*Advances in Differential Equations and Control Processes* 2025, 32(1), 2489.
https://doi.org/10.59400/adecp2489

Article

# Numerical solution of a 3D mathematical model for the progression of tumor angiogenic factor in a tissue

**Melike Keleş Duman, Serdal Pamuk***

Department of Mathematics, University of Kocaeli (Umuttepe Campus), Kocaeli 41001, Turkey
*** Corresponding author:** Serdal Pamuk, spamuk@kocaeli.edu.tr

**CITATION**

Duman MK, Pamuk S. Numerical solution of a 3D mathematical model for the progression of tumor angiogenic factor in a tissue. Advances in Differential Equations and Control Processes. 2025; 32(1): 2489.
https://doi.org/10.59400/adecp2489

**ARTICLE INFO**

Received: 31 December 2024
Accepted: 5 March 2025
Available online: 18 March 2025



**Abstract:** In this work, the movement of tumor angiogenic factor in a three-dimensional tissue is obtained by the Method of Lines. This method transforms a partial differential equation into a system of ordinary differential equations together with the initial and boundary conditions. The more the number of lines is increased, the more the accuracy of the method increases. This method results in very accurate numerical solutions for linear and non-linear problems in contrast with other existing methods. We present Matlab-generated figures, which are the movement of tumor angiogenic factor in porous medium and explain the biological importance of this progression. The computer codes are also provided.

**Keywords:** method of lines; porosity; tumor angiogenic factor; numerical solution; porous medium

## 1. Introduction

Angiogenesis is known as the formation of new capillaries from pre-existing vessels in a tissue. Tumor growth and metastasis depend on angiogenesis and lymphangiogenesis triggered by a chemical signal from tumor angiogenic factor (TAF), in a phase of rapid growth [1]. Over the last few years, some mathematical models have appeared in the literature to estimate and observe the behavior of the tumor growth process.

The Method of Lines (MOL) is a general way of converting a partial differential equation (PDE) to a system of ordinary differential equations (ODE) [2,3]. The partial derivatives concerning the space variables are discretized to get a system of ODE together with initial and boundary conditions. The choice of spatial discretization and the position of the spatial discretization points are important for the performance of the method [4]. One can find a general framework for a convergence analysis of this method to nonlinear problems in [5] where the readers may find out the 'distance' between the solution to the system of ODE and the solution to the given PDE. It is worth emphasizing a few words about the accuracy of the method: High-order spatial and temporal discretization improves accuracy but increases computational cost; also, explicit schemes require small time steps for stability, while implicit schemes avoid this but are more expensive. Sometimes, choosing the 'right' scheme depends on the structure of the problem.

The advantage of the MOL is that it has not only the simplicity of the explicit methods but also the stability advantage of the implicit ones [3,6]. MOL is popularly used in science and engineering fields [7–10]. Moreover, it has wide applicability to physical, chemical, and biological systems modeled by PDE.

The MOL is applied to various types of problems such as the fractional diffusion equation [11,12], parabolic integro-differential equations [13], two-dimensional sine-Gordon equation [14], elliptic partial differential equations which describe steady-state mass and energy transport in solids [15], parabolic partial differential equations [4,16] and hyperbolic partial differential equations with stiff nonlinear source terms [17]. Also, the Authors of [18] applied the MOL for 1D, 2D and 3D heat equations with Dirichlet, Neumann, Robin and periodic boundary conditions and obtained the convergence criteria for this method for all types of boundary conditions.

In [2] the Authors applied the MOL to get the numerical solution of a 1D model of TAF for the movement of endothelial cells in the capillary while in [19] they solved numerically the analogous 2D model in ExtraCellular Matrix (ECM), where they envisaged a capillary segment of length L microns located along the $x$-axis on the interval $[0, L]$ with a tumor located $l$ microns above the $y$-axis as shown in **Figure 1**. After they scaled $x$ by $x/L$ and $y$ by $y/l$ they obtained a unit square $[0,1] \times [0,1]$, which we will be using 'this' matrix in our computations.

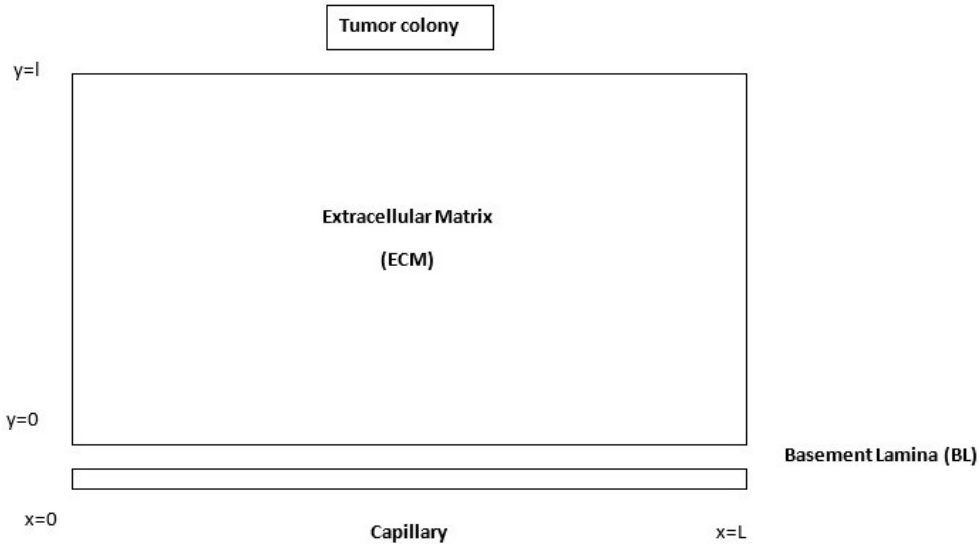


**Figure 1.** ExtraCellular Matrix (ECM).

Moving from 2D to 3D tumor modeling introduces several key novelties that significantly improve the realism and predictive power of the model. While 2D models are computationally efficient and useful for preliminary studies, 3D models capture the full spatial complexity of tumor growth, making them more biologically relevant [20–28]. In 2D models cells move only in a plane, ignoring migration in the z-axis while in 3D models cells migrate in all directions, better modeling metastasis, invasion, and ECM interactions. Therefore, the novelty in transitioning from 2D to 3D tumor modeling is that 3D tumor modeling brings us closer to real tumor behavior**,** allowing for personalized medicine, accurate treatment simulations, and better predictions for patient outcomes [18,29–34].

In light of this fact, we focus on the progression of TAF in 3D and study the following problem originally studied in [19] in 2D version:

$$u_t = \nabla.\big(D(x,y,z)\nabla(u^n)\big) + u_r(y,z,t), (x,y,z) \in \Omega, 0 < t \leq T \qquad (1)$$

$$u(x,y,z,0) = 0, (x,y,z) \in \Omega \tag{2}$$

$$u_x(0,y,z,t) - \alpha u(0,y,z,t) = 0, (y,z) \in \partial\Omega|_{x=0},\ 0 < t \leq T \tag{3}$$

$$u(1,y,z,t) = 0, (y,z) \in \partial\Omega|_{x=1},\ 0 < t \leq T \tag{4}$$

$$u_y(x,0,z,t) = 0, (x,z) \in \partial\Omega|_{y=0},\ 0 < t \leq T \tag{5}$$

$$u_y(x,1,z,t) = 0, (x,z) \in \partial\Omega|_{y=1},\ 0 < t \leq T \tag{6}$$

$$u_z(x,y,0,t) = 0, (x,y) \in \partial\Omega|_{z=0},\ 0 < t \leq T \tag{7}$$

$$u_z(x,y,1,t) = 0, (x,y) \in \partial\Omega|_{z=1},\ 0 < t \leq T \tag{8}$$

(One may find the derivation of Equation (1) in Appendix A) where $u = u(x,y,z,t)$ and $D(x,y,z)$ represent the TAF concentration and the TAF diffusion in a porous medium, respectively. Here, $\Omega \subset \boldsymbol{R}^3$ is an open bounded domain with a smooth boundary $\partial\Omega$ and $\alpha$ is a positive constant. As mentioned in [35], the diffusion process is a complex phenomenon that occurs in three dimensions. There, the Authors provide a nice analysis for their fractal diffusion equation model to obtain the relation between the diffusion coefficient and the two-scale fractality for the porosity. Therefore, one may call $n$ in Equation (1) as the parameter associated with porosity. Also, $u_r(y,z,t)$ represents a source function for the TAF that is being supplied to the tissue throughout the domain.

$$u_r(y,z,t) = \begin{cases} v_0 \left[ \dfrac{\left(1 - e^{-\left(1-\frac{r}{\delta}\right)^2}\right)}{1 - e^{-1}} \right]^m, & r \leq \delta \\ 0, & r > \delta \end{cases}$$

where $r = \sqrt{\left(y - \frac{1}{2}\right)^2 + \left(z - \frac{1}{2}\right)^2}$, $(0 \leq y,z \leq 1)$, the radius of a circle centered at the point $(1/2,1/2)$, and $v_0$, $\delta$ and $m$ are some positive constants (see **Figure 2**). Our source function $u_r$ is the type of a unimodal function (kind of a Gaussian Source Function). In tumor modeling it refers to a function that has a single peak, meaning the tumor growth rate initially increases in the ECM, reaches a maximum, and then declines. This function is particularly useful for capturing tumor growth dynamics influenced by resource limitations, immune response, or therapy effects**.** It is known that a unimodal function allows for the inclusion of both proliferation and apoptosis (cell death).

## 2. Method

We now apply MOL to our problem defined by Equations (1)–(8) by choosing as a block-shaped domain. The reason why we choose such a domain is that the treatment of boundary conditions in the finite difference approach (and by extension MOL) is crucial in determining the accuracy, stability and convergence of the solution. Even

small errors in how the boundary conditions are handled can lead to large inaccuracies in the solution, particularly when sharp gradients or fluxes are involved.

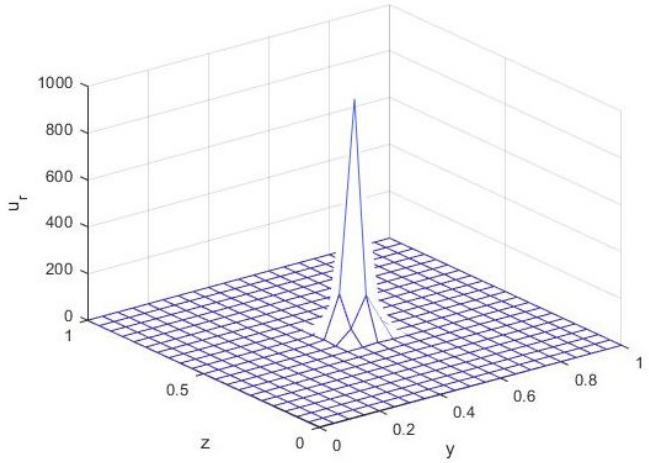


**Figure 2.** The source function $u_r(y, z, t)$.

While numerically solving our problem we may have sharp gradients (on the boundaries of the domain) especially for nonlinear cases. Therefore, we take $\Omega = [0,1] \times [0,1] \times [0,1]$ and envisage a uniform grid $W = \{(x_i, y_j, z_k, t)\colon x_i = (i-1)h_x, y_j = (j-1)h_y, z_k = (k-1)h_z, i = 1\colon M, j = 1\colon N, k = 1\colon K, h_x = 1/(M-1), h_y = 1/(N-1), h_z = 1/(K-1)\}$ on $\Omega$ (as sketched in **Figure 3**).

Since it is hard to find experimental data for $D(x, y, z)$ and $n$, we take them as positive constants (using literature values) for our numerical computations. We also suppose that the solution of Equation (1), $u(x, y, z, t)$ can be approximated by $u_{i,j,k} = u(x_i, y_j, z_k, t)$. We now discretize Equation (1) by central difference approximation to obtain the following difference scheme.

$$\frac{du_{i,j,k}}{dt} \sim D\left(\frac{u^n_{i+1,j,k} - 2u^n_{i,j,k} + u^n_{i-1,j,k}}{h_x^2} + \frac{u^n_{i,j+1,k} - 2u^n_{i,j,k} + u^n_{i,j-1,k}}{h_y^2} + \frac{u^n_{i,j,k+1} - 2u^n_{i,j,k} + u^n_{i,j,k-1}}{h_z^2}\right) + u_{r\,j,k} \quad (9)$$

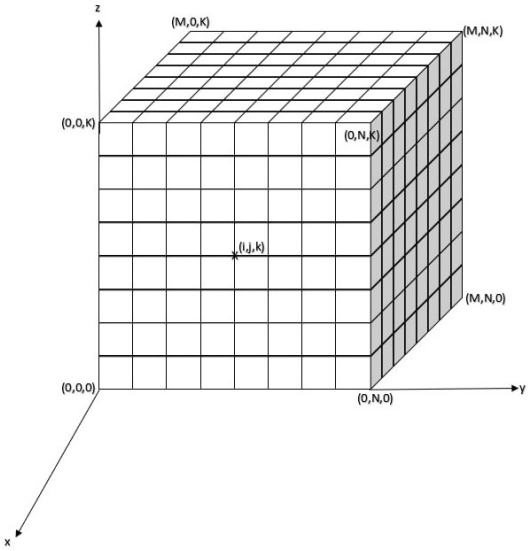


**Figure 3.** Geometry of the mesh points.

The initial and boundary conditions in Equations (2)–(8) become

$$u_{i,j,k} = 0, i = 1{:}M, j = 1{:}N, k = 1{:}K, t = 0 \tag{10}$$

$$\frac{\partial u_{1,j,k}}{\partial x} - \alpha u_{1,j,k} = 0, j = 1{:}N, k = 1{:}K, t > 0 \tag{11}$$

$$u_{M,j,k} = 0, j = 1{:}N, k = 1{:}K, t > 0 \tag{12}$$

$$\frac{\partial u_{i,1,k}}{\partial y} = \frac{\partial u_{i,N,k}}{\partial y} = 0, i = 1{:}M, k = 1{:}K, t > 0 \tag{13}$$

$$\frac{\partial u_{i,j,1}}{\partial z} = \frac{\partial u_{i,j,K}}{\partial z} = 0, i = 1{:}M, j = 1{:}N, t > 0 \tag{14}$$

Similarly, using the central difference approximations for the boundary conditions in Equations (11), (13) and (14), we get the following large system of ODE from Equation (9)

$$\frac{du_{1,1,1}}{dt} = \frac{2D}{h_x^2}u_{2,1,1}^n - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} + \frac{\alpha}{h_x}\right)u_{1,1,1}^n + \frac{2D}{h_y^2}u_{1,2,1}^n + \frac{2D}{h_z^2}u_{1,1,2}^n + u_{r_{1,1}} \tag{15}$$

$$\frac{du_{M,1,1}}{dt} = \frac{D}{h_x^2}\left(u_{M-1,1,1}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right)u_{M,1,1}^n + \frac{2D}{h_y^2}u_{M,2,1}^n + \frac{2D}{h_z^2}u_{M,1,2}^n + u_{r_{1,1}} \tag{16}$$

$$\frac{du_{1,N,1}}{dt} = \frac{2D}{h_x^2}u_{2,N,1}^n - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} + \frac{\alpha}{h_x}\right)u_{1,N,1}^n + \frac{2D}{h_y^2}u_{1,N-1,1}^n + \frac{2D}{h_z^2}u_{1,N,2}^n + u_{r_{N,1}} \tag{17}$$

$$\frac{du_{1,1,K}}{dt} = \frac{2D}{h_x^2}u_{2,1,K}^n - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} + \frac{\alpha}{h_x}\right)u_{1,1,K}^n + \frac{2D}{h_y^2}u_{1,2,K}^n + \frac{2D}{h_z^2}u_{1,1,K-1}^n + u_{r_{1,K}} \tag{18}$$

$$\frac{du_{1,N,K}}{dt} = \frac{2D}{h_x^2}u_{2,N,K}^n - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2} + \frac{\alpha}{h_x}\right)u_{1,N,K}^n + \frac{2D}{h_y^2}u_{1,N-1,K}^n + \frac{2D}{h_z^2}u_{1,N,K-1}^n + u_{r_{N,K}} \tag{19}$$

$$\frac{du_{M,1,K}}{dt} = \frac{D}{h_x^2}\left(u_{M-1,1,K}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right)u_{M,1,K}^n + \frac{2D}{h_y^2}u_{M,2,K}^n + \frac{2D}{h_z^2}u_{M,1,K-1}^n + u_{r_{1,K}} \tag{20}$$

$$\frac{du_{M,N,1}}{dt} = \frac{D}{h_x^2}\left(u_{M-1,N,1}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right)u_{M,N,1}^n + \frac{2D}{h_y^2}u_{M,N-1,1}^n + \frac{2D}{h_z^2}u_{M,N,2}^n + u_{r_{N,1}} \tag{21}$$

$$\frac{du_{M,N,K}}{dt} = \frac{D}{h_x^2}\left(u_{M-1,N,K}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right)u_{M,N,K}^n + \frac{2D}{h_y^2}u_{M,N-1,K}^n + \frac{2D}{h_z^2}u_{M,N,K-1}^n + u_{r_{N,K}} \tag{22}$$

$$\frac{du_{i,1,1}}{dt} = \frac{D}{h_x^2}\left(u_{i+1,1,1}^n + u_{i-1,1,1}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right)u_{i,1,1}^n + \frac{2D}{h_y^2}u_{i,2,1}^n + \frac{2D}{h_z^2}u_{i,1,2}^n + u_{r_{1,1}},\ \ i = 2{:}M-1 \tag{23}$$

$$\frac{du_{i,1,K}}{dt} = \frac{D}{h_x^2}\left(u_{i+1,1,K}^n + u_{i-1,1,K}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{1}{h_z^2}\right)u_{i,1,K}^n + \frac{2D}{h_y^2}u_{i,2,K}^n + \frac{2D}{h_z^2}u_{i,1,K-1}^n + u_{r_{1,K}},\ \ i = 2{:}M-1 \tag{24}$$

$$\frac{du_{i,N,1}}{dt}=\frac{D}{h_x^2}\left(u_{i+1,N,1}^n+u_{i-1,N,1}^n\right)-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u_{i,N,1}^n+\frac{2D}{h_y^2}u_{i,N-1,1}^n+\frac{2D}{h_z^2}u_{i,N,2}^n+u_{r_{N,1}},\ \ i=2:M-1 \tag{25}$$

$$\frac{du_{i,N,K}}{dt}=\frac{D}{h_x^2}\left(u_{i+1,N,K}^n+u_{i-1,N,K}^n\right)-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u_{i,N,K}^n+\frac{2D}{h_y^2}u_{i,N-1,K}^n+\frac{2D}{h_z^2}u_{i,N,K-1}^n+u_{r_{N,K}},\ \ i=2:M-1 \tag{26}$$

$$\frac{du_{1,j,1}}{dt}=\frac{2D}{h_x^2}u_{2,j,1}^n-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}+\frac{\alpha}{h_x}\right)u_{1,j,1}^n+\frac{D}{h_y^2}\left(u_{1,j+1,1}^n+u_{1,j-1,1}^n\right)+\frac{2D}{h_z^2}u_{1,j,2}^n+u_{r_{j,1}},\ j=2:N-1 \tag{27}$$

$$\frac{du_{1,j,K}}{dt}=\frac{2D}{h_x^2}u_{2,j,K}^n-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}+\frac{\alpha}{h_x}\right)u_{1,j,K}^n+\frac{D}{h_y^2}\left(u_{1,j+1,K}^n+u_{1,j-1,K}^n\right)+\frac{2D}{h_z^2}u_{1,j,K-1}^n+u_{r_{j,K}},j=2:N-1 \tag{28}$$

$$\frac{du_{M,j,1}}{dt}=\frac{D}{h_x^2}\left(u_{M-1,j,1}^n\right)-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u_{M,j,1}^n+\frac{2D}{h_z^2}u_{M,j,2}^n+\frac{D}{h_y^2}\left(u_{M,j+1,1}^n+u_{M,j-1,1}^n\right)+u_{r_{j,1}},\ \ j=2:N-1 \tag{29}$$

$$\frac{du_{M,j,K}}{dt}=\frac{D}{h_x^2}\left(u_{M-1,j,K}^n\right)-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u_{M,j,K}^n+\frac{2D}{h_z^2}u_{M,j,K-1}^n+\frac{D}{h_y^2}\left(u_{M,j+1,K}^n+u_{M,j-1,K}^n\right)+u_{r_{j,K}},\ j=2:N-1 \tag{30}$$

$$\frac{du_{1,1,k}}{dt}=\frac{2D}{h_x^2}u_{2,1,k}^n-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}+\frac{\alpha}{h_x}\right)u_{1,1,k}^n+\frac{2D}{h_y^2}u_{1,2,k}^n+\frac{D}{h_z^2}\left(u_{1,1,k+1}^n+u_{1,1,k-1}^n\right)+u_{r_{1,k}},k=2:K-1 \tag{31}$$

$$\frac{du_{M,1,k}}{dt}=\frac{D}{h_x^2}\left(u_{M-1,1,k}^n\right)-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u_{M,1,k}^n+\frac{2D}{h_y^2}u_{M,2,k}^n+\frac{D}{h_z^2}\left(u_{M,1,k+1}^n+u_{M,1,k-1}^n\right)+u_{r_{1,k}},\ \ k=2:K-1 \tag{32}$$

$$\frac{du_{1,N,k}}{dt}=\frac{2D}{h_x^2}u_{2,N,k}^n-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}+\frac{\alpha}{h_x}\right)u_{1,N,k}^n+\frac{2D}{h_y^2}u_{1,N-1,k}\frac{D}{h_z^2}+\left(u_{1,N,k+1}^n+u_{1,N,k-1}^n\right)+u_{r_{N,k}},\ \ k=2:K-1 \tag{33}$$

$$\frac{du_{M,N,k}}{dt}=\frac{D}{h_x^2}\left(u_{M-1,N,k}^n\right)-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}h_x\right)u_{M,N,k}^n+\frac{2D}{h_y^2}u_{M,N-1,k}\frac{D}{h_z^2}+\left(u_{M,N,k+1}^n+u_{M,N,k-1}^n\right)+u_{r_{N,k}},\ \ k=2:K-1 \tag{34}$$

$$\frac{du_{1,j,k}}{dt}=\frac{D}{h_y^2}\left(u_{1,j+1,k}^n+u_{1,j-1,k}^n\right)-2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}+\frac{\alpha}{h_x}\right)u_{1,j,k}^n+\frac{D}{h_z^2}\left(u_{1,j,k+1}^n+u_{1,j,k-1}^n\right)+\frac{2D}{h_x^2}u_{2,j,k}^n+u_{r_{j,k}},\ j=2:N-1,k=2:K-1 \tag{35}$$

$$\frac{du_{M,j,k}}{dt} = \frac{D}{h_x^2}\left(u^n_{M-1,j,k}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u^n_{M,j,k} + \frac{D}{h_y^2}\left(u^n_{M,j+1,k}+u^n_{M,j-1,k}\right) + \frac{D}{h_z^2}\left(u^n_{M,j,k+1}+u^n_{M,j,k-1}\right) + u_{r_{j,k}}, j = 2:N-1, k = 2:K-1 \tag{36}$$

$$\frac{du_{i,1,k}}{dt} = \frac{D}{h_x^2}\left(u^n_{i+1,1,k}+u^n_{i-1,1,k}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u^n_{i,1,k} + \frac{2D}{h_y^2}u^n_{i,2,k} + \frac{D}{h_z^2}\left(u^n_{i,1,k+1}+u^n_{i,1,k-1}\right) + u_{r_{1,k}},\ i = 2:M-1, k = 2:K-1 \tag{37}$$

$$\frac{du_{i,N,k}}{dt} = \frac{D}{h_x^2}\left(u^n_{i+1,N,k}+u^n_{i-1,N,k}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u^n_{i,N,k} + \frac{2D}{h_y^2}u^n_{i,N-1,k} + \frac{D}{h_z^2}\left(u^n_{i,N,k+1}+u^n_{i,N,k-1}\right) + u_{r_{N,k}},\ i = 2:M-1, k = 2:K-1 \tag{38}$$

$$\frac{du_{i,j,1}}{dt} = \frac{D}{h_x^2}\left(u^n_{i+1,j,1}+u^n_{i-1,j,1}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u^n_{i,j,1} + \frac{D}{h_y^2}\left(u^n_{i,j+1,1}+u^n_{i,j-1,1}\right) + \frac{2D}{h_z^2}u^n_{i,j,2} + u_{r_{j,1}}, i = 2:M-1, j = 2:N-1, \tag{39}$$

$$\frac{du_{i,j,K}}{dt} = \frac{D}{h_x^2}\left(u^n_{i+1,j,K}+u^n_{i-1,j,K}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{1}{h_z^2}\right)u^n_{i,j,K} + \frac{2D}{h_z^2}u^n_{i,j,K-1} + \frac{D}{h_y^2}\left(u^n_{i,j+1,K}+u^n_{i,j-1,K}\right) + u_{r_{j,K}}, i = 2:M-1, j = 2:N-1 \tag{40}$$

To validate this system of ODE, since our 3D model is an extension of the 2D model by [19], we show below that our system reduces exactly to the system on page 893 of [19]. We do it by taking $k = 1$ (which corresponds to taking $z = 0$) in the uniform grid $W$ defined above so that it becomes $W = \{(x_i, y_j, t): x_i = (i-1)h_x, y_j = (j-1)h_y, i = 1:M, j = 1:N, h_x = 1/(M-1), h_y = 1/(N-1)\}$ on the unit square $[0,1] \times [0,1]$. In this case $u(x,y,z,t)$ can be approximated by $u_{i,j,l}$ (or simply by $u_{i,j}$), and the system of ODE Equations (15)–(40) now reads

$$\frac{du_{1,1}}{dt} = \frac{2D}{h_x^2}u^n_{2,1} - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{\alpha}{h_x}\right)u^n_{1,1} + \frac{2D}{h_y^2}u^n_{1,2} + u_{r_1} \tag{41}$$

$$\frac{du_{M,1}}{dt} = \frac{D}{h_x^2}\left(u^n_{M-1,1}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right)u^n_{M,1} + \frac{2D}{h_y^2}u^n_{M,2} + u_{r_1} \tag{42}$$

$$\frac{du_{1,N}}{dt} = \frac{2D}{h_x^2}u^n_{2,N} - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}+\frac{\alpha}{h_x}\right)u^n_{1,N} + \frac{2D}{h_y^2}u^n_{1,N-1} + u_{r_N} \tag{43}$$

$$\frac{du_{M,N}}{dt} = \frac{D}{h_x^2}\left(u^n_{M-1,N}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right)u^n_{M,N} + \frac{2D}{h_y^2}u^n_{M,N-1} + u_{r_N} \tag{44}$$

$$\frac{du_{i,1}}{dt} = \frac{D}{h_x^2}\left(u^n_{i+1,1}+u^n_{i-1,1}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right)u^n_{i,1} + \frac{2D}{h_y^2}u^n_{i,2} + u_{r_1}, i = 2:M-1 \tag{45}$$

$$\frac{du_{i,N}}{dt} = \frac{D}{h_x^2}\left(u^n_{i+1,N}+u^n_{i-1,N}\right) - 2D\left(\frac{1}{h_x^2}+\frac{1}{h_y^2}\right)u^n_{i,N} + \frac{2D}{h_y^2}u^n_{i,N-1} + u_{r_N}, i = 2:M-1 \tag{46}$$

$$\frac{du_{1,j}}{dt} = \frac{2D}{h_x^2}u_{2,j}^n - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2} + \frac{\alpha}{h_x}\right)u_{1,j}^n + \frac{D}{h_y^2}\left(u_{1,j+1}^n + u_{1,j-1}^n\right) + u_{r_j}, j = 2{:}N-1 \quad (47)$$

$$\frac{du_{M,j}}{dt} = \frac{D}{h_x^2}\left(u_{M-1,j}^n\right) + \frac{D}{h_y^2}\left(u_{M,j+1}^n + u_{M,j-1}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right)u_{M,j}^n + u_{r_j}, j = 2{:}N-1 \quad (48)$$

$$\frac{du_{i,j}}{dt} = \frac{D}{h_x^2}\left(u_{i+1,j}^n + u_{i-1,j}^n\right) - 2D\left(\frac{1}{h_x^2} + \frac{1}{h_y^2}\right)u_{i,j}^n + \frac{D}{h_y^2}\left(u_{i,j+1}^n + u_{i,j-1}^n\right) + u_{r_j}, i = 2{:}M-1, j = 2{:}N-1 \quad (49)$$

which is exactly the system of ODE on page 893 of [19].

## 3. Numerical experiments

In this section we obtain the numerical solution of the initial-boundary value problem given in Equations (1)–(8). We let $D = 0.0036, \alpha = 0.05, v_0 = 1000, m = 3, \delta = 0.15, M = 21, N = 21$ and $K = 21$ for numerical purposes. We take $n = 1$ for the linear case and take $n = 2$ for the non-linear case. We write the following computer code in Matlab in order to perform numerical solutions of the large ODE system appearing in Equations (15)–(40) together with the initial and boundary conditions given in Equations (10)–(14). The layout of the code is as follows.

First, we give the Matlab code that solves our system of ODE's, and name this code as 'fparabolic3dsolver.m'. Second, we present the code that generates the source function $u_r\ (y, z, t)$ whose name is 'bound.m'. Finally, the discretized ODE system (15)–(40) is coded in an m-file called 'fparabolic3d.m'. As seen in the code fparabolic3dsolver.m, we have obtained the numerical solution of our large system of ODE by using the ODE solver ODE45 built in Matlab. The contents of these m-files can be found in Appendix B (Algorithm 1).

### 3.1. A test problem on the accuracy of MOL

Since it is difficult to find an analytical solution to realistic PDE problems in science, the numerical solution is an approximation to the analytical solution, and one expects that it represents the analytical solution with good accuracy. The accuracy of the solution of our ODE system (obtained from the discretization of the given PDE) satisfies the user-specified error tolerances, which ensures the semi-analytical property inherent in this class of methods. Replacing the second spatial derivatives $\frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}, \frac{\partial^2 u}{\partial z^2}$ by three-point central difference (with $O(h^2)$ accuracy) leads to a set of ODE defined on discrete lines of $x_i = (i-1)h_x, y_j = (j-1)h_y, z_k = (k-1)h_z, (i = 1{:}M, j = 1{:}N, k = 1{:}K)$ as appear in Equations (15)–(40). Note that the stability limit places an upper limit on $\Delta t$ for a given $D$, $\Delta x, \Delta y$ and $\Delta z$; if one attempts to increase the accuracy of our ODE system by using a smaller $\Delta x, \Delta y$ and $\Delta z$ (larger number of grid points in $x, y$ and $z$ by increasing $M, N$ and $K$), a smaller value of $\Delta t$ is required to keep the CFL number, $D\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)$, below its critical value. We have used 'ode45', an ODE solver built in Matlab, to solve our system of ODE. This function implements a Runge-Kutta method with a variable time step for efficient computation. A 4th order Runge-Kutta solver uses 4 evaluations, so it can exactly integrate cubic

functions. A 5th order Runge-Kutta solver actually uses 6 terms, so it can be used for quartic and quintic functions. Since 'ode45' uses both 4th and 5th order methods to obtain error estimates in each step, these methods will have a growing error discrepancy when going from cubic to quartic functions and beyond. This is because the 4th order method will not be able to integrate functions of order higher than a cubic very accurately, and it will do progressively worse as the order increases. In order to obtain a higher accuracy for polynomials of order 4 and above, reduce the 'RelTol' (Relative Error Tolerance) and 'AbsTol' (Absolute Error Tolerance) properties using the 'odeset' function. Since 'ode45' automatically chooses the value of $\Delta t$ for each time step it is very accurate.

We are now testing the accuracy of our method on the following initial-boundary value problem:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + 7, 0 < x, y, z < 1, t \geq 0 \tag{50}$$

$$u(x, y, z, 0) = x(1 - x) + y(1 - y) + z(1 - z), 0 < x, y, z < 1 \tag{51}$$

$$u_x(0, y, z, t) = u_y(x, 0, z, t) = u_z(x, y, 0, t) = 1, 0 < x, y, z < 1 \tag{52}$$

$$u_x(1, y, z, t) = u_y(x, 1, z, t) = u_z(x, y, 1, t) = -1, 0 < x, y, z < 1 \tag{53}$$

It is easy to see that the exact solution of the above problem is

$$u(x, y, z, t) = x(1 - x) + y(1 - y) + z(1 - z) + t \tag{54}$$

**Figures 4** and **5** show the comparisons of numerical and exact solutions of Equations (50)–(54) at $t = 0.5$ and $t = 1$ (with $M = N = K = 21$), respectively. And, the accuracy result with respect to the MOL solution provided by 'ode45' is obtained with the maximum relative error tolerance of $2.0 \times 10^{-10}$ for the solution at $t = 0.5$, while it is obtained with the maximum relative error tolerance of $1.0 \times 10^{-12}$ for the solution at $t = 1$. Of course, the length of the integration interval also matters. Longer intervals may accumulate errors despite high local accuracy per step due to numerical instability or propagation of round-off errors. Since we obtained numerical instability after $t = 1.2$, we quit the solver. One could without a doubt continue solving the problem after this time by increasing the relative error tolerance. But doing this can result in impaired performance.

## 3.2. Error analysis of MOL

In this subsection we provide a $L2$ norm error analysis by comparing the exact and MOL solutions of the problem defined in Equations (50)–(53). As it is well known the $L2$ norm error is defined by the formula

$$L2 = \sqrt{\Delta x \sum_{j=1}^{N} \left(u_j^{MOL} - u_j^{exact}\right)^2}$$

where $N$ is the number of grid points and $\Delta x = \frac{1}{N}$.

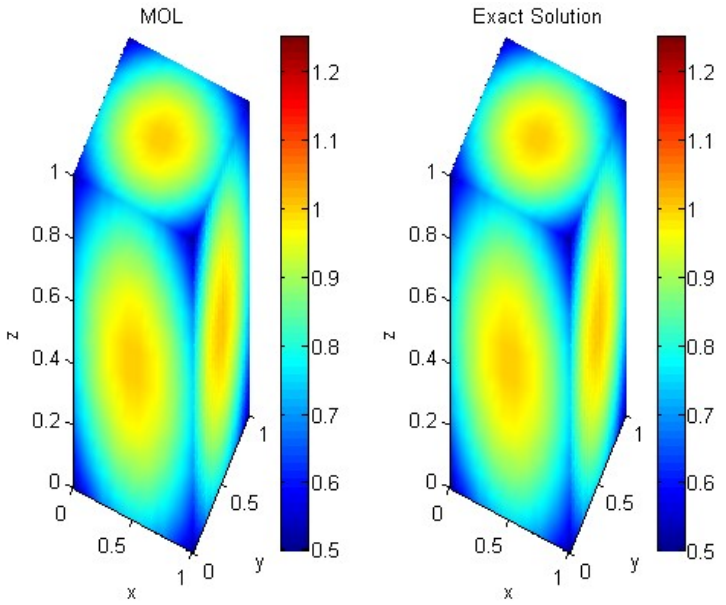


**Figure 4.** Comparison of MOL and exact solutions of Equations (50)–(54) at $t = 0.5$ with $M = N = K = 21$.

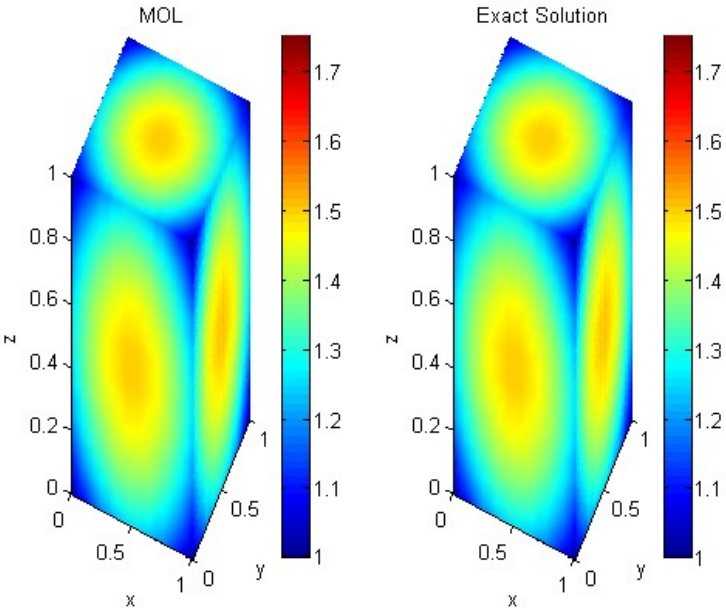


**Figure 5.** Comparison of MOL and exact solutions of Equations (50)–(54) at $t = 1$ with $M = N = K = 21$.

In our computations, we take $0 \leq x,\ y, z \leq 1$ and $M = N = K = 21$. We now use different values of *x, y* and z in the MOL and exact solutions of the problem, and obtain the $L2$ norm errors in **Table 1** below. Of course, these errors can be made smaller by choosing $N$ large.

As seen from **Table 1** the $L2$ norm errors are pretty small for various values of *x, y* and $z$. These results are in good agreement with the pictures we have obtained in **Figures 4** and **5**.

**Table 1.** $L2$ norm errors.

| $t$ | $x$ | $y$ | $z$ | $u_{exact}$ | $u_{MOL}$ | $L2$ errors |
|---|---|---|---|---|---|---|
| 0.5 | 0 | 0 | 0 | 0.5000 | 0.499999999668361 | $7.236963428151917 \times 10^{-11}$ |
| 0.5 | 0.1 | 0.1 | 0.1 | 0.7700 | 0.769999999668376 | $7.236620615357449 \times 10^{-11}$ |
| 0.5 | 0.2 | 0.2 | 0.2 | 0.9800 | 0.979999999668422 | $7.235632151610860 \times 10^{-11}$ |
| 0.5 | 0.3 | 0.3 | 0.3 | 1.1300 | 1.129999999668492 | $7.234091311064704 \times 10^{-11}$ |
| 0.5 | 0.4 | 0.4 | 0.4 | 1.2200 | 1.219999999668581 | $7.232157992266225 \times 10^{-11}$ |
| 0.5 | 0.5 | 0.5 | 0.5 | 1.2500 | 1.249999999668679 | $7.230026011636324 \times 10^{-11}$ |
| 0.5 | 0.6 | 0.6 | 0.6 | 1.2200 | 1.219999999668776 | $7.227903721827468 \times 10^{-11}$ |
| 0.5 | 0.7 | 0.7 | 0.7 | 1.1300 | 1.129999999668864 | $7.225984939260558 \times 10^{-11}$ |
| 0.5 | 0.8 | 0.8 | 0.8 | 0.9800 | 0.979999999668933 | $7.224473171177536 \times 10^{-11}$ |
| 0.5 | 0.9 | 0.9 | 0.9 | 0.7700 | 0.769999999668977 | $7.223506511778297 \times 10^{-11}$ |
| 0.5 | 1 | 1 | 1 | 0.5000 | 0.499999999668992 | $7.223177023862767 \times 10^{-11}$ |
| 1 | 0 | 0 | 0 | 1 | 0.999999999421016 | $1.263446608235148 \times 10^{-10}$ |
| 1 | 0.1 | 0.1 | 0.1 | 1.2700 | 1.269999999421016 | $1.263447092776200 \times 10^{-11}$ |
| 1 | 0.2 | 0.2 | 0.2 | 1.4800 | 1.479999999421016 | $1.263445639153044 \times 10^{-11}$ |
| 1 | 0.3 | 0.3 | 0.3 | 1.6300 | 1.629999999421017 | $1.263444185529887 \times 10^{-11}$ |
| 1 | 0.4 | 0.4 | 0.4 | 1.7200 | 1.719999999421018 | $1.263443216447782 \times 10^{-11}$ |
| 1 | 0.5 | 0.5 | 0.5 | 1.7500 | 1.749999999421018 | $1.263442731906730 \times 10^{-11}$ |
| 1 | 0.6 | 0.6 | 0.6 | 1.7200 | 1.719999999421017 | $1.263443700988834 \times 10^{-11}$ |
| 1 | 0.7 | 0.7 | 0.7 | 1.6300 | 1.629999999421017 | $1.263444670070939 \times 10^{-11}$ |
| 1 | 0.8 | 0.8 | 0.8 | 1.4800 | 1.479999999421016 | $1.263445639153044 \times 10^{-11}$ |
| 1 | 0.9 | 0.9 | 0.9 | 1.2700 | 1.269999999421016 | $1.263447092776200 \times 10^{-11}$ |
| 1 | 1 | 1 | 1 | 1 | 0.999999999421016 | $1.263447335046726 \times 10^{-10}$ |

## 4. Discussion

In this work we have performed the MOL for the numerical solution of a mathematical model for the progression of TAF in 3D. **Figure 2** represents the source function $u_r$ , the growth factor that is being supplied to the tissue throughout the domain. This function may also allow for the situation in which TAF may be generated at certain sites in the ECM. This is needed to initiate angiogenesis by enzymatic degradation of capillary basement lamina (BL). **Figures 6** and **7** show the TAF contours at different cross-sections of the cube at $t = 20$ for $0 \leq y \leq 1$ for $n = 1$ and $n = 2$, respectively. **Figure 8**, which is generated from **Figure 6a**, shows the TAF concentrations for $y = 0.3,0.4,0.5,0.6$ at $t = 20$ with $n = 1$, and **Figure 9**, which is generated from **Figure 7a**, shows the TAF concentrations for $y = 0.3, 0.4, 0.5, 0.6$ at $t = 20$ with $n = 2$. From these two figures we obtain that the maximum values of TAF on the cube at $t = 20$ are $630.5965$ for $n = 1$, and $35.5193$ for $n = 2$. Both of these values are reached when $y = 0.5$.

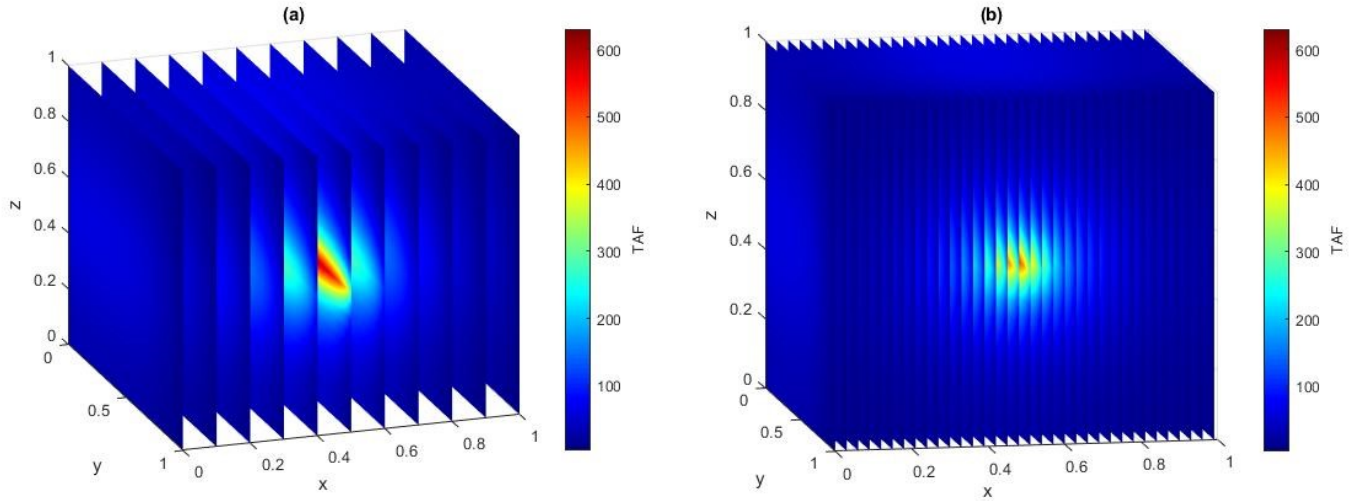


**Figure 6.** TAF contours at $t = 20$ for $0 \leq y \leq 1$ with **(a)** $h_y = 0.1$ and **(b)** $h_y = 0.03$ for $n = 1$.

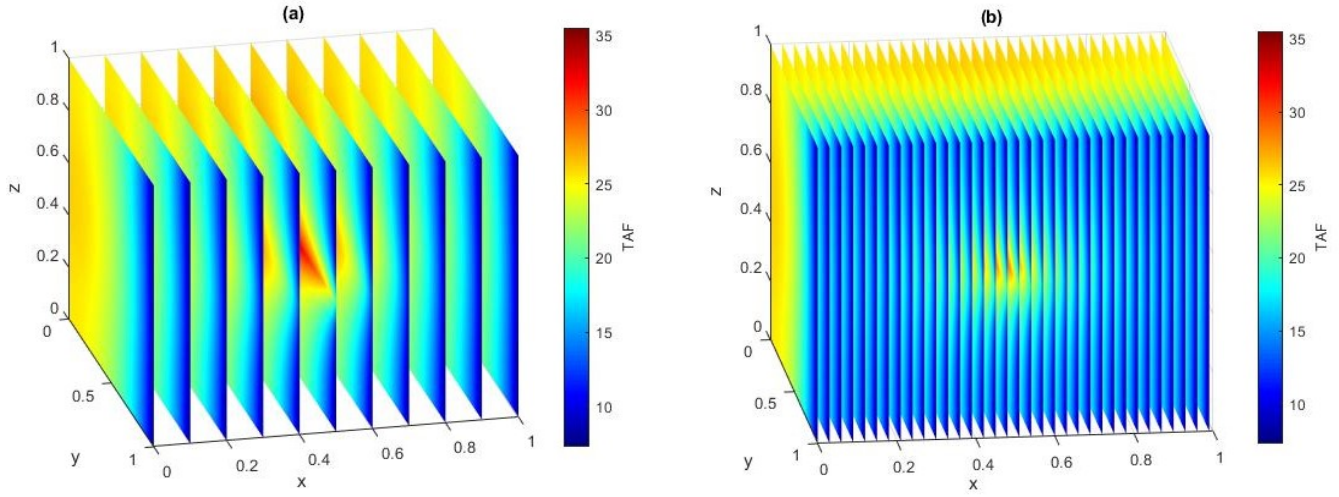


**Figure 7.** TAF contours at $t = 20$ for $0 \leq y \leq 1$ with **(a)** $h_y = 0.1$ and **(b)** $h_y = 0.03$ for $n = 2$.

It is worth saying a few words about the steady-state solutions of the model for both linear and non-linear cases. One sees from the comparison of **Figures 10** and **11** that in the non-linear case TAF reaches the steady state at about $t = 1.32$ while it reaches the steady state at about $t = 5$ in the linear case, which is the same result as in our 2D model presented in [19]. The reasons for this are as follows: In the linear case ($n = 1$), the diffusion rate is constant, so sharp peaks in $u$ diffuse at a steady rate which can lead to slow equilibration, especially in regions with initially large $\nabla u$, while in the nonlinear case ($n > 1$), diffusion is faster where $u$ is small and slower where $u$ is large which causes sharp peaks to spread out rapidly, accelerating the decay of large gradients and smoothing the solution much more efficiently. Hence, the nonlinearity enhances gradient-driven smoothing, leading to a faster transition to a steadystate.

Also, we have tested our method to verify it by taking

$$u_r(y, z, t) = 1 - \epsilon\cos(2\pi y)\cos(2\pi z)$$

And $z = 0$ in our code since the Authors of [19] did choose this source function in their model. Here $\epsilon$ is some small positive number. Once we run the code with these choices we get **Figures 12** and **13** which show the TAF concentrations for $n = 1$ and $n = 2$, respectively. These figures are the same as the ones obtained on pages 894 and 895 of [19]. In fact, one could test our method by letting $z = 1$ in $u_r$ and get the same results.

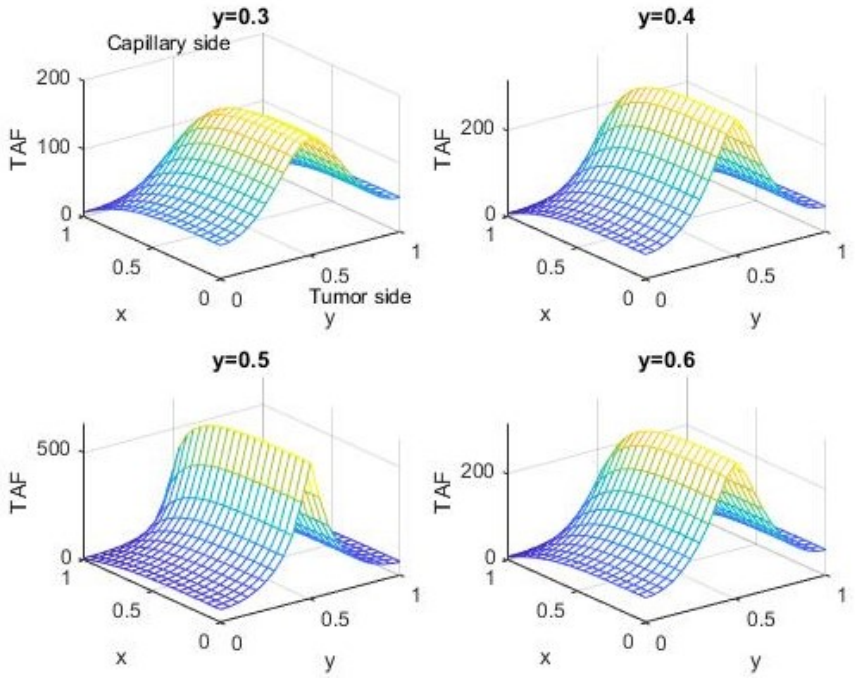


**Figure 8.** TAF concentration at $t = 20$ and $n = 1$ (linear case).

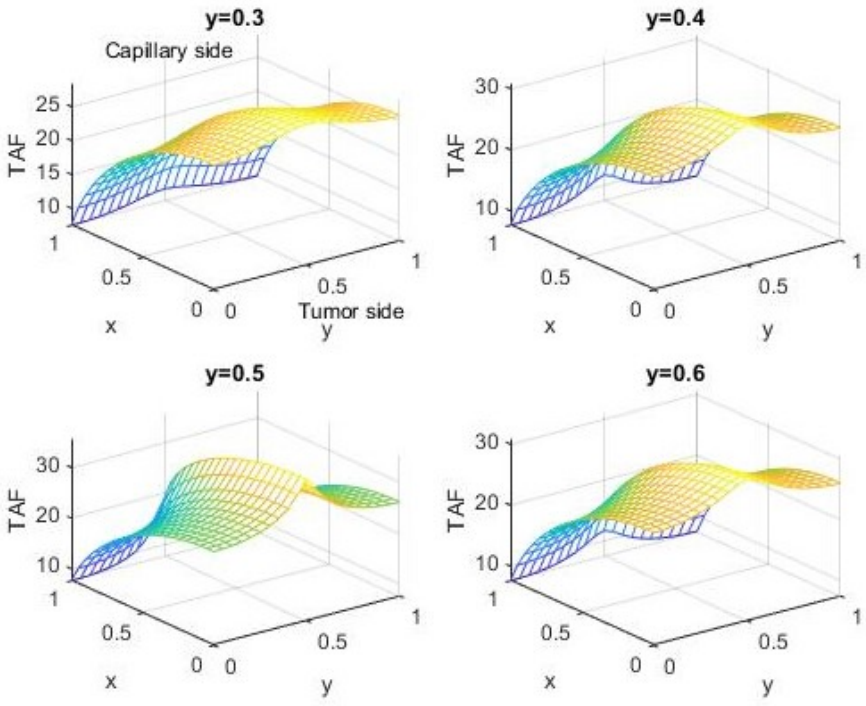


**Figure 9.** TAF concentration at $t = 20$ and $n = 2$ (nonlinear case).

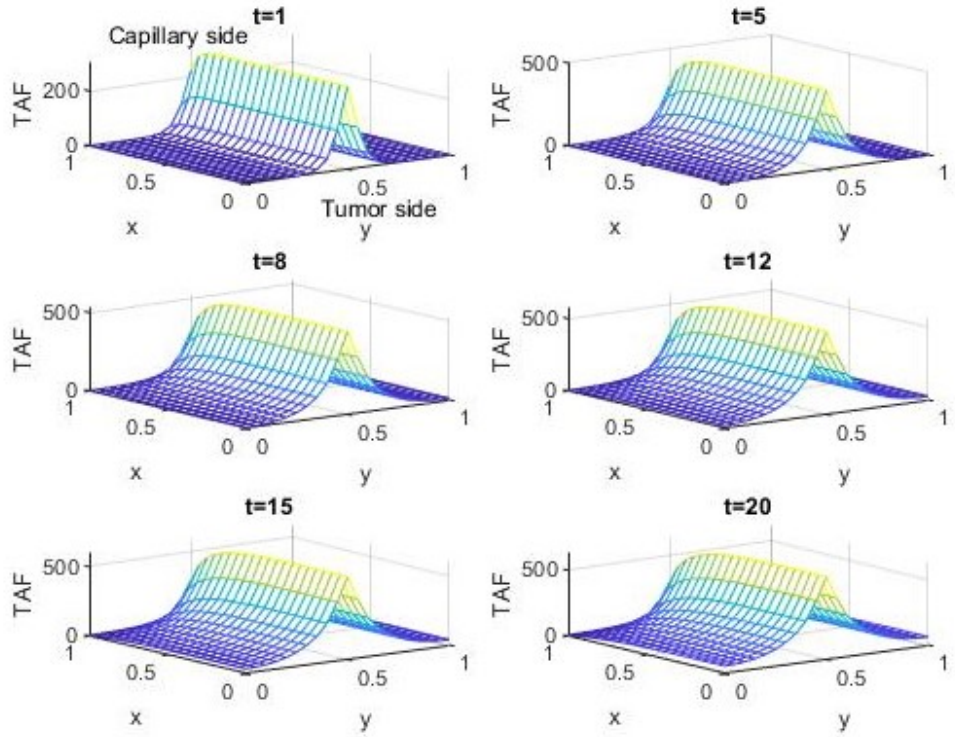


**Figure 10.** TAF concentration at $y = 0.5$ and $n = 1$ (linear case).

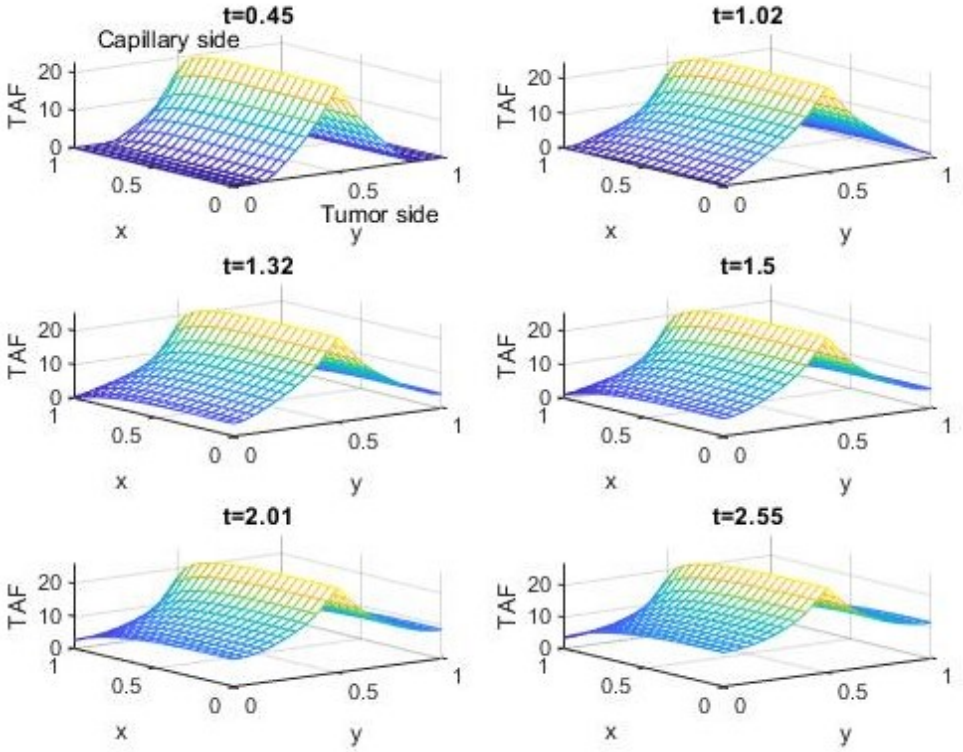


**Figure 11.** TAF concentration at $y = 0.5$ and $n = 2$ (nonlinear case).

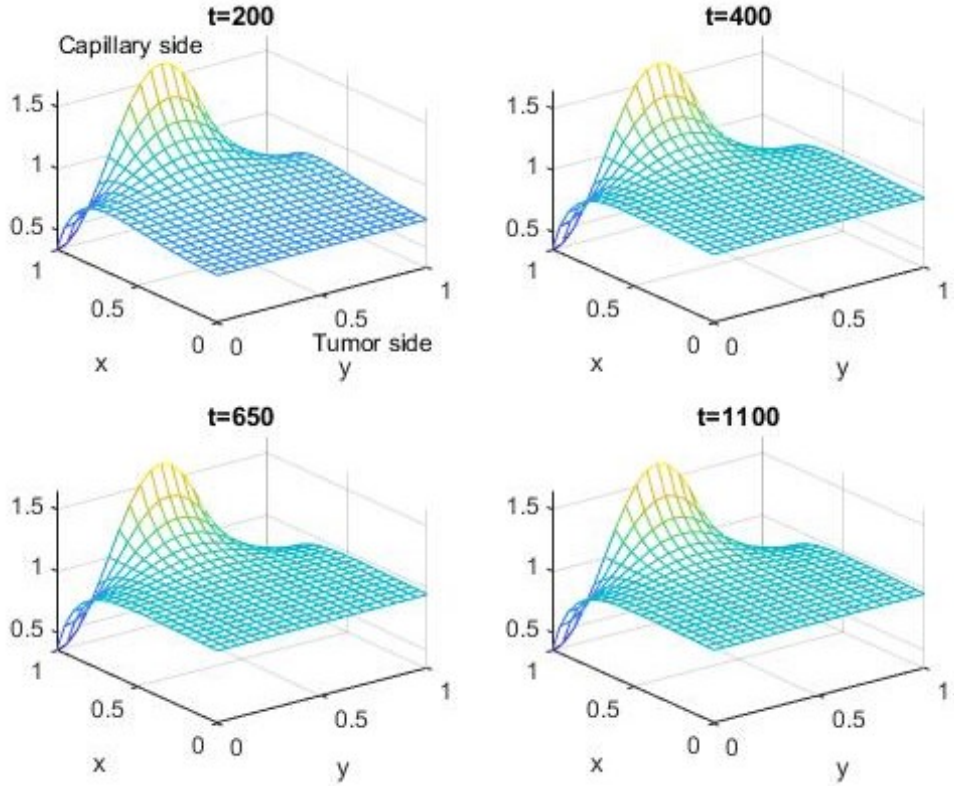


**Figure 12.** TAF concentration at $z = 0$ and $n = 1$ (linear case).

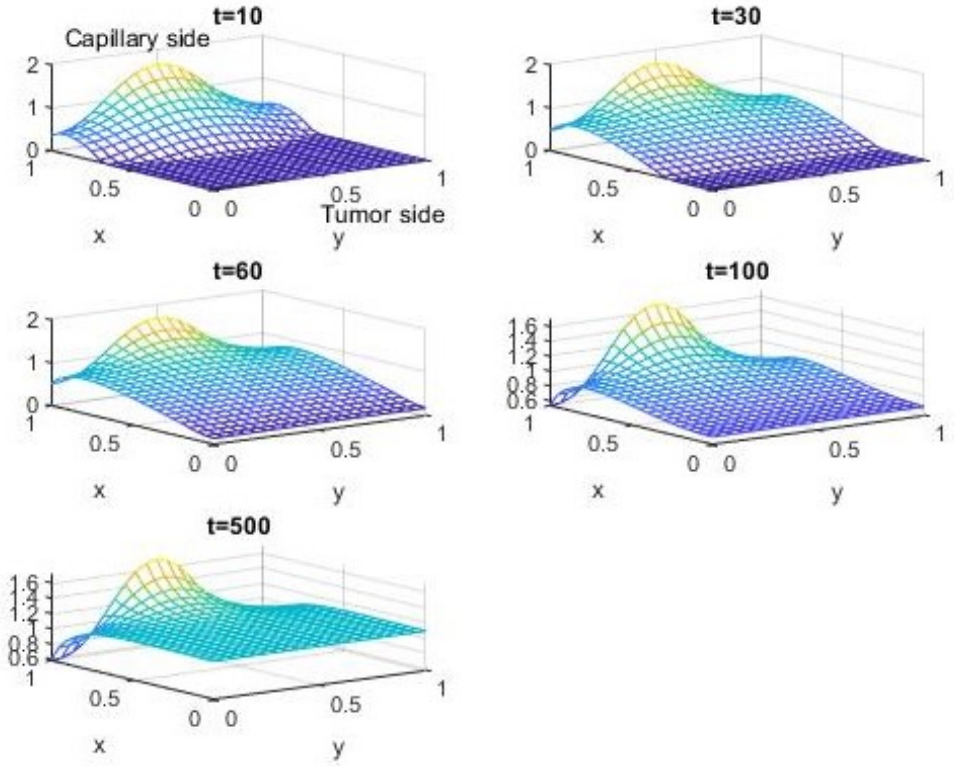


**Figure 13.** TAF concentration at $z = 0$ and $n = 2$ (nonlinear case).

## 5. Conclusion

In this study we have solved numerically a 3D mathematical model for the progression of TAF, a chemical that is needed to initiate angiogenesis. We did this using the MOL, a general way of converting a PDE to a system of ODE. When using the MOL one does not need large computer memory, linearization and physically unrealistic assumptions. Also, we have validated our large system of ODE by showing that it reduces to a relatively small system of ODE obtained by the 2D version of the model, and we have verified our method by showing a special case to match an already published paper. As a result, the MOL performs accurate and numerically stable solutions for nonlinear PDE appearing in science and engineering fields, as our outcomes support this fact.

**Author contributions:** Conceptualization, SP and MKD; methodology, SP; software, MKD; validation, SP and MKD; formal analysis, SP and MKD; investigation, SP and MKD; resources, SP; data curation, SP and MKD; writing—original draft preparation, MKD; writing—review and editing, SP and MKD; visualization, SP and MKD; supervision, SP; project administration, SP; funding acquisition, SP. All authors have read and agreed to the published version of the manuscript.

**Conflict of interest:** The authors declare no conflict of interest.

## References

1. Emamjome M, Azarnavid B, Ghehsareh HR. A reproducing kernel Hilbert space pseudospectral method for numerical investigation of a two-dimensional capillary formation model in tumor angiogenesis problem. Neural Computing and Applications. 2049; 31: 2233–2241.
2. Pamuk S, Erdem A. The method of lines for the numerical solution of a mathematical model for capillary formation: The role of endothelial cells in the capillary. Applied Mathematics and Computation. 2007; 186(1): 831–835.
3. Shakeri F, Dehghan M. The method of lines for solution of the one-dimensional wave equation subject to an integral conservation condition. Computers and Mathematics with Applications. 2008; 56(9): 2175–2188.
4. Sharaf AA, Bakodah HO. A good spatial discretisation in the method of lines. Applied Mathematics and Computation. 2005; 171(2): 1253–1263.
5. Verwer JG, Sanz-Serna JM. Convergence of method of lines approximations to partial differential equations. Computing. 1984; 33: 297–313. doi: 10.1007/BF02242274.
6. Mousa MM. Efficient numerical scheme based on the method of lines for the shallow water equations. Journal of Ocean Engineering and Science. 2018; 3(4): 303–309.
7. Ali MR. The Method of Lines Analysis of Heat Transfer of Ostwald-de Waele Fluid Generated by a Non-uniform Rotating Disk with a Variable Thickness. Journal of Applied and Computational Mechanics. 2021;7(2): 432–441. doi: 10.22055/JACM.2020.30890.1787
8. Agud Albesa L, Boix García M, Pla Ferrando ML, Navarrete SCC. A study about the solution of convection–diffusion–reaction equation with Danckwerts boundary conditions by analytical, method of lines and Crank–Nicholson techniques. Mathematical Methods in the Applied Sciences. 2023; 46(2): 2133–2164. doi: 10.1002/mma.8633
9. Shi A, Persson PO, Zahr MJ. Implicit shock tracking for unsteady flows by the method of lines. Journal of Computational Physics. 2022; 454: 110906. doi: 10.1016/j.jcp.2021.110906
10. Spiller W. An enhancement of instruments for solution of general transmission line equations with method of lines, impedance-/admittance and field transformation in combination with finite differences. Optical and Quantum Electronics. 2022; 54: 591. doi: 10.1007/s11082-022-03894-3
11. Kazem S, Dehghan M. Semi-analytical solution for time-fractional diffusion equation based on finite difference method of lines (MOL). Engineering with Computers. 2019; 35: 229–241.

12. Salehi Y, Darvishi MT, Schiesser WE. Numerical solution of space fractional diffusion equation by the method of lines and splines. Applied Mathematics and Computation. 2018; 336: 465–480.
13. Soradi-Zeid S, Mesrizadeh M. The method of lines for parabolic integro-differential equations. Journal of Mathematical Modeling. 2020; 8(3): 291–308.
14. Bratsos AG. The solution of the two-dimensional sine-Gordon equation using the method of lines. Journal of Computational and Applied Mathematics. 2007; 206(1): 251–277.
15. Subramanian VR, White RE. Semianalytical method of lines for solving elliptic partial differential equations. Chemical Engineering Science. 2004; 59(4): 781–788.
16. Manshoor B, Salleh H, Khalid A, Abdelaal MAS. Method of Lines and Runge-Kutta Method in Solving Partial Differential Equation for Heat Equation. Journal of Complex Flow. 2021; 3(1): 21–25.
17. Ahmad I, Berzins M. MOL solvers for hyperbolic PDEs with source terms. Mathematics and Computers in Simulation. 2001; 56(2): 115–125.
18. Kazem S, Dehghan M. Application of finite difference method of lines on the heat equation. Numerical Methods Partial Differential Eq. 2018; 34(2): 626–660.
19. Erdem A, Pamuk S. The method of lines for the numerical solution of a mathematical model for capillary formation: The role of tumor angiogenic factor in the extra-cellular matrix. Applied Mathematics and Computation. 2007; 186(1): 891–897.
20. Eikenberry SE, Sankar T, Preul MC, et al. Virtual Glioblastoma: Growth, Migration and Treatment in a Three-Dimensional Mathematical Model. Cell Cell Proliferation. 2009; 42(4): 511–528.
21. Perfahl H, Byrne HM, Chen T, et al. Multiscale Modelling of Vascular Tumour Growth in 3D: The Roles of Domain Size and Boundary Conditions. PLoS One. 2011; 6(4): e14790.
22. Sakai K, Hayashi T, Sakai Y, et al. A three-dimensional model with two-body interactions for endothelial cells in angiogenesis. Scientific Reports. 2023; 13: 20549. doi: 10.1038/s41598-023-47911-1
23. Shirinifard A, Gens JS, Zaitlen BL, et al. 3D Multi-Cell Simulation of Tumor Growth and Angiogenesis. PLoS One. 2009; 4(10): e7190.
24. Stéphanou A, McDougall SR, Anderson ARA, Chaplain MAJ. Mathematical Modelling of Flow in 2D and 3D Vascular Networks: Applications to Anti-Angiogenic and Chemotherapeutic Drug Strategies. Mathematical and Computer Modelling. 2005; 41(10): 1137–1156.
25. Tang L, van de Ven AL, Guo D, et al. Computational Modeling of 3D Tumor Growth and Angiogenesis for Chemotherapy Evaluation. PLoS One. 2014; 9(1): e83962.
26. Xie H, Jiao Y, Fan Q, et al. Modeling Three-Dimensional Invasive Solid Tumor Growth in Heterogeneous Microenvironment under Chemotherapy. PLoS One. 2018; 13(10): e0206292.
27. Xu J, Vilanova G, Gomez H. Phase-Field Model of Vascular Tumor Growth: Three-Dimensional Geometry of the Vascular Network and Integration with Imaging Data. Computer Methods in Applied Mechanics and Engineering. 2020; 359: 112648.
28. Zhao G, Chen E, Yu X, et al. Three-Dimensional Model of Metastatic Tumor Angiogenesis in Response to Anti-Angiogenic Factor Angiostatin. Journal of Mechanics in Medicine and Biology. 2017; 17(10): 1750094.
29. Alias N, bin Masseri MIS, Islam MR, Khalid SN. The Visualization of Three Dimensional Brain Tumors' Growth on Distributed Parallel Computer Systems. Journal of Applied Sciences. 2009; 9(3): 505–512.
30. Karaa S, Zhang J, Yang F. A numerical study of a 3D bioheat transfer problem with different spatial heating. Mathematics and Computers in Simulation. 2005; 68(4): 375–388.
31. Al Qubeissi M. Proposing a Numerical Solution for the 3D Heat Conduction Equation. In: Proceedings of the 2012 Sixth Asia Modelling Symposium; 29–31 May 2012; Bali, Indonesia.
32. Tsega EG. Numerical Solution of Three-Dimensional Transient Heat Conduction Equation in Cylindrical Coordinates. Journal of Applied Mathematics. 2022; 1993151.
33. Tsega EG. A Numerical Solution of Three-Dimensional Unsteady State Heat Equation. International Journal of Mathematical Modelling & Computations. 2021; 11(01): 49–60.
34. Wang TY, Lee YM, Chen CP. 3D thermal-ADI—An efficient chip-level transient thermal simulator. In: Proceedings of the ISPD' 03: International Symposium on Physical Design; 6–9 April 2003; Monterey, CA, USA.
35. He CH, Liu HW, Liu C. A Fractal-Based Approach to the Mechanical Properties of Recycled Aggregate Concretes. Series: Mechanical Engineering. 2024; 22(2): 329–342. doi: 10.22190/FUME240605035H

# Appendix A

## Derivation of Equation (1)

**Fick's Law of Diffusion:** Fick's law describes the flux of a substance due to diffusion. In one dimension, it is given by

$$\boldsymbol{J} = -D\frac{\partial u}{\partial x}$$

where $\boldsymbol{J}$ is the flux (rate of flow of the substance), $D$ is the diffusion coefficient (which can be a constant or a function of position), and $u$ is the concentration.

In three dimensions, Fick's law becomes

$$J = -D(x, y, z)\nabla u$$

where $\nabla u$ is the gradient of the concentration field, and $D(x, y, z)$ is the diffusion coefficient, which may vary with spatial position (i.e., $D$ can depend on $x, y$, and $z$).

**Conservation of Mass (Continuity Equation):** The change in concentration of the substance $u$ at a point in space is governed by the conservation of mass. The rate of change of $u$ at a point is related to the flux through the boundaries of an infinitesimal volume around that point. The general form of the continuity equation is:

$$\frac{\partial u}{\partial t} = -\nabla \cdot \boldsymbol{J}$$

where $\frac{\partial u}{\partial t}$ is the rate of change of concentration at a point in time, and $\nabla \cdot \boldsymbol{J}$ is the divergence of the flux, representing how the substance is spreading out in space.

**Substituting Fick's Law:** Now, substituting the expression for the flux $\boldsymbol{J} = -D(x, y, z)\nabla u$ into the continuity equation

$$\frac{\partial u}{\partial t} = -\nabla \cdot (-D(x, y, z)\nabla u)$$

This simplifies to

$$\frac{\partial u}{\partial t} = \nabla \cdot (D(x, y, z)\nabla u)$$

This is a standard diffusion equation, where $D(x, y, z)$ is the spatially varying diffusion coefficient.

**Adding the Nonlinear Reaction Term:** The equation we have provided includes a nonlinear term $u^n$ which indicates that there is a reaction process occurring in the system that depends nonlinearly on $u$. In many physical and biological processes (such as in chemical reactions, population dynamics or progression of tumor cells in a nonlinear media), the rate of change of the concentration might depend on some power of the concentration $u$, leading to a reaction term of the form $u^n$ where $n$ is a constant that determines the nonlinear behavior.

The nonlinear term could represent, for instance, autocatalysis (where the rate of change of concentration increases as the concentration itself increases) or other forms of reaction kinetics. Hence, the equation becomes

$$u_t = \nabla \cdot (D(x, y, z)\nabla(u^n)).$$

This form of the equation suggests that the concentration of $u$ changes over time due to a combination of:

- **Diffusion** is represented by $\nabla \cdot (D(x, y, z)\nabla(u^n))$,
- **Nonlinear reaction kinetics** is the term $u^n$ that governs how the concentration itself evolves due to local interactions.

**Adding the source term:** We add a source function $u_r(y,z,t)$ to the above equation. It represents the TAF that is being supplied to the tissue throughout the domain.

$$u_t = \nabla \cdot \left(D(x,y,z)\nabla(u^n)\right) + u_r(y,z,t)$$

# Appendix B

**Algorithm 1** Matlab Code

```
%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%fparabolic3dsolver.m\\
clc; clear all; global D M N K hx hy hz alpha n tt
D=0.0036; alpha=0.05; M=21; N=21; K=21; hx=1./(M-1); hy=1./(N-1);hz=1./(K-1);
time=input('please enter the final time '); tt=input('the value of tt'); n=input('the value of porosity')
tspan=linspace(0,time,201); x=linspace(0,1,M); y=linspace(0,1,N); z=linspace(0,1,K); u0(1:(M*N*K))=0; u0=u0(:);
 option=odeset('reltol',10^-8,'abstol',10^-8);
 [t,U]=ode45('fparabolic3d',t,u0,option); u3=[];
 for i=1:length(t)
 u3=[u3 U(i,:)];
 end
 u3=u3(:);
 u_numeric = reshape(u3,M,N,K,length(t));
 u_numeric_1=pagetranspose(u_numeric); [X Y Z]=meshgrid(x,y,z);
 slice(X,Y,Z,u_numeric_1(:,:,:,find(tspan==tt)),[],[0:0.1:1],[]); xlabel('x'); ylabel('y');  zlabel('z'); view([45 50 55])
 colormap(jet) cb=colorbar; shading interp cb.Label.String='TAF';
 \%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\bound.m\\
 function v=bound(y,z);
 delta=0.15; v0=1000; N=21; K=21; a=1/2; b=1/2; y=linspace(0,1,N); z=linspace(0,1,K);
 [Y ,Z]=meshgrid(y,z);
 f = @(y,z) sqrt(((y-a).^2 + (z-b).^2)); r=f(Y,Z); m=3; U_y= length(y); U_z= length(z);
 for i=1:U_y;
 for j=1:U_z;
  if r(i,j)<=delta;
  v(i,j)=v0*((1-exp(-(1-r(i,j)/delta).^2))/(1-exp(-1))).^m;
  else r(i,j)>$delta;
 v(i,j)=0;
  end
  end
  end
\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%\%fparabolic3d.m\\
function u2=fparabolic3d(t,u);
global D alpha M N K hx hy hz n y z
u=reshape(u,M,N,K); A=bound(y,z);
D=0.0036; alpha=0.05; M=21; N=21; K=21; hx=1./(M-1); hy=1./(N-1); hz=1./(K-1); H_x=D./hx^2; H_y=D./hy^2;
 H_z=D./hz^2; H=1./hx^2+1./hy^2+1./hz^2;
u2(1,1,1)=(2*H_x).*u(2,1,1)^n-2*D.*(u(1,1,1^n).*(H+alpha/hx)+
(2*H_y).*u(1,2,1)^ n+(2*H_z).*u(1,1,2)^ n  +A(1,1);
u2(M,1,1)=(H_x).*(u(M-1,1,1)^n)+(2*H_y).*u(M,2,1)^n+
(2*H_z).*u(M,1,2)^n-2*D*u(M,1,1)^ n.*H +A(1,1);
u2(1,N,1)=(2*H_x).*u(2,N,1)^n+(2*H_y).*u(1,N-1,1)^n+
(2*H_z).*u(1,N,2)^n-2*D*u(1,N,1)^n.*(H+alpha/hx) +A(N,1);
u2(1,1,K)=(2*H_x).*u(2,1,K)^n+(2*H_y).*u(1,2,K)^n+
(2*H_z).*u(1,1,K-1)^n-2*D*u(1,1,K)^n.*(H+alpha/hx) +A(1,K);
u2(1,N,K)=(2*H_x).*u(2,N,K)^n+(2*H_y).*u(1,N-1,K)^n+
(2*H_z).*u(1,N,K-1)^n-2*D*u(1,N,K)^n.*(H+alpha/hx) +A(N,K);
u2(M,1,K)=(H_x).*(u(M-1,1,K)^n)+(2*H_y).*u(M,2,K)^n+...
(2*H_z).*u(M,1,K-1)^n-2*D*u(M,1,K)^n.*(H) +A(1,K);
u2(M,N,1)=(H_x).*( u(M-1,N,1)^n)+(2*H_y).*u(M,N-1,1)^n+...
(2*H_z).*u(M,N,2)^n-2*D*u(M,N,1)^n.*(H)+ A(N,1);
u2(M,N,K)=(H_x).*(u(M-1,N,K)^n)+(2*H_y).*u(M,N-1,K)^n+(2*H_z).*u(M,N,K-1)^n-2*D*u(M,N,K)^n.*(H) +A(N,K);
for i=2:M-1;
u2(i,1,1)= (H_x).*(u(i+1,1,1)^n+u(i-1,1,1)^n)+(2*H_y).*u(i,2,1)^n+...
(2*H_z).*u(i,1,2)^n-2*D.*u(i,1,1)^n.*(H) +A(1,1);
u2(i,1,K)=(H_x).*(u(i+1,1,K)^n+u(i-1,1,K)^n)+(2*H_y).*u(i,2,K)^n+...
(2*H_z).*u(i,1,K-1)^n-2*D.*u(i,1,K)^n.*(H) +A(1,K);
u2(i,N,1)=(H_x).*(u(i+1,N,1)^n+u(i-1,N,1)^n)+(2*H_y).*u(i,N-1,1)^n+...
```

**Algorithm 1** (*Continue*)

```
(2*H_z).*u(i,N,2)^n-2*D.*u(i,N,1)^n.*(H) +A(N,1);
u2(i,N,K)=(H_x).*(u(i+1,N,K)^n+u(i-1,N,K)^n)+(2*H_y).*u(i,N-1,K)^n+...
(2*H_z).*u(i,N,K-1)^n-2*D.*u(i,N,K)^n.*(H) +A(N,K);
end
for j=2:N-1;
u2(1,j,1)=(2*H_x).*u(2,j,1)^ n+(2*H_z).*u(1,j,2)^n+
(H_y).*(u(1,j+1,1)^n+u(1,j-1,1)^n)-2*D.*u(1,j,1)^ n.*(H+alpha/hx) +A(j,1);
u2(1,j,K)=(2*H_x).*u(2,j,K)^n+(2*H_z).*u(1,j,K-1)^n-
2*D.*u(1,j,K)^n.*(H+alpha/hx)+(H_y).*(u(1,j+1,K)^n+u(1,j-1,K)^n) +A(j,K);
u2(M,j,1)=(H_x).*(u(M-1,j,1)^n)-2*D.*u(M,j,1)^n.*(H)+...
(H_y).*(u(M,j+1,1)^n+u(M,j-1,1)^n)+(2*H_z).*u(M,j,2)^n+A(j,1);
u2(M,j,K)=(H_x).*(u(M-1,j,K)^n)-2*D.*u(M,j,K)^n.*(H)+...
(H_y).*(u(M,j+1,K)^n+u(M,j-1,K)^n)+(2*H_z).*u(M,j,K-1)^n+A(j,K);
end
for k=2:K-1;
u2(1,1,k)= (2*H_x).*u(2,1,k)^n+(2*H_y).*u(1,2,k)^n+...
(H_z).*(u(1,1,k+1)^n+u(1,1,k-1)^n)-2*D.*u(1,1,k)^n.*(H+alpha/hx) +A(1,k);
u2(M,1,k)=(H_x).*(u(M-1,1,k)^n)+(2*H_y).*u(M,2,k)^n+...
(H_z).*(u(M,1,k+1)^n+u(M,1,k-1)^n)-2*D.*u(M,1,k)^n.*(H) +A(1,k);
u2(1,N,k)=(2*H_x).*u(2,N,k)^n+(2*H_y).*u(1,N-1,k)^n+...
(H_z).*(u(1,N,k+1)^n+u(1,N,k-1)^n)-2*D.*u(1,N,k)^n.*(H+alpha/hx) +A(N,k);
u2(M,N,k)=(H_x).*(u(M-1,N,k)^n)+(2*H_y).*u(M,N-1,k)^n+...
(H_z).*(u(M,N,k+1)^n+u(M,N,k-1)^n)-2*D.*u(M,N,k)^n.*(H) +A(N,k);
end
for j=2:N-1;
for k=2:K-1;
u2(1,j,k)= (2*H_x).*u(2,j,k)^n+(H_y).*(u(1,j+1,k)^n+u(1,j-1,k)^n)+...
(H_z).*(u(1,j,k+1)^n+u(1,j,k-1)^n)-2*D.*u(1,j,k)^n.*(H+alpha/hx) +A(j,k);
u2(M,j,k)=(H_x).*(u(M-1,j,k)^n)-2*D.*u(M,j,k)^n.*(H)+...
(H_y).*(u(M,j+1,k)^n+u(M,j-1,k)^n)+(H_z).*(u(M,j,k+1)^n+u(M,j,k-1)^n) +A(j,k);
end
end
for i=2:M-1;
for k=2:K-1;
u2(i,1,k)=(H_x).*(u(i+1,1,k)^n+u(i-1,1,k)^n)+(2*H_y).*u(i,2,k)^n+...
(H_z).*(u(i,1,k+1)^n+u(i,1,k-1)^n)-2*D.*u(i,1,k)^n.*(H) +A(1,k);
u2(i,N,k)=(H_x).*(u(i+1,N,k)^n+u(i-1,N,k)^n)+(2*H_y).*u(i,N-1,k)^n-
2*D.*u(i,N,k)^n.*(H)+(H_z).*(u(i,N,k+1)^n+u(i,N,k-1)^n) +A(N,k);
end
end
for i=2:M-1;
for j=2:N-1;
u2(i,j,1)= (H_x).*(u(i+1,j,1)^n+u(i-1,j,1)^n)+(2*H_z).*u(i,j,2)^n+...
(H_y).*(u(i,j+1,1)^n+u(i,j-1,1)^n)-2*D.*u(i,j,1)^n.*(H) +A(j,1);
u2(i,j,K)=(H_x).*(u(i+1,j,K)^n+u(i-1,j,K)^n)+(2*H_z).*u(i,j,K-1)^n-2*D.*u(i,j,K)^n.*(H)+(H_y).*(u(i,j+1,K)^n+u(i,j-
1,K)^n) +A(j,K);
end
end
for i=2:M-1;
for j=2:N-1;
for k= 2:K-1;
u2(i,j,k)=(H_x).*(u(i+1,j,k)^n+u(i-1,j,k)^n)-2*D.*u(i,j,k)^n.*(H)+...
(H_y).*(u(i,j+1,k)^n+u(i,j-1,k)^n)+(H_z).*(u(i,j,k+1)^n+u(i,j,k-1)^n) +A(j,k);
end
end
end
u2=u2(:);
```