Mathematics and Computers in Simulation 234 (2025) 31–49

Contents lists available at ScienceDirect

Mathematics and Computers in Simulation

journal homepage: www.elsevier.com/locate/matcom

Original articles

A new MATLAB software for numerical computation of biological observables for metastatic tumor growth

Iulia Martina Bulai [a,*], Maria Carmela De Bonis [b], Concetta Laurita [b]

[a] Dipartimento di Scienze Chimiche, Fisiche, Matematiche e Naturali, University of Sassari, Via Vienna 2, 7100, Sassari, Italy
[b] Dipartimento di Scienze di Base e Applicate, University of Basilicata, Via dell'Ateneo Lucano 10, 85100 Potenza, Italy

ARTICLE INFO

MSC:
35Q92
45D05
45A05
65M25

Keywords:
Metastatic tumor growth
PDEs
Volterra integral equation
Nyström method

ABSTRACT

In this paper a new MATLAB Toolbox is introduced, Metastatic Tumor Growth Modeling (MTGM). MTGM Toolbox is freely available on a GitHub repository https://github.com/IuliaMartinaBulai/MTGM_Toolbox and equips the researchers with five different functionalities. The first one refers to the numerical resolution of a general Volterra Integral Equation (VIE) of the second kind on the positive semiaxis while the last four ones to the computation of biological observables related to metastatic tumor growth models. In particular, the computed observables are the cumulative number of metastases (CNM) and the total metastatic mass (TMM) derived from: (i) a 1D metastatic tumor growth (t.g.) model where the analytical solution of the ODE describing the t.g. law is known, (ii) a 1D model where this solution has to be numerically computed, (iii) a 2D non-autonomous metastatic t.g. model where also the treatment is considered, (iv) a 2D autonomous model, i.e., in the absence of therapies. Moreover, the Toolbox implementation was designed in order to give the users the possibility to extend its functionalities.

1. Introduction

The numerical resolution of mathematical models describing complex systems is an important tool for a better understanding of the dynamics described by the model itself and of the real life problem studied in the model, [1–4]. The main objectives of the newly introduced numerical methods are, above others, to ensure an accurate numerical solution and a fast run time execution without loosing in accuracy of the solution. Considering the real life problems described via PDEs (partial differential equations), solving directly the PDE model, in most cases, could be prohibitive both in the matter of machine memory and of execution time. In this paper we focus on a model describing the metastatic tumor growth (t.g.) whose unknown is the metastatic density. In this particular case it is of paramount importance to predict how the metastases behave for long intervals of time, in the shortest possible time and preferably using a personal computer.

In [5,6], the authors proposed an efficient numerical method to compute biological observables of interest as solution of some Volterra Integral Equations (VIE) arising by a reformulation of the PDE metastatic growth model. Moreover, an open source MATLAB Toolbox, called `VIE_Toolbox`, was published in [7]. The Toolbox can be used to solve VIEs obtained from the generalized version of the Iwata metastatic PDE model [8], where different laws for the growth of the primary and secondary tumors (metastases) were considered and it was also assumed that the new born metastases can be formed by clusters of several cells. Furthermore, in [6] the authors focused on the numerical solution of VIEs obtained from a mathematical model, developed in [9], which describes metastatic t.g. under treatment, taking also into account the angiogenic process. Beside reformulating the 2D non-autonomous PDE

* Corresponding author.
E-mail addresses: imbulai@uniss.it (I.M. Bulai), mariacarmela.debonis@unibas.it (M.C. De Bonis), concetta.laurita@unibas.it (C. Laurita).

https://doi.org/10.1016/j.matcom.2025.02.014
Received 31 July 2024; Received in revised form 7 February 2025; Accepted 10 February 2025
Available online 19 February 2025

model in terms of a VIE, an efficient and fast Nyström type numerical method to solve the VIE is introduced. The method allows to compute accurate solutions with fast time execution for short intervals of time, *e.g.*, a few weeks. Its drawback is loosing accuracy of the solution or, even, loosing convergence in the cases when the observables need to be computed for a longer interval of time. With the MATLAB Toolbox introduced in this paper we aim to overcome this issue. This goal is achieved by the introduction of an "iterative" version of the Nyström method introduced in [6]. More precisely, the first step of the iterative method consists in approximating the solution of the VIE in the interval $(0, t_0]$, where $t_0 \in \mathbb{R}^+$ is an upper threshold under which a certain precision is guaranteed. In the kth iteration of the method the solution in the interval $(kt_0, (k+1)t_0]$ is computed by solving an equivalent VIE defined on the first interval $(0, t_0]$, which involves as known terms the solutions already obtained in the previous iterations.

The major contribution of this work is the newly implemented MATLAB software Metastatic Tumor Growth Modeling (MTGM) Toolbox designed to compute numerically biological observables, *i.e.*, the cumulative number of metastases and the total metastatic mass, CNM and TMM from now on, respectively, derived from a coupled ODE-PDE metastatic t.g. model, both in the absence [5] and in the presence of treatment [9], [6]. The software comprises five different functionalities. (i) The software can be used to solve a linear VIE of the second kind on the positive semiaxis. (ii) An updated version of the `VIE_Toolbox` introduced in [7] is implemented, with the advantage of gaining in run time execution but, also, of solving the VIE for longer intervals of time without loosing in accuracy. (iii) An extension of the `VIE_Toolbox` to the cases where the analytical solutions of the ODE system describing the primary and secondary tumor growths are not known is proposed. In these cases a numerical method, such as the one implemented by the MATLAB functions `ode45` or `ode23t`, must be used. (iv) The MATLAB Toolbox compute the observables of interest in the non-autonomous 2D case, where the metastatic t.g. model takes into account also the treatment (the ODE system depends directly on the variable time). (v) The biological observables related to an autonomous 2D model describing the metastatic t.g. in the absence of therapies (the ODE system do not depend directly on time) are computed. The Toolbox is licensed under the GNU General Public License v3.0 and is openly obtainable through https://github.com/IuliaMartinaBulai/MTGM_Toolbox, which seeks to give easy access and high visibility. The main advance of the software MTGM with respect to the previously published software `VIE_Toolbox`, [7], is to expand its functionalities. In fact, `VIE_Toolbox` is applicable to solve only 1D t.g. models in the autonomous case, while MTGM offers a tool that, under the same umbrella, allows to make simulations related to different applicative models, combining 1D or 2D cases with autonomous or non-autonomous cases. Moreover, the way the software was designed encourages the interaction and contribution of the community to the toolbox, by adding new cases which correspond to new models. This has the practical advantage of giving the opportunity of analyzing and comparing different types of cancer treatments and/or different treatment schedules by means of a non invasive approach. Finally, compared to the numerical method implemented in `VIE_Toolbox`, the one implemented in MTGM, has the following two further advantages: to gain in the execution time and to compute the solutions of the VIEs for long interval of times preserving the accuracy.

The paper is structured as follows. Sections 2 and 3 are devoted to the model formulation and the description of the implemented numerical method. After a brief presentation of the Nyström method introduced in [6], a new iterative implementation of it is here proposed. In Section 4 we explain the structure of the software by describing the MATLAB functions while in Section 5 several demos serve as examples of the use of the software. We conclude the paper with the Conclusions Section 6.

2. Introduction to the model formulation

2.1. The 1D model

We consider the following one-dimensional size-structured model proposed by Iwata et al. (see [8]) to describe the evolution of the colony size distribution of metastatic tumors

$$\begin{cases} \dfrac{\partial}{\partial t}\rho(t,x) + \dfrac{\partial}{\partial x}[g(x)\rho(t,x)] = 0, & t \geq 0,\ x \in [x_0, b), \\ g(x_0)\rho(t,x_0) = \beta_p(x_p(t)) + \displaystyle\int_{x_0}^{b} \beta_m(v)\rho(t,x)\,dx, & t \in (0,+\infty), \\ \rho(0,x) = 0, & x \in [x_0, b), \end{cases} \tag{1}$$

where x represents the size (number of cells or volume) of the tumor at time t, $\rho(t,x)$ is the unknown metastatic density, x_0 is the volume of the newly created metastases, b is the volume of the tumor at the saturated level, and β_p, β_m are the rates of emission of new metastases by the primary tumor and the metastases, respectively, depending on their size. We assume that $x_p(t)$, which is the volume of the primary tumor, will grow at rate $g_p(x)$ per unit time, *i.e.*, it is the solution of the following Cauchy problem

$$\begin{cases} \dfrac{dx_p(t)}{dt} = g_p(x_p(t)), \quad t \geq 0, \\ x_p(0) = x_{p,0}, \end{cases} \tag{2}$$

while the volume $x(t)$ of a metastasis emitted at time $t = 0$ is the solution of

$$\begin{cases} \dfrac{dx(t)}{dt} = g(x(t)), \quad t \geq 0, \\ x(0) = x_0. \end{cases} \tag{3}$$

We remark that the model covers the scenario in which primary and secondary tumors could follow different growth dynamics as well as they could emit new metastases with different rates. Moreover, also the case when the new born metastases is formed by clusters of several cells is contemplated (for more details, see [5] and the references therein).

For model (1), we are interested in computing the biological observable that can be expressed as a weighted integral of the metastatic density $\rho(t,x)$

$$F_\psi(t) = \int_{x_0}^{b} \psi(x)\rho(t,x)\,dx. \tag{4}$$

Main examples of observables are the total metastatic mass (TMM), $M(t)$, and the cumulative number of metastases (CNM) whose volume is larger than a certain size $\bar{x}$, $N_{\bar{x}}(t)$, at time t. They can be represented as in (4) with $\psi(x) = x$ and $\psi(x) = \chi_{x\geq\bar{x}}(x)$ ($\chi_{x\geq\bar{x}}$ denotes the characteristic function of the interval $[\bar{x},+\infty)$), respectively.

Following an idea in [10], the problem of computing the observable $F_\psi(t)$ given in (4), can be reformulated in terms of the VIE

$$F_\psi(t) - \int_0^t \beta_m(x(t-s))F_\psi(s)\,ds = \int_0^t \psi(x(t-s))\beta_p(x_p(s))\,ds. \tag{5}$$

2.2. The 2D model

The 1D model (1), introduced by Iwata et al., was extended by Benzekry (see [9,11]) which introduced a 2D metastatic model including the angiogenic process in the tumoural growth and an eventual chemotherapy/antiangiogenic treatment. It consists of the following two-dimensional PDE

$$\begin{cases} \dfrac{\partial}{\partial t}\rho(t,X) + \operatorname{div}[G(t,X)\rho(t,X)] = 0, & (t,X)\in(0,T)\times\Omega \\ -G(t,\sigma)\cdot\nu(\sigma)\rho(t,\sigma) = \mathcal{N}(\sigma)\beta(X_p(t)) + \mathcal{N}(\sigma)\displaystyle\int_\Omega \beta(X)\rho(t,X)\,dX, & (t,\sigma)\in(0,T)\times\partial\Omega, \\ \rho(0,X) = 0, & X\in\Omega, \end{cases} \tag{6}$$

where $X=(x,\theta)$ is a structuring variable with two components, with x the size (number of cells or volume) and θ the angiogenic capacity, $\rho(t,X)$ is the density of the population of metastases, $G(t,X) = \big(G_1(t,x,\theta), G_2(t,x,\theta)\big)$ represents the growth rate of the primary tumor $X_p(t)$ and of the metastatic one. Moreover, in (6) T is a positive time, the domain is assumed to be $\Omega=[1,b]\times[1,b]$, with $b>1$ the maximal reachable size and angiogenic capacity, and ν represent the outward unit normal vector at the boundary $\partial\Omega$. The function $\mathcal{N}:\partial\Omega\to\mathbb{R}$ represents the distribution of the size and carrying capacity along the boundary, while $\beta:\Omega\to\mathbb{R}$ the emission rate of new mestastes both from the primary and from the secondary tumor. Following [9], we assume $\mathcal{N}(\sigma)$ the Dirac measure in $(1,\theta_0)$, assuming 1 ($\simeq 10^{-6}\text{mm}^3$) and θ_0 the number of cells and the vasculature of a metastasis at birth, respectively, and, according to [8], we consider $\beta(X)=\mu x^\alpha$, with μ and α two parameters of metastatic emission.

In order to take into account the effect of both cytotoxic (CT) and anti-angiogenic (AA) therapies, the growth of the primary and secondary tumor is modeled by the subsequent ODE system developed in [12] endowed with initial conditions

$$\begin{cases} \dfrac{dX(t)}{dt} = G(t,X(t)), \quad t\geq 0, \\ X(0) = (x_0,\theta_0), \end{cases} \tag{7}$$

with $X(t)=(x(t),\theta(t))$ and

$$G_1(t,x,\theta) = ax\ln\left(\frac{\theta}{x}\right) - h\gamma_C(t)\left(x-x_{\min}\right)^+, \tag{8}$$

$$G_2(t,x,\theta) = cx - d\theta x^{\frac{2}{3}} - e\gamma_A(t)\left(\theta-\theta_{\min}\right)^+, \tag{9}$$

where $x_{\min}$ and $\theta_{\min}$ are minimal values for the drug to be effective, h and e are efficacy parameters of the CT and AA drugs, respectively, $\gamma_C(t)$ and $\gamma_A(t)$, describing the time evolution of the effective concentration of the drugs on their respective targets, are taken as, [9]

$$\gamma_C(t) = D_C\sum_{i=1}^{n} e^{-clr_C(t-t_i^C)}H\left(t-t_i^C\right), \tag{10}$$

$$\gamma_A(t) = D_A\sum_{i=1}^{n} e^{-clr_A(t-t_i^A)}H\left(t-t_i^A\right), \tag{11}$$

with D_C, D_A the administered doses, n the total number of administrations, clr_C, clr_A the clearances, t_i^C, t_i^A, $i=1,\dots,n$ the administration times of the CT and AA drugs, and the function H the Heaviside function.

We will consider two biological observables to be computed: the TMM, $M(t)$, and the CNM, $N_{\bar{x}}(t)$, greater than a certain threshold $\bar{x}$. Their expression are given by

$$M(t) = \iint_\Omega x\rho(t,x,\theta)\,dx\,d\theta, \tag{12}$$

$$N_{\bar{x}}(t) = \iint_{\Omega} \chi_{x\geq\bar{x}}(x)\rho(t,x,\theta)\,dx\,d\theta. \tag{13}$$

In particular, when $\bar{x} = 1\,\text{cell} \simeq 10^{-6}\text{mm}^3$, by (13) we have the total number of metastases

$$N(t) = \iint_{\Omega} \rho(t,x,\theta)\,dx\,d\theta. \tag{14}$$

In a recent work [6], following an idea proposed in [10], it is shown that the problem to determine a biological observable

$$F_{\psi}(t) = \iint_{\Omega} \psi(X)\rho(t,X)\,dX = \iint_{\Omega} \psi(x,\theta)\rho(t,x,\theta)\,dx\,d\theta, \tag{15}$$

with $\rho(t,X)$ the metastatic density, solution of the PDE model (6), can be reformulated as the problem of solving a Volterra integral equation.

More precisely, the biological observable $F_{\psi}(t)$ can be computed using the formula

$$F_{\psi}(t) = \int_0^t \int_{\partial\Omega} \psi(X(t;s,\sigma))\mathcal{N}(\sigma)\left[F_{\beta}(s) + \beta(X_p(s))\right]\,d\sigma\,ds$$

which, taking into account that $\mathcal{N}(\sigma)$ has been chosen equal to the Dirac measure in $(1,\theta_0)$, becomes

$$F_{\psi}(t) = \int_0^t \psi(X(t;s,(1,\theta_0)))\left[F_{\beta}(s) + \beta(X_p(s))\right]\,ds, \tag{16}$$

where $X(t;s,\sigma)$ is a characteristic curve, *i.e.*, the solution of the following system of ODE

$$\begin{cases} X'(t;s,\sigma) = G\,(t, X(t;s,\sigma)) \\ X(s;s,\sigma) = \sigma, \end{cases} \tag{17}$$

and the observable $F_{\beta}(t)$ solves the subsequent VIE of the second kind

$$F_{\beta}(t) - \int_0^t K(s,t)F_{\beta}(s)\,ds = g(t), \tag{18}$$

with the kernel $K(s,t)$ and the right-hand side $g(t)$ given by

$$K(s,t) = \beta(X(t;s,(1,\theta_0))), \tag{19}$$

and

$$g(t) = \int_0^t \beta(X(t;s,(1,\theta_0)))\beta(X_p(s))\,ds. \tag{20}$$

Finally, we remark that in the case without treatment (*i.e.*, when the tumor size x and the carrying capacity θ grow obeying the laws (8) and (9) with $h = e = 0$), the observable $F_{\psi}(t)$ given in (15), can be more efficiently computed by solving directly the subsequent VIE of the second kind (see [10])

$$F_{\psi}(t) - \int_0^t \beta(X(t-s;(1,\theta_0)))F_{\psi}(s)\,ds = \int_0^t \psi(X(s;(1,\theta_0)))\beta(X_p(t-s))\,ds, \tag{21}$$

where $X(t;\sigma)$ is a characteristic curve, *i.e.*, the solution of the following system of ODE

$$\begin{cases} X'(t;\sigma) = G\,(X(t;\sigma))\,, \\ X(0;\sigma) = \sigma, \end{cases} \tag{22}$$

with

$$G(X) = \left(G_1(x,\theta), G_2(x,\theta)\right) = \left(ax\ln\left(\frac{\theta}{x}\right), cx - d\theta x^{\frac{2}{3}}\right).$$

3. The implemented numerical method

This section is devoted to a brief description of the implemented numerical method for the solution of the integral equations (5), (18), (21), and the consequent computation of the biological observables of interest.

3.1. An iterative Nyström method

The method we are going to describe is an iterative version of the Nyström type method proposed in [6] for a more general VIE of the second kind of the following type

$$f(t) - \int_0^t K(s,t)f(s)\,ds = g(t), \quad t > 0, \tag{23}$$

where $K(s,t)$ defined on $\Delta = \{(s,t) \mid 0 \le s \le t,\ t \in \mathbb{R}^+\}$ and $g(t)$ defined on $\mathbb{R}^+$ are known functions and $f(t)$ is the solution to be computed.

The new iterative approach aims to overcome the drawback of loss of accuracy which could occur when the previous procedure is used to approximate long-time solutions of Eq. (23). More precisely, assuming that $t_0 \in \mathbb{R}^+$ is an upper threshold under which a certain precision is guaranteed in the numerical approximation of the solution f, the objective is to compute the values $f(t)$ for $t > t_0$ preserving the same accuracy.

We start by recalling that the Nyström method in [6] consists in approximating the solution f of the VIE (23) by the solution f_m of the approximating equation

$$f_m(t) - \int_0^t L_{m+1}(K(\cdot,t)f_m, s)ds = g(t), \tag{24}$$

where $L_{m+1}(F,t)$ denotes the truncated Lagrange polynomial which interpolates the function F at the m zeros $z_{m,1}, z_{m,2}, \dots, z_{m,m}$, of the classical Laguerre polynomial of degree m, $p_m(w)$, being $w(x) = e^{-x}$, and at the additional point $4m$ (see [13–15]). More explicitly, denoted by $j = \min_{i=1,\dots,m}\{i \ :\ z_{m,i} \ge 4\vartheta m\}$ where $0 < \vartheta < 1$ is fixed, the truncated interpolating polynomial can be represented as follows

$$L_{m+1}(F,s) = \sum_{i=1}^{j} \ell_{m+1,i}(s)F(z_{m,i}), \tag{25}$$

where

$$\ell_{m+1,i}(s) = \lambda_{m,i}\frac{(4m-s)}{(4m-z_{m,i})}\sum_{k=0}^{m-1} p_k(w,z_{m,k})p_k(w,s), \tag{26}$$

with $\lambda_{m,i}$ the ith Christoffel number associated to the Laguerre weight w.

By multiplying both sides of Eq. (24) by a suitable weight function $u(t) = t^\gamma e^{-\frac{t}{2}}$, $\gamma \ge 0$, we get the equation

$$u(t)f_m(t) - u(t)\sum_{i=1}^{j} K(z_{m,i},t)\frac{(f_m u)(z_{m,i})}{u(z_{m,i})}c_i(t) = u(t)g(t), \tag{27}$$

where the so called modified moments $c_i(t)$, $i = 1, \dots, j$, are the following integrals

$$c_i(t) = \int_0^t \ell_{m+1,i}(s)ds.$$

They can be computed using the formula

$$c_i(t) = \int_0^t \ell_{m+1,i}(s)ds = \frac{\lambda_{m,i}}{4m - z_{m,i}}\sum_{k=0}^{m-1} p_k(w,z_{m,i})m_k(t)$$

with the coefficients $m_k(t)$, $k = 0, 1, \dots, m-1$, satisfying the following recurrence relation

$$m_k(t) = -\bar{\beta}_k p_{k-1}(w,t) + (4m - \bar{\alpha}_{k+1} - \bar{\beta}_k)p_k(w,t) + (4m - \bar{\alpha}_{k+1} - \bar{\beta}_{k+1})p_{k+1}(w,t) - \bar{\beta}_{k+1}p_{k+2}(w,t), \tag{28}$$

being $\bar{\alpha}_k$ and $\bar{\beta}_k$ the coefficients of the three-term recurrence relation for the Laguerre polynomials.

Collocating Eq. (27) at the zeros $z_{m,r}$, $r = 1, \dots, j$, we obtain the linear system

$$A_m\mathbf{a} = \mathbf{b}, \tag{29}$$

with the matrix A_m given by

$$\left[A_m\right]_{r,i} = \delta_{r,i} - K(z_{m,i},z_{m,r})\frac{u(z_{m,r})}{u(z_{m,k})}\,c_i(z_{m,r}), \quad r,i = 1,\dots,j,$$

$\mathbf{b} = (b_1,\dots,b_j)^T$, $b_r = u(z_{m,r})g(z_{m,r})$, $r = 1,\dots,j$, $\mathbf{a} = (a_1,\dots,a_j)^T$, $a_i = u(z_{m,i})f_m(z_{m,i})$, $i = 1,\dots,j$ and $\delta_{r,i}$ the Kronecker delta.

When linear system (29) is solved, the Nyström interpolating function

$$f_m(t) = g(t) + \sum_{i=1}^{j} K(z_{m,i},t)\frac{c_i(t)}{u(z_{m,i})}a_i \tag{30}$$

is computed.

Now, we are going to propose a new numerical iterative scheme which is based on the method described above. In what follows, we will adopt the following notation

$$f^{[k]}(t) = f(t), \qquad t \in (kt_0,(k+1)t_0], \qquad k = 0,1,2,\dots, \tag{31}$$

where $t_0 > 0$ is fixed and sufficiently small.

The first step of the iterative procedure consists in approximating the solution of Eq. (23) in the interval $(0,t_0]$, *i.e.*, in approximating the solution $f^{[0]}(t)$ of the equation

$$f^{[0]}(t) - \int_0^t K(s,t)f^{[0]}(s)\,ds = g^{[0]}(t), \qquad t \in (0,t_0], \tag{32}$$

with $g^{[0]}(t) = g(t)$, by the Nyström interpolant $f_m(t) = f_m^{[0]}(t)$ given in (30), with the coefficients $a_i = a_i^{[0]} = u(z_{m,i}) f_m^{[0]}(z_{m,i})$, $i = 1, \dots, j$, obtained by solving linear system (29).

In the kth iteration of the method, $k = 1, 2, \dots$, the aim is to approximate the function values $f^{[k]}(t)$ for $t \in (kt_0, (k+1)t_0]$, using the approximation $f_m^{[h]}(t)$ of $f^{[h]}(t)$ for $t \in (ht_0, (h+1)t_0]$, with $h = 0, 1, \dots, k-1$, provided by the previous iterations of the scheme.

Indeed, we can observe that, in the case when $t \in (kt_0, (k+1)t_0]$, we can rewrite (23) as follows

$$f^{[k]}(t) - \int_{kt_0}^{t} K(s,t) f^{[k]}(s)\, ds = g(t) + \sum_{h=0}^{k-1} \int_{ht_0}^{(h+1)t_0} K(s,t) f^{[h]}(s)\, ds, \qquad t \in (kt_0, (k+1)t_0]. \tag{33}$$

By introducing the change of variable $t = t + kt_0$ in Eq. (33), the change of the integration variable $s = s + kt_0$ in the integral on the left-hand side, and the change of the integration variable $s = s + ht_0$ in each integral

$$\int_{ht_0}^{(h+1)t_0} K(s,t) f^{[h]}(s)\, ds, \qquad h = 0, 1, \dots, k-1,$$

on the right-hand side, we get

$$f^{[k]}(t + kt_0) - \int_0^t K(s + kt_0, t + kt_0) f^{[k]}(s + kt_0)\, ds = g(t + kt_0) + \sum_{h=0}^{k-1} \int_0^{t_0} K(s + ht_0, t + kt_0) f^{[h]}(s + ht_0)\, ds, \tag{34}$$

with $t \in [0, t_0]$. Now, setting, for $s, t \in [0, t_0]$, $\bar{f}^{[k]}(s) = f^{[k]}(s + kt_0)$, $\bar{K}^{[k]}(s,t) = K(s + kt_0, t + kt_0)$, and

$$\bar{g}^{[k]}(t) = g(t + kt_0) + \sum_{h=0}^{k-1} \int_0^{t_0} K(s + ht_0, t + kt_0) f^{[h]}(s + ht_0)\, ds, \tag{35}$$

we can rewrite (34) as follows

$$\bar{f}^{[k]}(t) - \int_0^t \bar{K}^{[k]}(s,t) \bar{f}^{[k]}(s)\, ds = \bar{g}^{[k]}(t), \qquad t \in [0, t_0]. \tag{36}$$

Hence, Eq. (36) can be seen as an equation of type (23) defined on the interval $(0, t_0]$ in the unknown $\bar{f}^{[k]}(t)$, with kernel $\bar{K}^{[k]}(s,t)$ and right-hand side $\bar{g}^{[k]}(t)$. Consequently, we can apply the Nyström method (29)–(30) for approximating its solution $\bar{f}^{[k]}(t) = f^{[k]}(t + kt_0)$.

Nevertheless, since in order to construct the right-hand side $\mathbf{b}$ of the corresponding linear system we need to compute the values of the function $\bar{g}^{[k]}$ at the collocation knots, *i.e.*, $\bar{g}^{[k]}(z_{m,r})$, $r = 1, \dots, j$, and their analytical computation cannot be carried out, we use the following approximation of the integrals involved in formula (35)

$$\int_0^{t_0} K(s + ht_0, t + kt_0) \bar{f}^{[h]}(s)\, ds \simeq \int_0^{t_0} L_{m+1}(K(\cdot + ht_0, t + kt_0) \bar{f}^{[h]}, s)\, ds = \sum_{i=1}^{j} K(z_{m,i} + ht_0, t + kt_0) \frac{c_i(t_0)}{u(z_{m,i})} a_i^{[h]},$$

with the coefficients $a_i^{[h]} = u(z_{m,i}) f_m^{[h]}(z_{m,i})$, $i = 1, \dots, j$, determined in the previous iterations of the method.

3.2. The numerical computation of the observables

First, we observe that in the 1D case for the numerical evaluation of the biological observable $F_\psi(t)$ in (4) we simply apply the numerical method described in the previous subsection in order to solve the VIE (5). The same can be done in the 2D autonomous case by solving (21), as already observed at the end of Section 2.2. Instead, in the two-dimensional non-autonomous case we will approximate the observable in (16) by a discretization of it obtained by applying a truncated Gauss–Laguerre quadrature formula [16]. More precisely, in first place the variable change $s = te^{-z}$ is introduced into the integral in (16) obtaining

$$F_\psi(t) = \int_0^\infty \psi(X(t; te^{-z}, (1, \theta_0))) \left[F_\beta(te^{-z}) + \beta(X_p(te^{-z})) \right] e^{-z}\, dz. \tag{37}$$

Then, we compute the approximating function

$$F_{\psi,m}(t) = \sum_{i=1}^{j} \lambda_{m,i} \psi(X(t; te^{-z_{m,i}}, (1, \theta_0))) \left[F_\beta(te^{-z_{m,i}}) + \beta(X_p(te^{-z_{m,i}})) \right], \tag{38}$$

where the values $F_\beta(te^{-z_{m,i}})$, $i = 1, \dots, j$, are provided by the numerical solution of the VIE (18)–(20) by means of the iterative Nyström method described in Section 3.1.

3.3. Computational details

This section is dedicated to illustrate some further computational details concerning the numerical procedure we have proposed above.

In the VIE reformulation of both the metastatic t.g. models (1) and (6), the right-hand side function $g(t)$ is an integral of the type

$$g(t) = \int_0^t h(s,t)\, ds, \tag{39}$$

Table 1
Relative error for $M(t)$ and $N(t)$ for $t = 40$ obtained by applying the iterative Nyström method for different choice of the threshold t_0.

t_0	Relative error for $M(40)$	Relative error for $N(40)$
40	1.33e−03	1.52e−02
10	5.39e−07	1.92e−05
5	2.17e−08	7.62e−07

Table 2
Relative error for $M(t)$ and $N(t)$ for $t = 60$ obtained by applying the iterative Nyström method for different choice of the threshold t_0.

t_0	Relative error for $M(60)$	Relative error for $N(60)$
60	4.42e+01	4.08e+02
10	9.15e−06	4.19e−05
5	1.68e−06	4.28e−06

with $h(s,t) = \psi(x(t-s))\beta_p(x_p(s))$ in the 1D case, $h(s,t) = \beta(X(t;s,(x_0,\theta_0)))\beta(X_p(s))$ in the 2D non-autonomous case, and $h(s,t) = \psi(X(s;(1,\theta_0)))\beta(X_p(t-s))$ in the 2D autonomous model. Since we cannot compute it exactly, we approximate this integral by using the truncated Gauss–Laguerre formula introduced in Section 3.2, but with a greater number $M \gg m$ of nodes, getting the following approximation

$$g_M(t) = \sum_{i=1}^{J} \lambda_{M,i} h(te^{-z_{M,i}}, t), \tag{40}$$

where the index J has been found as stated by [5, (39)].

Finally, we remark that, in order to build linear system (29) as well as in order to construct the Nyström interpolant in (30), we need to solve the ODEs (2) and (3) in the 1D case, and (7) and (22) in the 2D case. When it is not possible to compute the exact solution, suitable MATLAB routines (as, for instance, `ode45` or `ode23t`) are used to compute a corresponding numerical solution.

3.4. Iterative vs non-iterative scheme: numerical results

In this subsection we would like to show the performance of the new iterative Nyström method in the computation of the biological observables $M(t)$ and $N(t)$ given in (12) and (14), respectively, in the case without treatment. In particular, we would like to highlights how the accuracy of the results improves as the number of iterations of the scheme increases. To this end, we have applied the method for decreasing thresholds t_0, starting from the value $t_0 = t$, which corresponds to the non-iterative Nyström method proposed in [6]. As one can see in Tables 1 and 2, choosing a smaller value of t_0, *i.e.*, making a greater number of iterations, allows us to reduce the numerical errors. This is especially advantageous when the observables need to be computed for long times.

4. MATLAB software: Metastatic Tumor Growth Modeling (MTGM) toolbox

The MATLAB software MTGM introduced in this paper is freely available on a GitHub repository at https://github.com/IuliaMartinaBulai/MTGM_Toolbox. The functionalities of the Toolbox are five and they can be selected by the user by means of the input parameter `kind` as follows:

0 to solve a Volterra Integral Equation;
1 to compute the CNM and the TMM for the 1D metastatic t.g. model assuming that the growth laws for the primary and secondary tumor are known analytically;
2 to compute the CNM and the TMM for the 1D metastatic t.g. model solving numerically the ODE describing the growth of the tumor;
3 to compute the CNM and the TMM for the 2D metastatic t.g. model with treatment solving numerically the 2D ODE non-autonomous system;
4 to compute the CNM and the TMM for the 2D metastatic t.g. model solving numerically the 2D ODE autonomous system.

Next, we will go into the details of the MATLAB functions of MTGM Toolbox while in Section 5 a practical use of the Toolbox will be shown via several demos.

In Figs. 1–3 the schematic trees of all the functions of the Toolbox are represented. On the leafs of the trees we have represented the MATLAB functions that are recalled by the routine at the root in sequential order. In total twenty new functions have been developed plus eleven different demos, for a better understanding of the Toolbox potentialities (see Section 5).

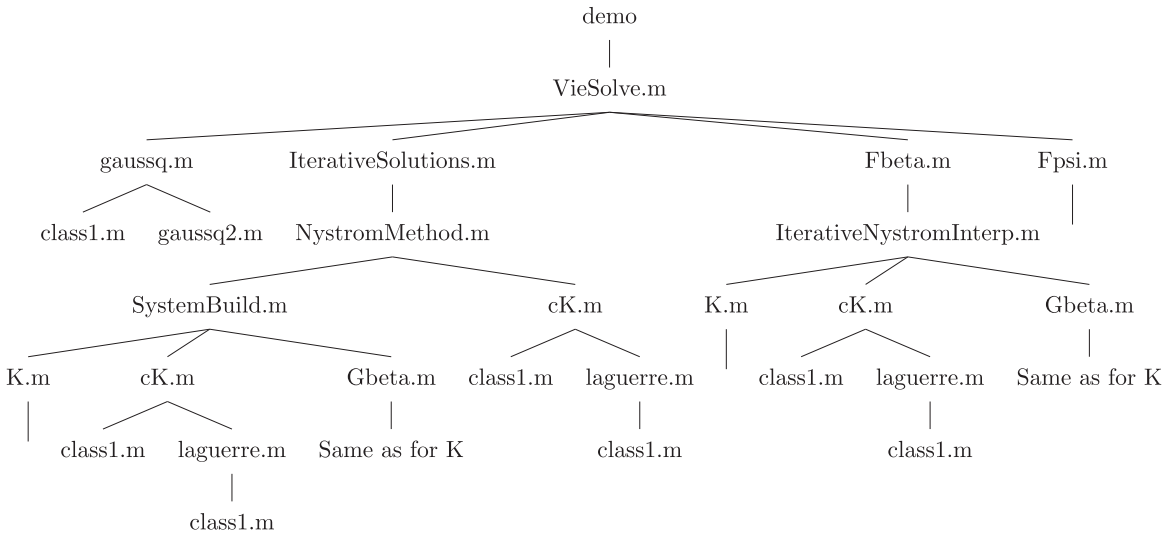

Fig. 1. Schematic tree of the MATLAB functions of the Toolbox MTGM. See Figs. 2 and 3 for the continuation regarding K.m and Fpsi.m recalls.

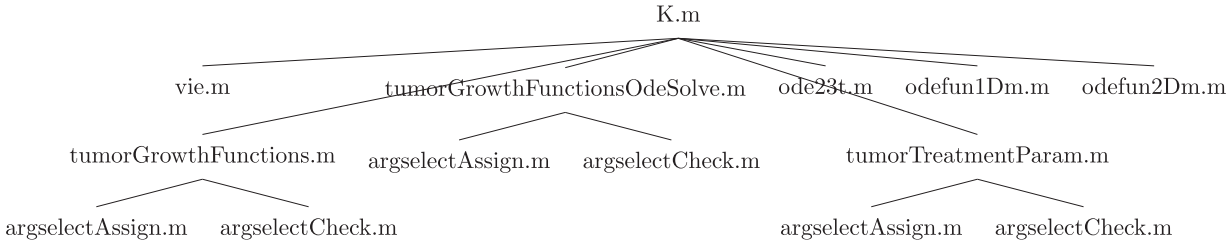

Fig. 2. Schematic subtree of the MATLAB function K.m.

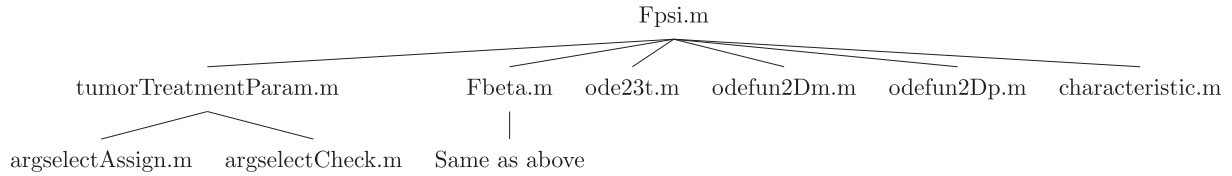

Fig. 3. Schematic subtree of the MATLAB function Fpsi.m. Notice that Fpsi.m is used only for the 2D non-autonomous model.

The first leaf of the tree in Fig. 1, regarding the demos, will be explained in detail in the next section, while the description of the other newly introduced functions is given below, starting with `VieSolve.m`:

```
%-------------------------------------------------------------------------------------
% File: VieSolve.m
%
% Goal: Compute the required solution of the Volterra Integral Equation, the dimension of the
%       solved linear system and its condition number
%
% Use: [M,N,j,C] = VieSolve(kind,T0,t,m,type_OB,varargin)
%
% Input: kind - can assume values:
%       - 0 - to solve a Volterra Integral Equation
%       - 1 - to compute the cumulative number of metastases and the total metastatic mass for
%             the 1D metastatic tumor growth model assuming that the growth laws for the primary
%             and secondary tumor are known analytically;
%       - 2 - to compute the cumulative number of metastases and the total metastatic mass for
```

```
%            the 1D metastatic tumor growth model solving numerically the differential problem
%            describing the growth of the tumor;
%        - 3 - to compute the cumulative number of metastases and the total metastatic mass for
%             the 2D metastatic tumor growth model with treatment solving the 2D ODE
%             non-autonomous system;
%        - 4 - to compute the cumulative number of metastases and the total metastatic mass for
%             the 2D metastatic tumor growth model solving the 2D ODE autonomous system
%        T0 - step for iteration
%        t - row array of the evaluation points
%        m - number of knots
%        type_OB - can assume value:
%                - 0 - for kind = 0 and kind = 3
%                - 1 - to compute the volume of the metastatic mass
%                - 2 - to compute the cumulative number of metastases
%        varargin - has different elements depending on the value of kind,
%                   for kind = 0 see vie.m file
%                   for kind = 1 see tumorGrowthFunctions.m
%                   for kind = 2 see tumorGrowthFunctionsOdeSolve.m
%                   for kind = 3 and kind = 4 see tumorTreatmentParam.m
%
% Output: M - 1xT array of the approximate values of:
%           - the solution of the VIE at the evaluation array t for kind = 0;
%           - the metastatic mass at t = [1:T] days for kind = 1,2,3,4
%         N - empty array for kind = 0;
%             1xT array of the approximate values of the cumulative number of metastases
%             whose volume is larger than vbar at t = [1:T] days for kind = 1,2,3,4
%         j - dimension of the solved linear system
%         C - condition number of the solved linear system
%
% Recalls: gaussq.m, IterativeSolutions.m, Fbeta.m, Fpsi.m
%-------------------------------------------------------------------------------------------
```

The function `VieSolve.m` recalls `gaussq.m`, `IterativeSolutions.m`, `Fbeta.m` and `Fpsi.m`. The first one was already introduced in VIE-Toolbox [7], while `IterativeSolutions.m` computes the solution of the linear system obtained applying the Nyström method (see Section 3 for more details) along with the condition number of the coefficient matrix. The above cited quantities are computed at the given iteration step and recalling as many times as needed the functions `NystromMethod.m`. The functions `IterativeSolutions.m` and `NystromMethod.m` are implemented as described below:

```
%-------------------------------------------------------------------------------------------
% File: IterativeSolutions.m
%
% Goal: Compute the solutions of the linear systems and their condition numbers at the
%       iteration step iter
%
% Use: [a,C] = IterativeSolutions(kind,T0,T,m,x,w,w1,E1,U,type_OB,varargin)
%
% Input: kind - see VieSolve.m file
%        T0 - step for iteration
%        T - greatest evaluation point
%        m - number of knots
%        x - row array of the zeros of the m-th Laguerre polynomial
%        w - row array of the corresponding Christoffel numbers
%        w1 - weights of the M-point Gauss-Laguerre rule with with M = 2048
%        E1 - array exp(-x1), with x1 array of the zeros of the M-th Laguerre polynomial
%        U - u(x), i.e weight function u computed ad the Laguerre zeros x
%        type_OB - see VieSolve.m
%        varargin - see VieSolve.m
%
% Output: a - column array of length 2*j*(iter+1) whose entries are the solutions of the
%         linear systems solved at the iteration step iter
%         C - condition number of the last solved linear system
%
% Recalls: NystromMethod.m
%-------------------------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------------------------
% File: NystromMethod.m
%
% Goal: Compute the solution of the linear system and its condition number
%
% Use: [a,C] = NystromMethod(kind,T0,iter,a,m,x,w,w1,E1,U,type_OB,varargin)
%
% Input: kind - see VieSolve.m file
%        T0 - step for iteration
%        iter - number of iterations
%        a - column array of length 2*j*iter whose entries are the solutions of the
%           linear systems solved at the iteration step iter
%        m - number of knots
%        x - row array of the zeros of the m-th Laguerre polynomial
%        w - row array of the corresponding Christoffel numbers
%        w1 - weights of the M-point Gauss-Laguerre rule with M = 2048
%        E1 - array exp(-x1), with x1 array of the zeros of the M-th Laguerre polynomial
%        U - u(x), i.e weight function u computed ad the Laguerre zeros x
%        type_OB - see VieSolve.m
%        varargin - see VieSolve.m
%
% Output: a - column array of length 2*j*(iter+1) whose entries are all the solutions of
%            the linear systems solved at the iteration step iter+1
%         C - condition number of the solved linear system
%
% Recalls: cK.m, SystemBuild.m
%-------------------------------------------------------------------------------------------
```

In order to apply the Nyström method, the so called modified moments $c_i(t)$, $i = 1, \dots, j$, must be computed and this can be done exactly by using the routine `cK.m` as below:

```
%-------------------------------------------------------------------------------------------
% File: cK.m
%
% Goal: Exact computation of the integrals
%         t
%  c_k(t) =  | l_{n+1,k}(z) dz,  k = 1,...,j
%        0
%
% Use: [c] = cK(t,j,x,w,U)
%
% Input: t - row array of evaluation points
%        j - size of the solved linear system
%        x - row array of the zeros of the n-th Laguerre polynomial
%        w - row array of the corresponding Christoffel numbers
%        U - u(x), i.e weight function u computed at the Laguerre zeros x
%
% Output: c - matrix (c_k(t_i))_{i = 1,...,length(t), k = 1,...,j}
%
% Recalls: class1.m, laguerre.m
%-------------------------------------------------------------------------------------------
```

In turn `cK.m` will recall `class1.m` and `laguerre.m` which computes the sequence of the orthonormal Laguerre polynomials by recurrence.

So as to build the matrix and the right-hand side term of the linear system solved in the function `NystromMethod.m`, the function `SystemBuild.m` is recalled. The latter needs `cK.m`, already introduced above, `K.m` and `Gbeta.m` which compute the kernel and the right-hand side of the VIE, respectively. In the sequel `SystemBuild.m`, `K.m` and `Gbeta.m` will be introduced:

```
%-------------------------------------------------------------------------------------------
% File: SystemBuild.m
%
% Goal: build the matrix and the right-hand side terms of the linear systems
%
% Use: [A,b] = SystemBuild(kind,T0,iter,a,x,w,w1,E1,U,c,type_OB,varargin)
%
% Input: kind - see VieSolve.m file
%        T0 - step for iteration
%        iter - number of iterations
%        a - solution of the linear system
%        x - row array of the zeros of the m-th Laguerre polynomial with m the number of knots
%        w - row array of the corresponding Christoffel numbers
%        w1 - weights of the M-point Gauss-Laguerre rule with with M = 2048
%        E1 - array exp(-x1), with x1 array of the zeros of the M-th Laguerre polynomial
%        U - u(x), i.e weight function u computed ad the Laguerre zeros x
%        c - matrix (c_k(t_i))_{i = 1,...,length(t), k = 1,...,j}
%        type_OB - see VieSolve.m
%        varargin - see VieSolve.m
%
% Output: A - matrix of the linear system
%         b - right-hand side of the linear system
%
% Recalls: K.m, cK.m, Gbeta.m
%-------------------------------------------------------------------------------------------

%-------------------------------------------------------------------------------------------
% File: K.m
%
% Goal: compute the kernel of the VIE
%
% Use: [Kt] = K(kind,x1,x2,varargin)
%
% Input: kind - see VieSolve.m file
%        x1 - row array of the evaluation points
%        x2 - row array of the evaluation points
%        varargin - see VieSolve.m
%
% Output: Kt - matrix K(x1(1:j),x2(1:j)')
%
% Recalls: vie.m, tumorGrowthFunctions.m, tumorGrowthFunctionsOdeSolve.m, ode23t.m,
%          odefun1Dm.m, tumorTreatmentParam.m, odefun2Dm.m
%-------------------------------------------------------------------------------------------

%-------------------------------------------------------------------------------------------
% File: Gbeta.m
%
% Goal: compute the right-hand side of the VIE
%
% Use: [gtbeta] = Gbeta(kind,t,w1,E1,type_OB,varargin)
%
% Input: kind - see VieSolve.m file
%        t - row array of the evaluation points
%        w1 - weights of the M-point Gauss-Laguerre rule with with M = 2048
%        E1 - array exp(-x1), with x1 array ofthe zeros of the M-th Laguerre polynomial
%             zeros of the M-th Laguerre polynomial
%        type_OB - see VieSolve.m
%        varargin - see VieSolve.m
%
% Output: gtbeta - array of the right-hand side of the VIE at the evaluation points t
%
% Recalls: vie.m, tumorGrowthFunctions.m, tumorGrowthFunctionsOdeSolve.m, ode23t.m,
%          odefun1Dm .m, odefun1Dp.m tumorTreatmentParam.m, odefun2Dm.m, odefun2Dp.m
%-------------------------------------------------------------------------------------------
```

In K.m and Gbeta.m the same functions are recalled, *i.e.*, the four routines which stores the data for the five case studies introduced above: vie.m for the VIE resolution; tumorGrowthFunctions.m for the computation of $M(t)$ and $N(t)$ in the 1D model with known analytical solutions for the ODE equations; tumorGrowthFunctionsOdeSolve.m for the computation of $M(t)$ and $N(t)$ in the 1D model where the numerical solutions for the ODE equations must be computed; tumorTreatmentParam.m for the computation of $M(t)$ and $N(t)$ in the 2D autonomous and non-autonomous cases.

K.m and Gbeta.m also recall the functions odefun1Dm.m and odefun2Dm.m where the ODEs modeling the growth of the metastases in the one-dimensional and two-dimensional cases, respectively, are defined. For the numerical resolution of the ODE the MATLAB routine ode23t is used. In Gbeta.m odefun1Dp.m and odefun2Dp.m are also recalled, which represent the growth laws of the primary tumor.

Now we describe vie.m, tumorGrowthFunctionsOdeSolve.m and tumorTreatmentParam.m in order. The function tumorGrowthFunctions.m contains a few changes with respect to the corresponding one introduced in [7], see the MATLAB code for more details.

```
%-------------------------------------------------------------------------------------------
% File: vie.m
%
% Goal: Define the analytical expressions of the kernel, the right-hand side term and
%       the weight function u of the weighted space, for the given VIE
%
% Use: [g] = vie()
%
% Input: -
% Output: g - 1X3 cell with:
%         g{1} kernel K(s,t) of the VIE
%         g{2} right-hand side of the VIE
%         g{3} weight function u of the weighted space
%
% Recalls: -
%-------------------------------------------------------------------------------------------

%-------------------------------------------------------------------------------------------
% File: tumorGrowthFunctionsOdeSolve.m
%
% Goal: Data for the primary and metastatic t.g.
%
% Use: [g,y0_m,y0_p,param,options] = tumorGrowthFunctionsOdeSolve(varargin)
%
% Input: varargin - 1X8 cell containing 4 couples:
%                   'emission_p',mu_p - colonization coefficient of the primary tumor
%                   'emission_m',mu_m - colonization coefficient of the metastases
%                   'v_p0', vp0 - volume of the primary tumor at time t = 0
%                   'v_m0', vm0 - volume of the newly created metastases
%         y0_p - [x0_p,theta0_p] initial conditions of the size of the tumor (x0_p) and
%                of the variable carrying capacity (theta0_p) for the primary tumor
%         y0_m - [x0_m,theta0_m] initial conditions of the size of the tumor (x0_m) and
%                of the variable carrying capacity (theta0_m) for the secondary tumor
%         param - [DA, clrA, DC, clrC, a, cc, d, e, h, xmin, thetamin, timesA, timesC]
%                parameters related to the treatment type
%         vbar - lower bound of the volume of the metastases whose cumulative number N
%                is computed
%         options - integration settings for the ODE solver
%
% Output: g - 1X3 cell with:
%         g{1} emission rate function for the primary tumor be_p
%         g{2} emission rate function for the metastases be_m
%         g{3} weight function u of the weighted space
%
% Recalls: argselectAssign.m, argselectCheck.m
%-------------------------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------------------------
% File: tumorTreatmentParam.m
%
% Goal: Create a database of possible tumor treatments
%
% Use: [g,y0_p,y0_m,param,vbar,options] = tumorTreatmentParam(varargin)
%
% Input: varargin - 1X12 cell containing 6 couples:
%                   'treatment_type', value1 - no_treat or type of treatment
%                   'emission_p',mu_p - colonization coefficient of the primary tumor
%                   'emission_m',mu_m  - colonization coefficient of the metastases
%                   'y0_p', [x0_p,theta0_p] - initial condition of the size of the tumor (x0_p)
%                          and of the carrying capacity (theta0_p) of the primary tumor
%                   'y0_m', [x0_m,theta0_m] - initial condition of the size of the tumor (x0_m)
%                          and of the carrying capacity  (theta0_m) of the secondary tumor
%                   'vbar', vbar - lower bound of the volume of the metastases whose
%                          cumulative number N is computed
%
% Output: g - 1X3 cell with:
%            g{1} emission rate function for the primary tumor be_p
%            g{2} emission rate function for the metastases be_m
%            g{3} weight function u of the weighted space
%         y0_p - [x0_p,theta0_p] initial conditions of the size of the tumor (x0_p) and
%               of the variable carrying capacity (theta0_p) for the primary tumor
%         y0_m - [x0_m,theta0_m] initial conditions of the size of the tumor (x0_m) and
%               of the variable carrying capacity (theta0_m) for the secondary tumor
%         param - [DA, clrA, DC, clrC, a, cc, d, e, h, xmin, thetamin, timesA, timesC]
%                parameters related to the treatment type
%         vbar - lower bound of the volume of the metastases whose cumulative number N is computed
%         options - integration settings for the ODE solver
%
% Recalls: argselectAssign.m, argselectCheck.m
%-------------------------------------------------------------------------------------------
```

The routines `tumorGrowthFunctions.m`, `tumorGrowthFunctionsOdeSolve.m` and `tumorTreatmentParam.m` all recall `argselectAssign.m` and `argselectCheck.m` already introduced in VIE-Toolbox, [7].

Going back on the third level of the tree in Fig. 1 `Fbeta.m` recalls `IterativeNystromInterp.m` which computes the Nyström interpolants and recalls the same routines as `SystemBuild.m`. Next, we introduce `Fbeta.m` and `IterativeNystromInterp.m`:

```
%-------------------------------------------------------------------------------------------
% File: Fbeta.m
%
% Goal: Compute Fbeta using the Nystrom interpolant
%
% Use: [fbeta] = Fbeta(kind,T0,t,a,x,w,w1,E1,U,type_OB,varargin)
%
% Input: kind - see VieSolve.m file
%        T0 - step for iteration
%        t - row array of the evaluation points
%        a - solution of the linear system
%        x - row array of the zeros of the m-th Laguerre polynomial with m the number of knots
%        w - row array of the corresponding Christoffel numbers
%        w1 - weights of the M-point Gauss-Laguerre rule with with M = 2048
%        E1 - array exp(-x1), with x1 array of the zeros of the M-th Laguerre polynomial
%        U - u(x), i.e weight function u computed ad the Laguerre zeros x
%        c - matrix (c_k(t_i))_{i = 1,...,length(t), k = 1,...,j}
%        type_OB - see VieSolve.m
%        varargin - see VieSolve.m
%
% Output: fbeta - array of the solution of the VIE at the evaluation points t
%
% Recalls: IterativeNystromInterp.m
%-------------------------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------------------------
% File: IterativeNystromInterp.m
%
% Goal: Compute the Nystrom interpolants, solutions of VIEs, for Fbeta at the iteration step
%       iter
%
% Use: [Fbeta] = IterativeNystromInterp(kind,T0,iter,t,a,x,w,w1,E1,U,type_OB,varargin)
%
% Input: kind - see VieSolve.m file
%        T0 - step for iteration
%        iter - number of iterations
%        t - row array of the evaluation points
%        a - solution of the linear system
%        x - row array of the zeros of the m-th Laguerre polynomial with m the number of knots
%        w - row array of the corresponding Christoffel numbers
%        w1 - weights of the M-point Gauss-Laguerre rule with with M = 2048
%        E1 - array exp(-x1), with x1 array of the zeros of the M-th Laguerre polynomial
%        U - u(x), i.e weight function u computed ad the Laguerre zeros x
%        type_OB - see VieSolve.m
%        varargin - see VieSolve.m
%
% Output: Fbeta - array of the solution of the VIE at the evaluation points t at the iteration
%          step iter
%
% Recalls: K.m, cK.m, Gbeta.m
%-------------------------------------------------------------------------------------------
```

Moreover, last, we consider `Fpsi.m` which computes the TMM, $M(t)$, and the CNM, $N(t)$, for the 2D non-autonomous case (`kind = 3`). The functions recalled by `Fpsi.m` have been already introduced before with the exception of `characteristic.m` which simply defines the characteristic function.

```
%-------------------------------------------------------------------------------------------
% File: Fpsi.m
%
% Goal: Compute the total metastatic mass, M, and the cumulative number of metastases, N, for
%       the 2D non-autonomous case
%
% Use: [M,N] = Fpsi(T0,t,a,x,w,w1,E,E1,U,varargin)
%
% Input: T0 - step for iteration
%        t - row array of the evaluation points
%        a - solution of the linear system
%        x - row array of the zeros of the m-th Laguerre polynomial with m the number of knots
%        w - row array of the corresponding Christoffel numbers
%        w1 - weights of the M-point Gauss-Laguerre rule with with M = 2048
%        E - array exp(-x)
%        E1 - array exp(-x1), with x1 array of the zeros of the M-th Laguerre polynomial
%        U - u(x), i.e weight function u computed ad the Laguerre zeros x
%        varargin - see VieSolve.m
%
% Output: M - 1xT array of the approximate values of the metastatic mass at t = [1:T] days
%         N - 1xT array of the approximate values of the cumulative number of metastases whose
%             volume is larger than vbar at t = [1:T] days
%
%
% Recalls: tumorTreatmentParam.m, Fbeta.m, ode23t.m, odefun2Dm .m, odefun2Dp.m,
%          characteristic.m
%-------------------------------------------------------------------------------------------
```

The Toolbox has a user friendly implementation, in fact choosing for example the `kind = 2` case study, the user should modify only `odefun1Dp.m` and `odefun1Dm.m`, routines, where the ODE models for the t.g. are defined and adjust accordingly `tumorGrowthFunctionsOdeSolve.m` where the parameters of the model are saved.

5. Demos

MATLAB R2022a was used to write the codes and for the simulations a personal computer with the specified features: Windows 11 Pro, Intel(R) Core(TM) i7-1065G7 CPU @ 1.30 GHz, 16 GB Memory, 512 GB SSD, was employed. In the Toolbox we have introduced eleven different demos in order to show its potentialities, *i.e.*,

- `mtgm_demo0`: This demo computes the weighted solution of a VIE.
- `mtgm_demo1`: This demo computes the TMM, $M(t)$, and the CNM, $N(t)$, for breast tumor data assuming that both the primary and secondary tumors growth laws are Gompertz laws. Moreover, for this 1D model the analytical solution of the growth laws is known.
- `mtgm_demo2`: Same as `mtgm_demo1` with the difference that here the analytical solution of the ODE equation is assumed to be unknown thus it is solved numerically. The results can be compared with those from `mtgm_demo1`.
- `mtgm_demo3`: This demo computes the CNM, $N(t)$, and the TMM, $M(t)$, for a 2D non-autonomous metastatic t.g. model (without considering treatment). The relative error corresponding to this simulation is also computed.
- `mtgm_demo4`: Same as for `mtgm_demo3` but assuming also that three different kinds of therapy are administered. For the sake of brevity no relative error is computed here.
- `mtgm_demo5`: This demo computes the CNM, $N(t)$, and the TMM, $M(t)$, for a 2D autonomous metastatic t.g. model. The results can be compared with those from `mtgm_demo3`.
- `mtgm_demo6`: This demo computes the CNM, $N(t)$, and the TMM, $M(t)$, for a 2D non-autonomous metastatic t.g. model considering only one type of treatment and dose but administered with different schedules.
- `mtgm_demo7`: This demo computes the CNM, $N(t)$, and the TMM, $M(t)$, for a 2D non-autonomous metastatic t.g. model considering only one type of treatment, same administration schedule but different doses.
- `mtgm_demo8`: This demo computes the CNM, $N(t)$, and the TMM, $M(t)$, for a 2D non-autonomous metastatic t.g. model combining two different treatments.
- `mtgm_demo9`: Same as for `mtgm_demo4` but for $T = 360$ days instead of $T = 15$ days.
- `mtgm_demo10`: Same as for `mtgm_demo3` but for $T = 360$ days instead of $T = 40$ days.

For the sake of brevity here we will show only the results of `mtgm_demo1`, `mtgm_demo4` and `mtgm_demo5`. For more details see the MTGM Toolbox.

Remark 5.1. Notice that the results of `mtgm_demo1` and `mtgm_demo2` are the same, in fact in both cases is assumed that the primary and secondary tumors grow following a Gompertz law. The difference lies in the fact that in the first case the analytical solution of the ODE is used while in the second case the ODE is solved numerically and, consequently, the numerical solution is used. Also `mtgm_demo3` and `mtgm_demo5` give the same output but solving two different models, the non-autonomous 2D model and the autonomous 2D model, respectively. For the non-autonomous 2D model, in `mtgm_demo3` we simulate the results without treatment taking $h = e = 0$ in the ODE system (8)–(9). Anyway, we point out that in this cas the more suitable procedure is the one implemented in `mtgm_demo5`.

5.1. *Mtgm_demo1*

```
Welcome to MTGM demo #1
Compute the TMM, M(t), and the CNM, N(t)
Case study: breast
Compute the TMM, M(t)
Number of iterations

iter = 1

Iteration step

i = 1

Compute the CNM, N(t)
```

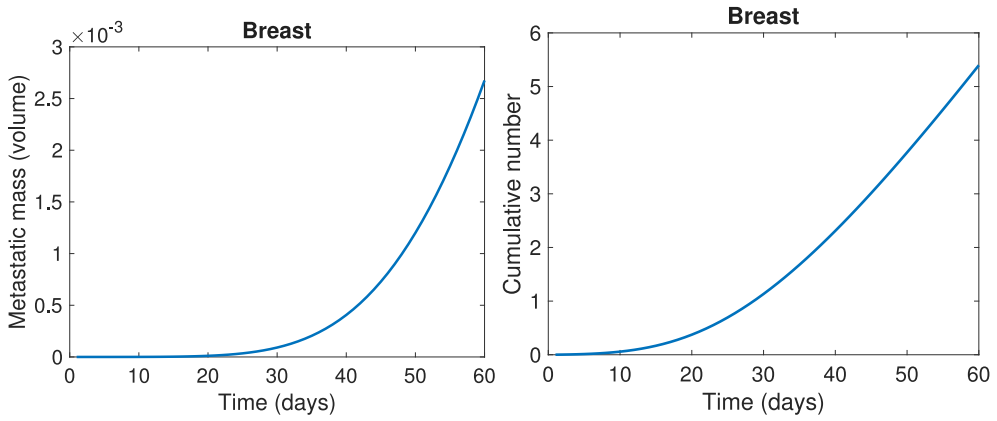

Fig. 4. Plots of `mtgm_demo1.m`. TMM in time (Left). CNM in time (Right).

```
Number of iterations

iter = 1

Iteration step

i = 1

Elapsed time is 12.225619 seconds.
Index j and condition number C for the TMM, M(t)

ans = 156.0000    1.0000

Index j and condition number C for the CNM, N(t)

ans = 156.0000    1.0000

Plot metastatic mass
Plot cumulative number of metastases
```

See Fig. 4 for the two output plots.

5.2. *Mtgm_demo4*

```
Welcome to MTGM demo #4
Compute the TMM, M(t), and the CNM, N(t)
2D non-autonomous model with treatment
No treatment
Number of iterations

iter = 1

Elapsed time is 81.339073 seconds.
 Index j and condition number C with No treatment

ans = 39.0000    1.3097

Endostatin 1
Number of iterations

iter = 1
```

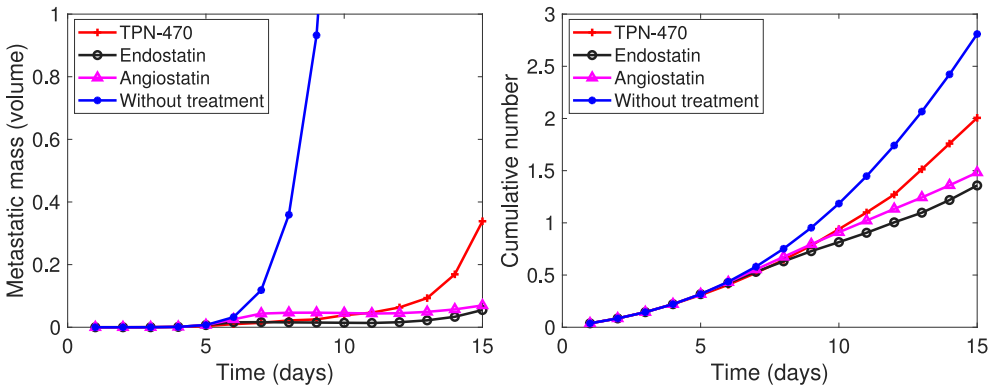

Fig. 5. Plots of `mtgm_demo4.m`. TMM in time (Left). CNM in time (Right).

```
Elapsed time is 216.697754 seconds.
 Index j and condition number C with Endostatin 1

ans = 39.0000    1.3098

TNP-470
Number of iterations

iter = 1

Elapsed time is 195.816115 seconds.
 Index j and condition number C with TNP-470

ans = 39.0000    1.3097

Angiostatin
Number of iterations

iter = 1

Elapsed time is 180.122657 seconds.
 Index j and condition number C with Angiostatin

ans = 39.0000    1.3093

Plot metastatic mass
Plot cumulative number of metastases
```

See Fig. 5 for the two output plots.

5.3. *Mtgm_demo5*

```
Welcome to MTGM demo #5
Compute the TMM, M(t), and the CNM, N(t)
2D autonomous model without treatment
Compute the TMM, M(t)
Number of iterations

iter = 1

Iteration step

i = 1
```

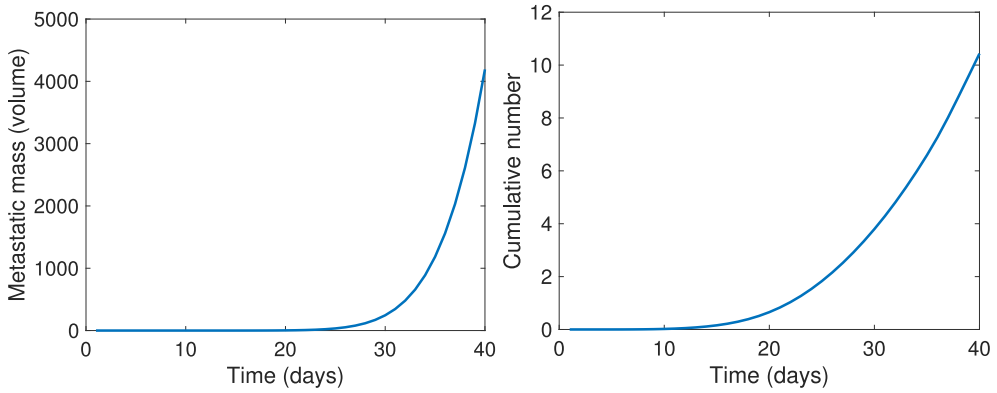

Fig. 6. Plots of `mtgm_demo5.m`. TMM in time (Left). CNM in time (Right).

```
Compute the CNM, N(t)
Number of iterations

iter = 1

Iteration step

i = 1

Elapsed time is 199.610399 seconds.
Index j and condition number C for the TMM, M(t)

ans = 39.0000   1.3097

Index j and condition number C for the CNM, N(t)

ans = 39.0000   1.3097

Plot metastatic mass
Plot cumulative number of metastases
```

See Fig. 6 for the two output plots.

6. Conclusions

In this paper we have introduced a new MATLAB software for numerical computation of biological observables derived from a metastatic t.g. model, *i.e.*, Metastatic Tumor Growth Modeling, MTGM Toolbox which is freely available on a GitHub repository https://github.com/IuliaMartinaBulai/MTGM_Toolbox. The Toolbox has five different functionalities which are all based on solving numerically a Volterra integral equation of the second type by applying a Nyström type method. Beside an extension of the `VIE_Toolbox`, already published in [7] for the 1D case (without treatment), the MTGM Toolbox proposed here considers also the more complex 2D non-autonomous case where the angiogenic process in the tumoural growth as well as the treatment of the tumor were taken in consideration.

The main application of the Toolbox is the numerical computation of two different biological observables, *i.e.*, the cumulative number of metastases and the metastatic burden, both derived from the reformulation of a coupled ODE-PDE model which describes the metastatic t.g.. Moreover, the effects of different types of treatment on the growth dynamics of the metastases were simulated.

The newly introduced Toolbox has two major improvements. On one side it has a faster execution time, on the other side it can be used to simulate cases of t.g. for long intervals of time without loosing accuracy in the numerical solution. Moreover, the Toolbox can be easily extended by the users by considering new models for the t.g. and/or new therapies.

CRediT authorship contribution statement

Iulia Martina Bulai: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Maria Carmela De Bonis:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision,

Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Concetta Laurita:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Acknowledgments

MCDB and CL have been supported by University of Basilicata (local funds) and by PRIN 2022 PNRR project AQuAInt Grant P20229RMLB, CUP-C53D23008400001, financed by the European Union - NextGeneration EU and by the Italian Ministry of University and Research (MUR). IMB has been supported by Mobilità Giovani Ricercatori 2022, and by the project TIGRECO funded by the Italian Ministry of University and Research (MUR), Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2022, Grant 20227TRY8H, CUP-J53D23003630001. This research has been accomplished within RITA (Research ITalian network on Approximation) and the UMI Group TAA (Approximation Theory and Applications).

References

[1] R. Alessandro, I. Gianní, F. Pes, T. Nottoli, F. Lipparini, Linear response equations revisited: A simple and efficient iterative algorithm, J. Chem. Theory Comput. 19 (2023).
[2] R. Cavoretto, A. De Rossi, A two-stage adaptive scheme based on RBF collocation for solving elliptic PDEs, Comput. Math. Appl. 79 (11) (2020) 3206–3222.
[3] G. Garmanjani, M. Esmaeilbeigi, Cavoretto, Adaptive residual refinement in an RBF finite difference scheme for 2D time-dependent problems, Comp. Appl. Math. (2024).
[4] A. Hernández-Inostroza, E. Gjini, MetaSpread: A cancer growth and metastatic spread simulation program in python, 2024, BioRxiv.
[5] I.M. Bulai, M.C. De Bonis, C. Laurita, V. Sagaria, Modeling metastatic tumor evolution, numerical resolution and growth prediction, Math. Comput. Simulation 203 (2023) 721–740.
[6] I.M. Bulai, M.C. De Bonis, C. Laurita, Numerical solution of metastatic tumor growth models with treatment, Appl. Math. Comput. 484 (2025) 128988.
[7] I.M. Bulai, M.C. De Bonis, C. Laurita, V. Sagaria, MatLab toolbox for the numerical solution of linear Volterra integral equations arising in metastatic tumor growth models, Dolomites Res. Notes Approx. 15 (2) (2022) 13–24.
[8] K. Iwata, K. Kawasaki, N. Shigesada, A dynamical model for the growth and size distribution of multiple metastatic tumors, J. Theoret. Biol. 203 (2000) 177–186.
[9] S. Benzekry, Mathematical analysis of a two-dimensional population model of metastatic growth including angiogenesis, J. Evol. Equ. 11 (2011) 187–213.
[10] N. Hartung, Efficient resolution of metastatic tumor growth models by reformulation into integral equations, Discret. Contin. Dyn. Syst. Ser. B. 20 (2) (2015) 445–467.
[11] S. Benzekry, Mathematical and numerical analysis of a model for anti-angiogenic therapy in metastatic cancers, ESAIM- Math. Model. Numer. 46 (2012) 207–237.
[12] P. Hahnfeldt, D. Panigraphy, J. Folkman, L. Hlatky, Tumor development under angiogenic signaling: a dynamical theory of tumor growth, treatment rensponse, and postvascular dormancy, Cancer Res. 59 (1999) 4770–4775.
[13] C. Laurita, G. Mastroianni, L^p-Convergence of Lagrange interpolation on the semiaxis, Acta Math. Hung. 120 (4) (2008) 249–273.
[14] M.C. De Bonis, B. Della Vecchia, G. Mastroianni, Approximation of the Hilbert transform on the real semiaxis using Laguerre zeros, J. Comput. Appl. Math. 140 (2002) 209–229.
[15] G. Mastroianni, D. Occorsio, Numerical approximation of weakly singular integrals on the half-line, J. Comput. Appl. Math. 140 (1–2) (2002) 587–598.
[16] G. Mastroianni, G. Monegato, Truncated Gauss-Laguerre quadrature rules, in: D. Trigiante (Ed.), in: Recent trends in Numerial Analysis, vol. 3, Nova Sci. Publ., Huntington, NY, 2001, pp. 213–221.