION// OXYGEN:GLUCOSE RATIO -> 1G1:6O2
8.
9.                 // Regla de 3:
10.                //              20% O2 ---------- 0.28 mM
11.                //              5% O2  ---------- 0.07 mM (38 mmHg)
12.
13.                // Regla de 3 para obtener % a partir de los mmHg que tenemos (38 mmHg = 5% O2)
14.
15.         double O2_perc = (pCell->parameters.pO2 * 5) / 38;
16.         pCell->parameters.mMglucose = (pCell->parameters.pGlucose / 180) * 1000; // glucose molecular weight [mM]
17.
18.         pCell->parameters.mMo2 = (O2_perc * 0.28) / 20; // [mM]
19.
20.
21.         pCell->parameters.Vg_max = pCell->parameters.qg_max *
22.                 (1 - (1 - pCell->parameters.qg_min / pCell->parameters.qg_max) * pCell->parameters.mMglucose / (pCell->parameters.mMglucose + pCell->parameters.k_g_o));
23.         pCell->parameters.Vo_max = pCell->parameters.qo_max *
24.                 (1 - (1 - pCell->parameters.qo_min / pCell->parameters.qo_max) * pCell->parameters.mMo2 / (pCell->parameters.mMo2 + pCell->parameters.k_o_g));
25.
26.         // SE OBTIENEN LOS RATIOS DE CONSUMO QUE DEPENDEN DE LA CONCENTRACION DE AMBAS SUSTANCIAS, SEGUN LA DISTRIBUCION MICHAELIS-MENTEN [5]
27.         pCell->parameters.qg = pCell->parameters.Vg_max * pCell->parameters.mMglucose / (pCell->parameters.mMglucose * pCell->parameters.k_g);
28.         pCell->parameters.qo2 = pCell->parameters.Vo_max * pCell->parameters.mMo2 / (pCell->parameters.mMo2 * pCell->parameters.k_o2);
29.
30.
31.         // Calculates production of ATP with the available amount of oxygen and glucose in a given moment
32.         pCell->parameters.P_atp = 2 * pCell->parameters.qg + 17.0 / 3.0 * pCell->parameters.qo2;
33.
34.         // K PROLIFERATION
35.         pCell->parameters.k_prolif = Ki67_basic.transition_rate(1, 0) * (pCell->parameters.P_atp - pCell->parameters.P_atp_min) / 60.0;
36.
37.         if (pCell->parameters.k_prolif < 0)
38.         {
39.                 pCell->parameters.k_prolif = 0.0;
40.         }
41.
42.         // K NECROSIS
43.         // For necrosis max value lysed cells ratio is used
44.         pCell->parameters.k_necrosis = necrosis.transition_rate(0, 1) * (pCell->parameters.P_atp_min - pCell->parameters.P_atp) / 60.0;
45.         if (pCell->parameters.k_necrosis < 0)
46.         {
47.                 pCell->parameters.k_necrosis = 0.0;
48.         }
49.
50.         // if oxygen is under hypoxic threshold (15 mmHg) anaerobic cycle begins but cells still proliferate even lacking oxygen.
51.         // If it decreases under necrosis threshold, it stops proliferating and only lives
52.
53.
54.         if (pCell->parameters.pO2 < pCell->parameters.o2_necrosis_threshold)
55.         {
56.                 pCell->parameters.k_prolif = 0.0;
57.         }
58.         return;
59.
60. }
```

Figura 34. Fichero de visualización de resultados.

7.7. Anexo 7. Adición del efecto del lactato

En este anexo se muestra el código implementado para la adición del lactato en el modelo. Para añadir esta parte de código se realiza en el mismo método mostrado en el apartado anterior, añadiendo las líneas de código necesarias para tratar el lactato en cada iteración. Este se muestra en la Figura 35.

```
1. void get_proliferation_necrosis_rates(Cell* pCell, Phenotype& phenotype, double dt, int start_phase_index, int end_phase_index) {
2.
3.         int oxygen_substrate_index = pCell->get_microenvironment()->find_density_index("oxygen");
4.         int glucose_substrate_index = pCell->get_microenvironment()->find_density_index("glucose");
5.         int lactate_substrate_index = pCell->get_microenvironment()->find_density_index("lactate");
6.
7.         //COMBUSTION// OXYGEN:GLUCOSE RATIO -> 1G1:6O2
8.
9.                 // Regla de 3:
10.                //              20% O2 ----------- 0.28 mM
11.                //              5% O2  ----------- 0.07 mM (38 mmHg)
12.
13.                // Regla de 3 para obtener % a partir de los mmHg que tenemos (38 mmHg = 5% O2)
14.
15.         double O2_perc = (pCell->parameters.pO2 * 5) / 38;
16.         pCell->parameters.mMglucose = (pCell->parameters.pGlucose / 180) * 1000; // glucose molecular weight [mM]
17.
18.         pCell->parameters.mMo2 = (O2_perc * 0.28) / 20; // [mM]
19.
20.
21.         pCell->parameters.Vg_max = pCell->parameters.qg_max *
22.                 (1 - (1 - pCell->parameters.qg_min / pCell->parameters.qg_max) * pCell->parameters.mMglucose / (pCell->parameters.mMglucose + pCell->parameters.k_g_o));
23.         pCell->parameters.Vo_max = pCell->parameters.qo_max *
24.                 (1 - (1 - pCell->parameters.qo_min / pCell->parameters.qo_max) * pCell->parameters.mMo2 / (pCell->parameters.mMo2 + pCell->parameters.k_o_g));
25.
26.         // SE OBTIENEN LOS RATIOS DE CONSUMO QUE DEPENDEN DE LA CONCENTRACION DE AMBAS SUSTANCIAS, SEGUN LA DISTRIBUCION MICHAELIS-MENTEN [5]
27.         pCell->parameters.qg = pCell->parameters.Vg_max * pCell->parameters.mMglucose / (pCell->parameters.mMglucose * pCell->parameters.k_g);
28.         pCell->parameters.qo2 = pCell->parameters.Vo_max * pCell->parameters.mMo2 / (pCell->parameters.mMo2 * pCell->parameters.k_o2);
29.
30.
31.         // Calculates production of ATP with the available amount of oxygen and glucose in a given moment
32.         pCell->parameters.P_atp = 2 * pCell->parameters.qg + 17.0 / 3.0 * pCell->parameters.qo2;
33.
34.         // K PROLIFERATION
35.         pCell->parameters.k_prolif = Ki67_basic.transition_rate(1, 0) * (pCell->parameters.P_atp - pCell->parameters.P_atp_min) / 60.0;
36.
37.         if (pCell->parameters.k_prolif < 0)
38.         {
39.                 pCell->parameters.k_prolif = 0.0;
40.         }
41.
42.         // K NECROSIS
43.         // For necrosis max value lysed cells ratio is used
44.         pCell->parameters.k_necrosis = necrosis.transition_rate(0, 1) * (pCell->parameters.P_atp_min - pCell->parameters.P_atp) / 60.0;
45.         if (pCell->parameters.k_necrosis < 0)
46.         {
47.                 pCell->parameters.k_necrosis = 0.0;
48.         }
49.
50.         // if oxygen is under hypoxic threshold (15 mmHg) anaerobic cycle begins but cells still proliferate even lacking oxygen.
51.         // If it decreases under necrosis threshold, it stops proliferating and only lives
52.
53.
54.         if ((pCell->nearest_density_vector())[oxygen_substrate_index] < pCell->parameters.o2_hypoxic_threshold)
55.         {
56.                 //pCell->phenotype.secretion.uptake_rates[oxygen_substrate_index] = 0.0;
57.                 pCell->phenotype.secretion.secretion_rates[lactate_substrate_index] = pCell->phenotype.secretion.uptake_rates[glucose_substrate_index] * 2;
58.                 //cell_defaults.phenotype.secretion.secretion_rates[lactate_substrate_index] = phenotype.secretion.uptake_rates[glucose_substrate_index] * 2;
59.
60.                 (pCell->nearest_density_vector())[lactate_substrate_index] = (pCell->nearest_density_vector())[lactate_substrate_index]
61.                         + pCell->phenotype.secretion.secretion_rates[lactate_substrate_index];
62.
63.
64.                 pCell->parameters.k_necrosis = necrosis.transition_rate(0, 1) * pow((pCell->nearest_density_vector())[oxygen_substrate_index], 2) /
65.                         (pow(pCell->parameters.L_max, 2) - pow((pCell->nearest_density_vector())[oxygen_substrate_index], 2));
66.
67.                 if ((pCell->nearest_density_vector())[lactate_substrate_index] > pCell->parameters.L_max)
68.                 {
69.                         pCell->parameters.k_necrosis = 9e99;
70.                         //phenotype.death.trigger_death(necrosis_index);
71.
72.                 }
73.
74.                 if (pCell->parameters.pO2 < pCell->parameters.o2_necrosis_threshold)
75.                 {
76.                         pCell->parameters.k_prolif = 0.0;
77.                 }
78.
79.
80.         }
81.
82.         return;
83.
84. }
```

Figura 35. Fichero de visualización de resultados.