RESEARCH ARTICLE

"MoSpec": A customized and integrated system for model development, verification and validation

Marcello Pompa[1], **Simona Panunzi**[1]*, **Alessandro Borri**[1], **Laura D'Orsi**[1], **Andrea De Gaetano**[1,2,3]

1 Institute of Systems Analysis and Informatics "A. Ruberti" (IASI), National Research Council of Italy, Rome, Italy, **2** Institute for Biomedical Research and Innovation (IRIB), National Research Council of Italy, Palermo, Italy, **3** Dept. of Biomatics Óbuda University, Budapest, Hungary

* simona.panunzi@cnr.it, simona.panunzi@biomatematica.it

Abstract

Background and objective

The growing availability of patient data from several clinical settings, fueled by advanced analysis systems and new diagnostics, presents a unique opportunity. These data can be used to understand disease progression and predict future outcomes. However, analysing this vast amount of data requires collaboration between physicians and experts from diverse fields like mathematics and engineering.

Methods

Mathematical models play a crucial role in interpreting patient data and enable *in-silico* simulations for diagnosis and treatment. To facilitate the creation and sharing of such models, the CNR-IASI BioMatLab group developed the "Gemini" (MoSpec/Autocoder) system, a framework allowing researchers with basic mathematical knowledge to quickly and correctly translate biological problems into Ordinary Differential Equations models. The system facilitates the development and computation of mathematical models for the interpretation of medical and biological phenomena, also using data from the clinical setting or laboratory experiments for parameter estimation.

Results

Gemini automatically generates code in multiple languages (C++, Matlab, R, and Julia) and automatically creates documentation, including code, figures, and visualizations.

Conclusions

This user-friendly approach promotes model sharing and collaboration among researchers, besides vastly increasing group productivity.

OPEN ACCESS

Citation: Pompa M, Panunzi S, Borri A, D'Orsi L, De Gaetano A (2025) "MoSpec": A customized and integrated system for model development, verification and validation. PLoS ONE 20(1): e0316401. https://doi.org/10.1371/journal.pone.0316401

Editor: Prince Jain, Parul University Parul Institute of Technology, INDIA

Received: June 6, 2024

Accepted: December 10, 2024

Published: January 2, 2025

Data Availability Statement: Data are all contained within the paper.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Large volumes of data are these days collected and archived from patients in a variety of clinical settings. This phenomenon is the product of the evolution of analysis systems and the appearance of new diagnostic techniques [1–3]. The physician has thus in many cases observations of the clinical status of the patient at different points in time. With the advent of "omic" sciences, modern system of analysis and development of new computer languages, it is now possible to have a detailed picture of the patient condition at different stages of a possible disease. This abundance of clinical data represents an opportunity to satisfy two big needs: to understand the evolution of the disease, focusing on the causes, for diagnosis; and to provide a forecast of the evolution of the disease itself, for prognosis.

This rapid expansion of available data has determined the need for an interdisciplinary approach to their analysis, involving experts with different backgrounds (engineers, physicists, mathematicians, etc.).

The physician is thus no longer forced to reinterpret himself in order to conduct research in isolation, but can collaborate with other experts with emerging profiles, such as bio-mathematicians, bio-engineers, bio-informaticians.

In this context, the development of mathematical models describing patient pathophysiology plays an important and ever-expanding role in improving data interpretation for diagnosis, in optimizing existing protocols, in simulating scenarios for innovative medical treatments [3–12]. Modeling is at the basis of the possibility to conduct simulations, the so-called "*in-silico*" testing, which supplements biological "*in-vivo*" testing (*in-vitro*, *ex-vivo*, animal or patient experiments), abating costs and ethically reducing discomfort and suffering.

Implementing a mathematical model, however, requires the knowledge of a specific programming language and extensive coding. During this process, the risk of making errors, which involves spending time debugging and correcting the code, can be significant, depending on the complexity of the model and the level of expertise of the user. Software tools for simplifying mathematical model implementation and computation are described in the literature. Modelica [13], Simulink [14] and Comsol Multiphysics [15] are prominent examples for simulating physical systems. Modelica is an open-source platform from Linköping University and the Open Source Modelica Consortium. It provides a flexible framework for modeling diverse systems. Simulink is a commercial tool from MathWorks, offering a graphical environment for simulating dynamical systems. It includes a vast library of pre-built blocks and a user-friendly interface. Comsol Multiphysics is a commercial software excelling at coupling and solving multiphysics problems. It allows users to combine different physical phenomena for simulating complex real-world scenarios. Beyond these, Wolfram Alpha [16] is an online service that, while not a dedicated modeling tool, is a powerful computational knowledge engine aiding modeling tasks through its mathematical capabilities and the Wolfram Language.

In this context, the CNR-IASI BioMatLab group has developed a system, named "Gemini", but ordinarily referred to as "MoSpec/Autocoder", aimed at simplifying and speeding up mathematical model implementation. The system is designed for use by researchers with a basic knowledge of Ordinary Differential Equations and with the ability to translate into equations the problems at hand. While it has been developed in a biomedical context, and used to address mainly pathophysiological situations, the system is actually completely general and can be used in other domains without modifications. The motivation for the development and progressive improvement of the system stems from the need of a group of researchers, collaborating from several different geographical locations, at different levels of career progression and expertise, to synthetically and unequivocally sharing successive versions of the models they work upon, in an easy-to-understand format.

The system, consists of a combination of a so-called "MoSpec", a customized Model Specification spreadsheet, with an Autocoder (Automatic code generation) program, a pre-compiled Matlab code taking the MoSpec in input. It is crucial to emphasize that "Gemini—MoSpec/Autocoder" is an independent and distinct tool from Google's Gemini large language model. Our system is neither derived from nor aligned with the "Google—Gemini" model, it was not developed to achieve the same objectives, and our choice of the name Gemini actually predates the appearance of the Google product of the same name.

Unlike the commercial nature of most of the tools mentioned above, the MoSpec/Autocoder is a free system that does not require specific programming knowledge but only basic spreadsheet knowledge for compilation, though accuracy is essential.

The system "Gemini—MoSpec/Autocoder" allows the researcher to completely specify mixed Algebraic and Ordinary Differential Equations models in the MoSpec, then run the Autocoder on it, obtaining in output working simulation code in C++, Matlab, and R (a Julia output implementation is also under way). This multi-language model implementation originated the name "Gemini" ("twins"). The Autocoder outputs complete LaTeX code for model description, including tables and integrating supplied figures, which can be used as a self-standing ("Companion") report, or integrated into manuscripts intended for publication. The Autocoder also outputs compilable code for a so called "Visualizer", a model implementation with a Matlab GUI and a C++ computing engine used to quickly visualize the effect of parameter changes on the time-course of the state variables and produce graphs thereof. Finally, the Autocoder provides compilable code that can be directly inserted into a standard PHP Web-server implementation, to offer Machine-to-Machine (M2M) access to the model via a SOAP protocol.

The work outlines the procedure to upload any mathematical model, described according to the MoSpec specifications, onto the Gemini system to provide users with a MATLAB, R and Julia implementation of the model itself.

Theoretical framework

The present section provides a brief overview of the programming languages connected with the Gemini system, together with a brief discussion of the Verification and Validation procedures, which are of fundamental importance for asserting that the model is able to represent the phenomenon of interest.

Verification and validation

Verification and validation (V & V) are two distinct procedures used to check that certain systems, services or, more in general, products fulfill their intended purpose and satisfy requirements and specifications [17]. In our context the goal is that of assuring that a software model implementation correctly reproduces numerically the behavior of the pathophysiological system under investigation.

- **Verification**: may be defined as the process of assuring that a product, service, or system meets the needs or requirements of the users. In our context this means that the software implementation of a model works as specified in the mathematical description of the model itself, without bugs.
- **Validation**: is the process of evaluating whether or not a product, service, or system complies with regulations, requirements, specifications, or imposed conditions. It is typically performed by testing the system with real users to see if it meets their needs. Again, in our context this means that the implemented model must adhere to experimental observations, both

those which are already available and those which are to be obtained in the quest for model falsification according to Popper [18].

Boehm [19] characterised the difference between the two procedures in the following two statements:

1. Verification is building the system right.
2. Validation is building the right system.

According to [20] the final aims of the V & V process are therefore:

1. to detect bugs in the implementation and resolve them;
2. to detect problems or highlight important missing parts in the application.

Similarly, the U.S. Food and Drug Administration (FDA) guidelines report the following definition for validation and verification [21], respectively:

- "Software verification provides objective evidence . . . for consistency, completeness, and correctness of the software and its supporting documentation . . . and provides support for a subsequent conclusion that software is validated".
- "[validation entails] confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled." (in our context, that the model behaves according to physiology)

Notice in the FDA's definition of verification the emphasis placed on documentation: this is important to guarantee that developers and users understand and agree on what is being implemented. Appropriate, exhaustive and precise documentation makes it simpler to improve on the current model to address more complex issues or different application areas.

It is evident that V & V of (biomedical) model construction is a complex task that requires a variety of skills and techniques. As such, it is typically carried out by a multidisciplinary team of professionals, including mathematicians/statisticians, engineers, physiologists. The proposed *formalized system* is designed to support the Verification process. It proves to be a particularly useful tool in a development environment to allow collaborators, working on the same project, to communicate and to understand each other's contribution to the development. It could be imagined as a universal language for the work group, a structured method to develop new projects. The use of such a *formalized system* allows to perform the verification process in a standardized, predictable, robust way: if each new project is tackled in the same way, following the same instructions, and the same steps, then making substantial errors becomes more difficult. In other words, a *formalized system* leads the developers to follow strict rules in the development phase: due to its very nature, a standard operating procedure makes development less liable to possible implementation errors. Furthermore, error detection by co-workers becomes easier as the common standards and formats adopted simplify communication and collaboration. Specifically, the *formalized system* presented in this paper (named "MoSpec" and comprehensively outlined in the following subsections) was specifically designed to streamline and enhance the Verification process during the development and programming implementation of a mathematical model.

Besides facilitating verification through the use of standardized fields and automatic checks, the system specifically provides for a verification step by producing, in the "companion" LaTeX document, side-by-side expressions for each equation's right-hand-side (one in mathematical notation, one in computational notation): it thus becomes easier for the modeler to check that the (automatically produced) code captures exactly what the originally imagined equation was meant to express. In addition, the "Visualizer" tool, presented in the "Visualizer and C++ Implementation" subsection, also facilitates the Validation process, enabling users to employ experimental data to calibrate the parameters of a mathematical model and to assess whether the specified model is able to reproduce the expected behaviour.

Matlab, R, Julia, C++, HTML and PHP

After the theoretical construction of a mathematical model, which represents the phenomenon under study with a certain level of simplification, it is necessary to implement the model equations in a programming language to obtain simulations regarding the system behavior. In the following a brief description of the most commonly used programming languages is provided:

- **Matlab** is a proprietary programming language by MathWorks [22]. It is an interactive environment for handling variables, calculations, and data; its coding logic resembles C++, making it easier to convert MATLAB code to C++.
- **C++** is a high-level language extending C, often compiled for systems and embedded software [23]. It's known for its performance, efficiency, and flexibility.
- **R** is a free, high-level language for statistics and graphics [24]. It's easy to learn and powerful for tasks like data analysis, visualization, and modeling. R is highly customizable with a vast community of contributors.
- **Julia** [25] is a new, high-level language combining C/C++ speed with Python/R ease. It's great for numerical analysis and computational science. Julia is open-source, actively developed within a strong community.
- **HTML** is the foundation of web pages. It uses tags to define a document's structure and content for web browsers. These tags tell the browser how to display elements like headings, paragraphs, images, and more. [26].
- **PHP** is a scripting language for web development [27]. It runs on web servers, producing HTML, images, and more. PHP can perform system tasks like file creation and manipulation, making it efficient and flexible.

The BioMatLab "Gemini" (MoSpec/Autocoder) system

The MoSpec, is an OpenDocument Spreadsheet template developed at the CNR-IASI BioMatLab, implementing internal standard content for addressing the issues and needs presented in the "Theoretical framework" section. The MoSpec is the formalized document, which completely describes in pseudo code a mathematical model, and which is used in pair with the Autocoder, a BioMatLab proprietary Matlab script: the Autocoder takes in input the MoSpec and automatically produces the implementation of the mathematical model described there in different programming languages (Matlab, C++, R and Julia). The information included in the MoSpec (such as parameter and variable descriptions as well as the relative units of measurement) also allows the Autocoder to generate a complete LaTeX document, the "Companion", which reports the model equations, pairing their mathematical form with their "verbatim" computational definition, along with tables for all model variables and parameters. The system also allows the integration, in the final LaTeX document, of figures and diagrams as referred to in the MoSpec. The diagram block in Fig 1 shows the output of the MoSpec/Autocoder system.

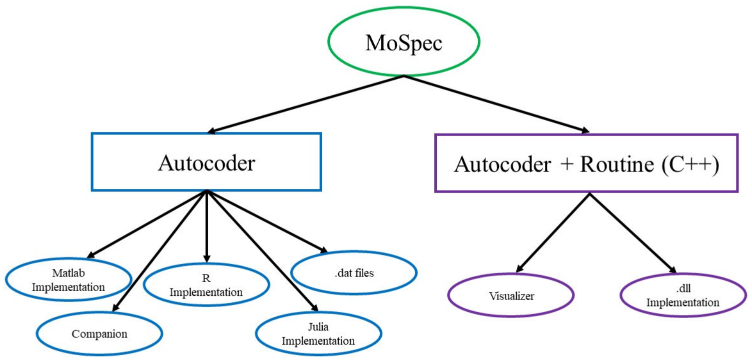


Fig 1. Block diagram of the MoSpec/Autocoder functioning.

https://doi.org/10.1371/journal.pone.0316401.g001

MoSpec

The MoSpec (Model Specification spreadsheet) is the place where the modeler specifies all information regarding the model: so much so that at the CNR IASI BioMatLab a given "model" is intended to be uniquely determined by the corresponding MoSpec, which is named according to the model type or application (the name), the model itself (version), the implementation (sub-version) and the edition of the describing MoSpec (sub-sub-version). So for instance, if in the class of models representing romantic love between people, model 2 contemplates two lovers exhibiting certain nonlinear dynamics, it has been iteratively corrected (hence with different computed results) four times and the corresponding MoSpec has been embellished and completed three times, the current version of the MoSpec is named "RomanticLove.MoSpec V02.04.03".

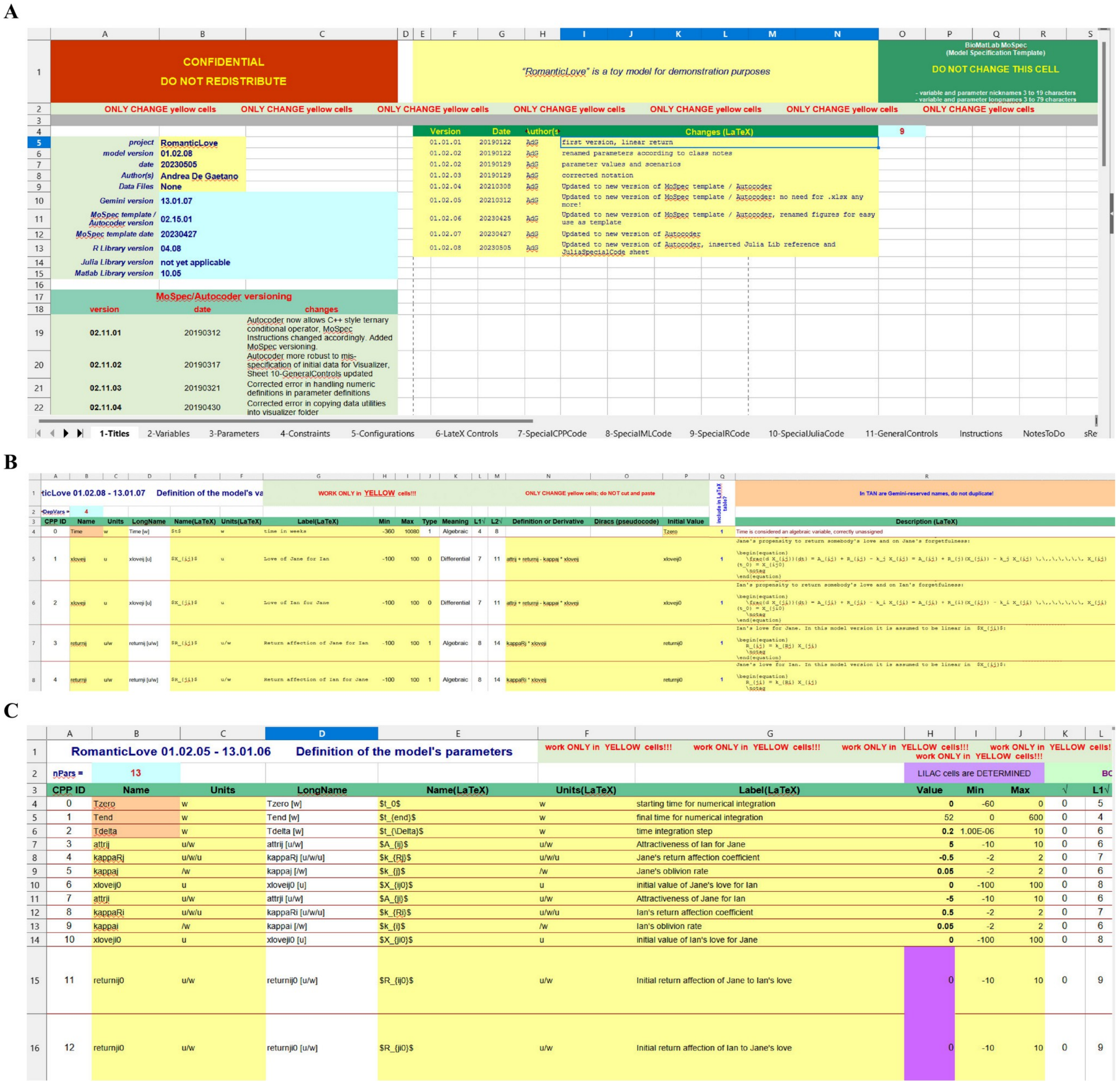
As mentioned above, the MoSpec is a ".ods" file (we use the free program LibreOffice [28], organized in 11 obligatory sheets, with additional "instructions" and "notes" sheets, and with zero or more "support" sheets helping the modeler to handle parameter values or conversions, or in general incorporating useful material. In the obligatory sheets, the cells which must be filled by the modeler have a yellow background: all other cells are reserved for system maintenance. The spreadsheet must be compiled in its entirety.

The first sheet ("Titles") of the MoSpec contains general information about the current project (Fig 2, panel A): name, dates, versions and changelog of any modification made during the project development.

The second ("Variables") sheet contains all the information regarding the state variables (Fig 2, panel B) including name, unit of measurement, description, range of variation, pseudo code defining the variable (either algebraically or differentially), initial conditions and the LaTeX code for the Companion.

The third sheet ("Parameters") has approximately the same structure of the "Variables" sheet and contains a description of the model parameters (Fig 2, panel C) accompanied by their reference value, admissible minima and maxima, etc. The two terms "free" or "determined" are here used to indicate those ("determined") parameters that are computed as function of other ("free"). To make the distinction between the two different types of parameters more immediately visible to the user the cells correspondent to the "determined" parameters are colored purple.

The fourth sheet "Constraints" (Fig 3) collects restrictions (if any) on the model parameters, one per line, which must be satisfied during the estimation or calibration procedure to obtain a coherent model prediction.



A

CONFIDENTIAL

DO NOT REDISTRIBUTE

"RomanticLove" is a toy model for demonstration purposes

BioMatLab MoSpec
(Model Specification Template)

DO NOT CHANGE THIS CELL

- variable and parameter nicknames 3 to 19 characters
- variable and parameter longnames 3 to 79 characters

ONLY CHANGE yellow cells ONLY CHANGE yellow cells ONLY CHANGE yellow cells ONLY CHANGE yellow cells ONLY CHANGE yellow cells ONLY CHANGE yellow cells ONLY CHANGE yellow cells

project	RomanticLove
model version	01.02.08
date	20230505
Author(s)	Andrea De Gaetano
Data Files	None
Gemini version	13.01.07
MoSpec template / Autocoder version	02.15.01
MoSpec template date	20230427
R Library version	04.08
Julia Library version	not yet applicable
Matlab Library version	10.05

MoSpec/Autocoder versioning

version	date	changes
02.11.01	20190312	Autocoder now allows C++ style ternary conditional operator, MoSpec Instructions changed accordingly. Added MoSpec versioning.
02.11.02	20190317	Autocoder more robust to mis-specification of initial data for Visualizer, Sheet 10-GeneralControls updated
02.11.03	20190321	Corrected error in handling numeric definitions in parameter definitions
02.11.04	20190430	Corrected error in copying data utilities into visualizer folder

Version	Date	Author(s	Changes (LaTeX)
01.01.01	20190122	AdG	first version, linear return
01.02.02	20190122	AdG	renamed parameters according to class notes
01.02.02	20190129	AdG	parameter values and scenarios
01.02.03	20190129	AdG	corrected notation
01.02.04	20210308	AdG	Updated to new version of MoSpec template / Autocoder
01.02.05	20210312	AdG	Updated to new version of MoSpec template / Autocoder: no need for .xlsx any more!
01.02.06	20230425	AdG	Updated to new version of MoSpec template / Autocoder, renamed figures for easy use as template
01.02.07	20230427	AdG	Updated to new version of Autocoder
01.02.08	20230505	AdG	Updated to new version of Autocoder, inserted Julia Lib reference and JuliaSpecialCode sheet

9

1-Titles 2-Variables 3-Parameters 4-Constraints 5-Configurations 6-LateX Controls 7-SpecialCPPCode 8-SpecialMLCode 9-SpecialRCode 10-SpecialJuliaCode 11-GeneralControls Instructions NotesToDo sRe

B

ticLove 01.02.08 - 13.01.07 Definition of the model's va

WORK ONLY in YELLOW cells!!!

ONLY CHANGE yellow cells; do NOT cut and paste

In TAN are Gemini-reserved names, do not duplicate!

DepVars = 4

CPP ID	Name	Units	LongName	Name(LaTeX)	Units(LaTeX)	Label(LaTeX)	Min	Max	Type	Meaning	L1√	L2√	Definition or Derivative	Diracs (pseudocode)	Initial Value	include in LaTeX table?	Description (LaTeX)
0	Time	w	Time [w]	\$t\$	w	time in weeks	-360	10080	1	Algebraic	4	8			Tzero	1	Time is considered an algebraic variable, correctly unassigned
1	xloveij	u	xloveij [u]	\$X_{ij}\$	u	Love of Jane for Ian	-100	100	0	Differential	7	11	attrij + returnij - kappaj * xloveij		xloveij0	1	Jane's propensity to return somebody's love and on Jane's forgetfulness: \begin{equation} \frac{d X_{ij}}{dt} = A_{ij} + R_{ij} - k_j X_{ij} = A_{ij} + R_{j}(X_{ji}) - k_j X_{ij} \,\,,\,\,\,\,\, X_{ij}(t_0) = X_{ij0} \notag \end{equation}
2	xloveji	u	xloveji [u]	\$X_{ji}\$	u	Love of Ian for Jane	-100	100	0	Differential	7	11	attrji + returnji - kappai * xloveji		xloveji0	1	Ian's propensity to return somebody's love and on Ian's forgetfulness: \begin{equation} \frac{d X_{ji}}{dt} = A_{ji} + R_{ji} - k_i X_{ji} = A_{ji} + R_{i}(X_{ij}) - k_i X_{ji} \,\,,\,\,\,\,\, X_{ji}(t_0) = X_{ji0} \notag \end{equation}
3	returnij	u/w	returnij [u/w]	\$R_{ij}\$	u/w	Return affection of Jane for Ian	-100	100	1	Algebraic	8	14	kappaRj * xloveji		returnij0	1	Ian's love for Jane. In this model version it is assumed to be linear in \$X_{ji}\$: \begin{equation} R_{ij} = k_{Rj} X_{ji} \notag \end{equation}
4	returnji	u/w	returnji [u/w]	\$R_{ji}\$	u/w	Return affection of Ian for Jane	-100	100	1	Algebraic	8	14	kappaRi * xloveij		returnji0	1	Jane's love for Ian. In this model version it is assumed to be linear in \$X_{ij}\$: \begin{equation} R_{ji} = k_{Ri} X_{ij} \notag

C

RomanticLove 01.02.05 - 13.01.06 Definition of the model's parameters

work ONLY in YELLOW cells!!! work ONLY in YELLOW cells!!! work ONLY in YELLOW cells!!! work ONLY in YELLOW cells! work ONLY in YELLOW cells!!!

nPars = 13

LILAC cells are DETERMINED

BC

CPP ID	Name	Units	LongName	Name(LaTeX)	Units(LaTeX)	Label(LaTeX)	Value	Min	Max	√	L1√
0	Tzero	w	Tzero [w]	\$t_0\$	w	starting time for numerical integration	0	-60	0	0	5
1	Tend	w	Tend [w]	\$t_{end}\$	w	final time for numerical integration	52	0	600	0	4
2	Tdelta	w	Tdelta [w]	\$t_{\Delta}\$	w	time integration step	0.2	1.00E-06	10	0	6
3	attrij	u/w	attrij [u/w]	\$A_{ij}\$	u/w	Attractiveness of Ian for Jane	5	-10	10	0	6
4	kappaRj	u/w/u	kappaRj [u/w/u]	\$k_{Rj}\$	u/w/u	Jane's return affection coefficient	-0.5	-2	2	0	7
5	kappaj	/w	kappaj [/w]	\$k_{j}\$	/w	Jane's oblivion rate	0.05	-2	2	0	6
6	xloveij0	u	xloveij0 [u]	\$X_{ij0}\$	u	initial value of Jane's love for Ian	0	-100	100	0	8
7	attrji	u/w	attrji [u/w]	\$A_{ji}\$	u/w	Attractiveness of Jane for Ian	-5	-10	10	0	6
8	kappaRi	u/w/u	kappaRi [u/w/u]	\$k_{Ri}\$	u/w/u	Ian's return affection coefficient	0.5	-2	2	0	7
9	kappai	/w	kappai [/w]	\$k_{i}\$	/w	Ian's oblivion rate	0.05	-2	2	0	6
10	xloveji0	u	xloveji0 [u]	\$X_{ji0}\$	u	initial value of Ian's love for Jane	0	-100	100	0	8
11	returnij0	u/w	returnij0 [u/w]	\$R_{ij0}\$	u/w	Initial return affection of Jane to Ian's love	0	-10	10	0	9
12	returnji0	u/w	returnji0 [u/w]	\$R_{ji0}\$	u/w	Initial return affection of Ian to Jane's love	0	-10	10	0	9

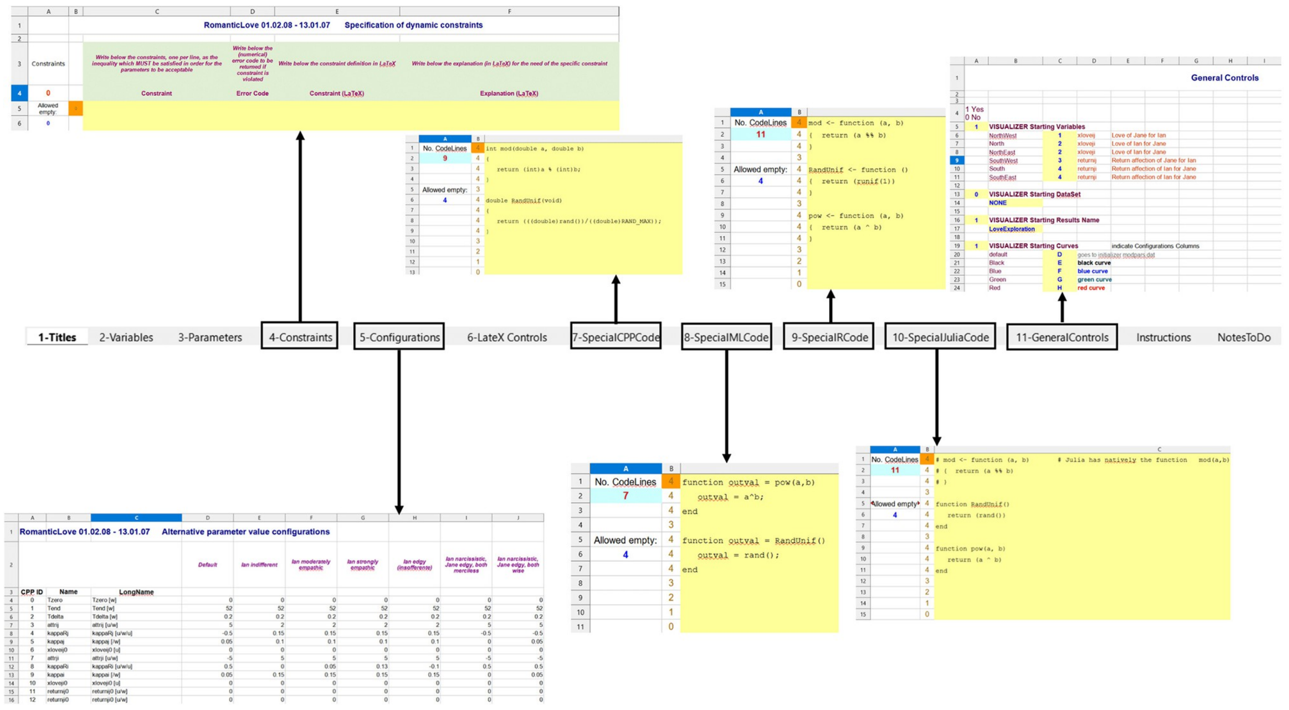
Fig 2. The main MoSpec sheets. (A) "1-Title" is the sheet reporting the general information of the current project; (B) "2-Variable" is the sheet including variable description; (C) "3-Parameters" is the sheet describing model parameters.

https://doi.org/10.1371/journal.pone.0316401.g002

The sheet allows to declare, for each parameter constraint, the error code (a number) to be shown in case the constraint is not satisfied, as well as corresponding LaTeX description of the constraint.

The fifth sheet "Configurations" (Fig 3) collects possible model parameter vector configurations to allow different scenarios (e.g. the "Default" scenario or the "Ian indifferent" scenario of the RomanticLove model in Fig 3).

The sixth sheet "LaTeX Controls" (Fig 4) collects a series of commands useful to modify tables (as for example the length of the table cells: "Tab length", "Name length", etc.) and figures (if any) for the generation of the "Companion" LaTeX file.



RomanticLove 01.02.08 - 13.01.07 Specification of dynamic constraints

Constraints		*Write below the constraints, one per line, as the inequality which MUST be satisfied in order for the parameters to be acceptable*	*Write below the (numerical) error code to be returned if constraint is violated*	*Write below the constraint definition in LaTeX*	*Write below the explanation (in LaTeX) for the need of the specific constraint*
0		**Constraint**	**Error Code**	**Constraint (LaTeX)**	**Explanation (LaTeX)**
Allowed empty:	0				
0					

No. CodeLines: 9; Allowed empty: 4

```
int mod(double a, double b)
{
   return (int)a % (int)b;
}

double RandUnif(void)
{
   return (((double)rand())/((double)RAND_MAX));
}
```

No. CodeLines: 11; Allowed empty: 4

```
mod <- function (a, b)
{  return (a %% b)
}

RandUnif <- function ()
{  return (runif(1))
}

pow <- function (a, b)
{  return (a ^ b)
}
```

General Controls

1 Yes 0 No				
1	**VISUALIZER Starting Variables**			
	NorthWest	1	xloveij	Love of Jane for Ian
	North	2	xloveji	Love of Ian for Jane
	NorthEast	2	xloveji	Love of Ian for Jane
	SouthWest	3	returnij	Return affection of Jane for Ian
	South	4	returnji	Return affection of Ian for Jane
	SouthEast	4	returnji	Return affection of Ian for Jane
0	**VISUALIZER Starting DataSet**			
	NONE			
1	**VISUALIZER Starting Results Name**			
	LoveExploration			
1	**VISUALIZER Starting Curves**			indicate Configurations Columns
	default	D	goes to initializer modpars.dat	
	Black	E	black curve	
	Blue	F	blue curve	
	Green	G	green curve	
	Red	H	red curve	

No. CodeLines: 7; Allowed empty: 4

```
function outval = pow(a,b)
   outval = a^b;
end

function outval = RandUnif()
   outval = rand();
end
```

No. CodeLines: 11; Allowed empty: 4

```
# mod <- function (a, b)        # Julia has natively the function   mod(a,b)
# {  return (a %% b)
# }

function RandUnif()
   return (rand())
end

function pow(a, b)
   return (a ^ b)
end
```

RomanticLove 01.02.08 - 13.01.07 Alternative parameter value configurations

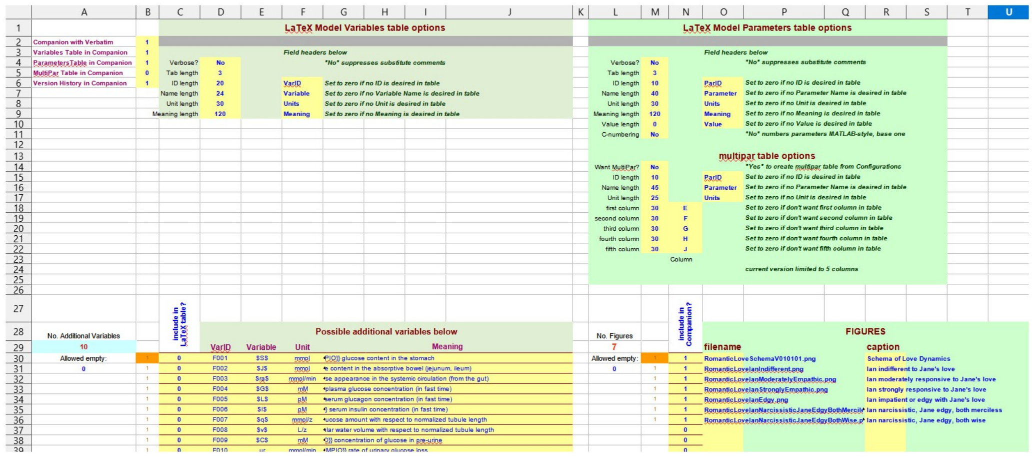
CPP ID	Name	LongName	*Default*	*Ian indifferent*	*Ian moderately empathic*	*Ian strongly empathic*	*Ian edgy (insofferente)*	*Ian narcissistic, Jane edgy, both merciless*	*Ian narcissistic, Jane edgy, both wise*
0	Tzero	Tzero [w]	0	0	0	0	0	0	0
1	Tend	Tend [w]	52	52	52	52	52	52	52
2	Tdelta	Tdelta [w]	0.2	0.2	0.2	0.2	0.2	0.2	0.2
3	attrij	attrij [u/w]	5	2	2	2	2	5	5
4	kappaRj	kappaRj [u/w/u]	-0.5	0.15	0.15	0.15	0.15	-0.5	-0.5
5	kappaj	kappaj [/w]	0.05	0.1	0.1	0.1	0.1	0	0.05
6	xloveij0	xloveij0 [u]	0	0	0	0	0	0	0
7	attrji	attrji [u/w]	-5	5	5	5	5	-5	-5
8	kappaRi	kappaRi [u/w/u]	0.5	0	0.05	0.13	-0.1	0.5	0.5
9	kappai	kappai [/w]	0.05	0.15	0.15	0.15	0.15	0	0.05
10	xloveji0	xloveji0 [u]	0	0	0	0	0	0	0
11	returnij0	returnij0 [u/w]	0	0	0	0	0	0	0
12	returnji0	returnji0 [u/w]	0	0	0	0	0	0	0

Fig 3. The MoSpec sheets.

https://doi.org/10.1371/journal.pone.0316401.g003

While the tables are automatically generated from the information contained in the "Variables" and "Parameters" sheets, the figures are included in the LaTeX file only if provided and their names indicated in the present sheet.

The seventh ("SpecialCPPCode"), eighth ("SpecialMLCode"), ninth ("SpecialRCode") and tenth ("SpecialJuliaCode") sheets (Fig 3) are used to declare in C++ code (sheet 7), in Matlab code (sheet 8), in R coe (sheet 9) and in Julia code (sheet 10) some specific functions useful for the model integration. In particular, since some frequently used functions are differently indicated in the four languages, an alternative, common form is used in the variable and parameter



Companion with Verbatim	1
Variables Table in Companion	1
ParametersTable in Companion	1
MultiPar Table in Companion	0
Version History in Companion	1

LaTeX Model Variables table options

		Field headers below	
Verbose?	No		*"No" suppresses substitute comments*
Tab length	3		
ID length	20	VarID	*Set to zero if no ID is desired in table*
Name length	24	Variable	*Set to zero if no Variable Name is desired in table*
Unit length	30	Units	*Set to zero if no Unit is desired in table*
Meaning length	120	Meaning	*Set to zero if no Meaning is desired in table*

LaTeX Model Parameters table options

		Field headers below	
Verbose?	No		*"No" suppresses substitute comments*
Tab length	3		
ID length	10	ParID	*Set to zero if no ID is desired in table*
Name length	40	Parameter	*Set to zero if no Parameter Name is desired in table*
Unit length	30	Units	*Set to zero if no Unit is desired in table*
Meaning length	120	Meaning	*Set to zero if no Meaning is desired in table*
Value length	0	Value	*Set to zero if no Value is desired in table*
C-numbering	No		*"No" numbers parameters MATLAB-style, base one*

multipar table options

Want MultiPar?	No			*"Yes" to create multipar table from Configurations*
ID length	10		ParID	*Set to zero if no ID is desired in table*
Name length	45		Parameter	*Set to zero if no Parameter Name is desired in table*
Unit length	25		Units	*Set to zero if no Unit is desired in table*
first column	30	E		*Set to zero if don't want first column in table*
second column	30	F		*Set to zero if don't want second column in table*
third column	30	G		*Set to zero if don't want third column in table*
fourth column	30	H		*Set to zero if don't want fourth column in table*
fifth column	30	J		*Set to zero if don't want fifth column in table*
		Column		
				current version limited to 5 columns

Possible additional variables below

No. Additional Variables		include in LaTeX table?	VarID	Variable	Unit	Meaning
10						
Allowed empty:	1	0	F001	\$S\$	mmol	PIO}} glucose content in the stomach
0	1	0	F002	\$J\$	mmol	e content in the absorptive bowel (jejunum, ileum)
	1	0	F003	\$ra\$	mmol/min	se appearance in the systemic circulation (from the gut)
	1	0	F004	\$G\$	mM	plasma glucose concentration (in fast time)
	1	0	F005	\$L\$	pM	serum glucagon concentration (in fast time)
	1	0	F006	\$I\$	pM	} serum insulin concentration (in fast time)
	1	0	F007	\$q\$	mmol/z	ucose amount with respect to normalized tubule length
	1	0	F008	\$v\$	L/z	lar water volume with respect to normalized tubule length
	1	0	F009	\$C\$	mM	O}} concentration of glucose in pre-urine
	1	0	F010	ur	mmol/min	MPIO}} rate of urinary glucose loss

FIGURES

No. Figures		include in Companion?	filename	caption
7				
Allowed empty:	1	1	RomanticLoveSchemaV010101.png	Schema of Love Dynamics
0	1	1	RomanticLoveIanIndifferent.png	Ian indifferent to Jane's love
	1	1	RomanticLoveIanModeratelyEmpathic.png	Ian moderately responsive to Jane's love
	1	1	RomanticLoveIanStronglyEmpathic.png	Ian strongly responsive to Jane's love
	1	1	RomanticLoveIanEdgy.png	Ian impatient or edgy with Jane's love
	1	1	RomanticLoveIanNarcissisticJaneEdgyBothMercile	Ian narcissistic, Jane edgy, both merciless
	1	1	RomanticLoveIanNarcissisticJaneEdgyBothWise.p	Ian narcissistic, Jane edgy, both wise
		0		
		0		
		0		

Fig 4. "LaTeX Controls" sheet.

https://doi.org/10.1371/journal.pone.0316401.g004

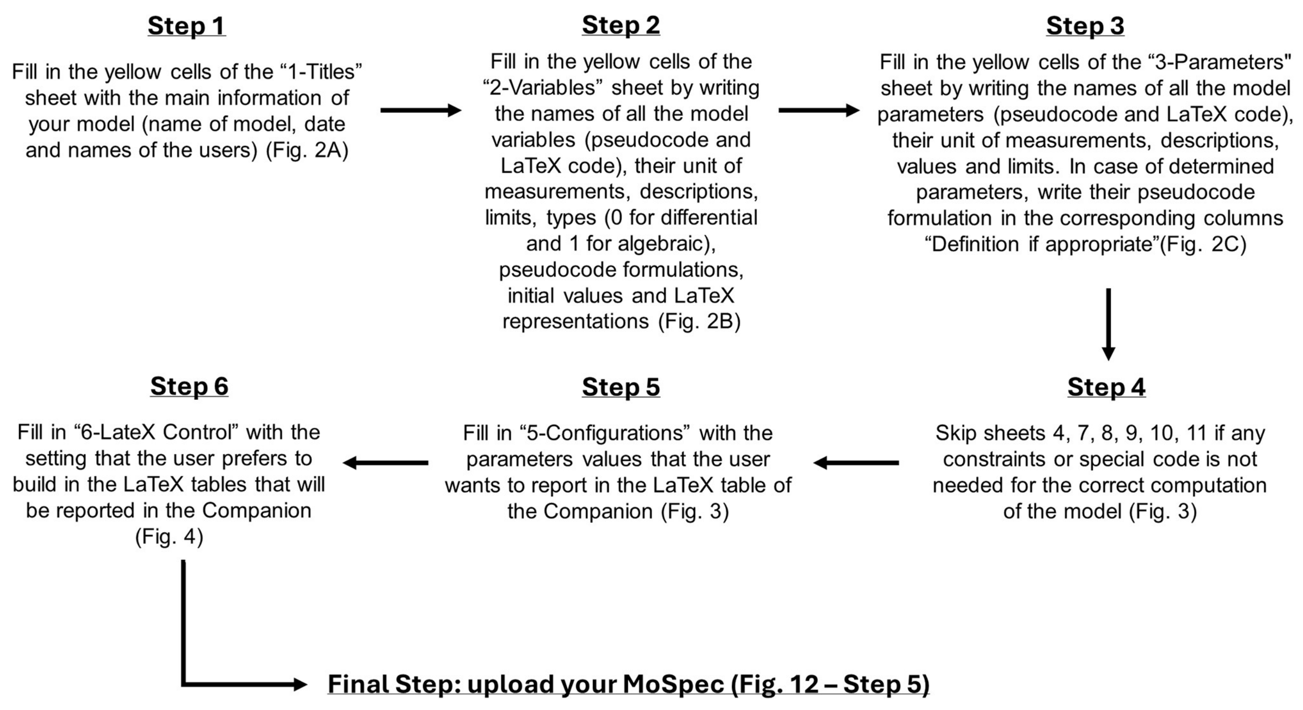

Step 1
Fill in the yellow cells of the "1-Titles" sheet with the main information of your model (name of model, date and names of the users) (Fig. 2A)
Step 2
Fill in the yellow cells of the "2-Variables" sheet by writing the names of all the model variables (pseudocode and LaTeX code), their unit of measurements, descriptions, limits, types (0 for differential and 1 for algebraic), pseudocode formulations, initial values and LaTeX representations (Fig. 2B)
Step 3
Fill in the yellow cells of the "3-Parameters" sheet by writing the names of all the model parameters (pseudocode and LaTeX code), their unit of measurements, descriptions, values and limits. In case of determined parameters, write their pseudocode formulation in the corresponding columns "Definition if appropriate"(Fig. 2C)
Step 4
Skip sheets 4, 7, 8, 9, 10, 11 if any constraints or special code is not needed for the correct computation of the model (Fig. 3)
Step 5
Fill in "5-Configurations" with the parameters values that the user wants to report in the LaTeX table of the Companion (Fig. 3)
Step 6
Fill in "6-LateX Control" with the setting that the user prefers to build in the LaTeX tables that will be reported in the Companion (Fig. 4)
Final Step: upload your MoSpec (Fig. 12 – Step 5)

Fig 5. Workflow for MoSpec sheet compilation and upload.

https://doi.org/10.1371/journal.pone.0316401.g005

definition pseudocode and this form is then defined for the four languages in their own syntax: as an example, the pseudocode "RandUnif()" function translates into the R "runif()" or the Matlab "Rand()" functions.

Sheet 11 "GeneralControls" (Fig 3) allows to define a series of options for the Visualizer tool, such as the model variables to be shown at the opening of the Visualizer (the number of whose graphical windows is limited), or the data sets to be loaded by default.

The MoSpec "Instructions" sheet is merely a collection of instructions for the correct compilation of the MoSpec itself, whereas the "NotesToDo" sheet includes communications to be shared among the users working on the project.

Fig 5 illustrates the workflow for correctly compiling the MoSpec sheets.

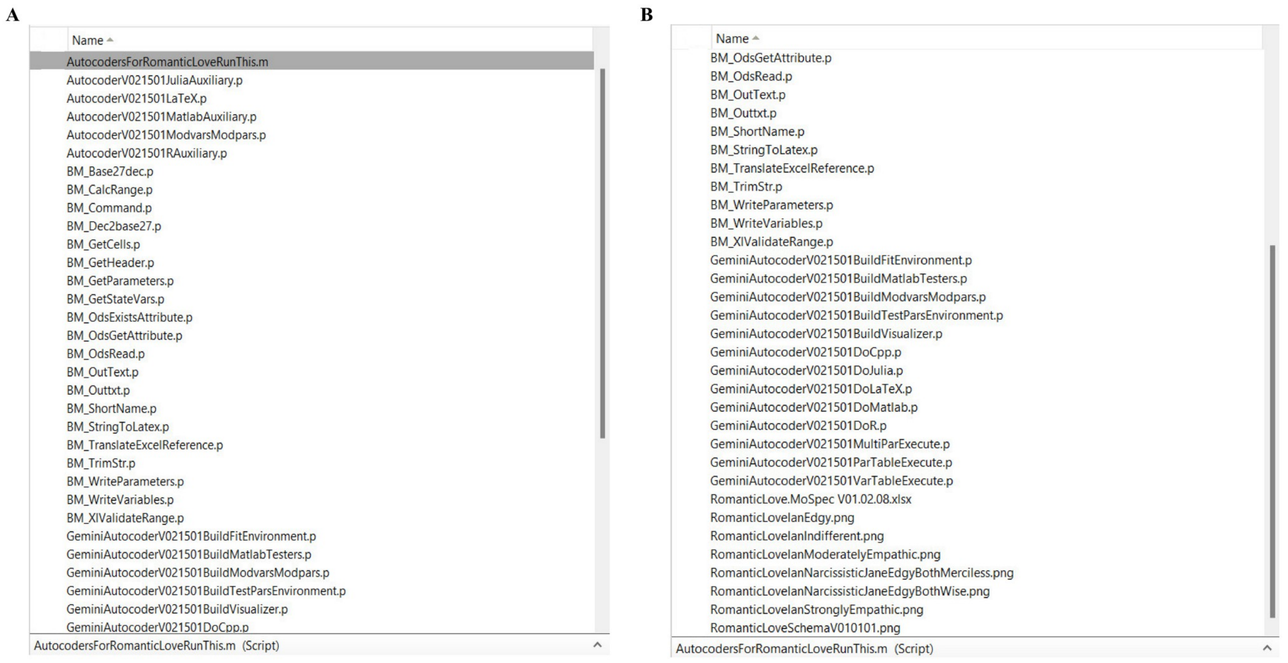
While the ".ods" MoSpec file is considered the single valid model specification, the Autocoder actually supports both ".ods" and ".xlsx" file formats.

Autocoder

The Autocoder is a Matlab program written at the BioMatLab for the interpretation of the information provided in the MoSpec (Fig 6).

While the input into the Autocoder is a valid MoSpec file, its output is a series of files in the diverse programming language for the simulation and visualization of the model described in the MoSpec.

Matlab, R and Julia. The execution of the Autocoder (which is itself a Matlab program) creates and populates separate folders containing all the necessary routines for the numerical integration of the model in each of the three languages (Matlab, R and Julia). The routines have the same names across languages and refer to similarly-named variables, to make switching context and debugging simpler. Files in Matlab, R and Julia code are shown in Figs 7–9, respectively.


A

```
Name
AutocodersForRomanticLoveRunThis.m
AutocoderV021501JuliaAuxiliary.p
AutocoderV021501LaTeX.p
AutocoderV021501MatlabAuxiliary.p
AutocoderV021501ModvarsModpars.p
AutocoderV021501RAuxiliary.p
BM_Base27dec.p
BM_CalcRange.p
BM_Command.p
BM_Dec2base27.p
BM_GetCells.p
BM_GetHeader.p
BM_GetParameters.p
BM_GetStateVars.p
BM_OdsExistsAttribute.p
BM_OdsGetAttribute.p
BM_OdsRead.p
BM_OutText.p
BM_Outtxt.p
BM_ShortName.p
BM_StringToLatex.p
BM_TranslateExcelReference.p
BM_TrimStr.p
BM_WriteParameters.p
BM_WriteVariables.p
BM_XlValidateRange.p
GeminiAutocoderV021501BuildFitEnvironment.p
GeminiAutocoderV021501BuildMatlabTesters.p
GeminiAutocoderV021501BuildModvarsModpars.p
GeminiAutocoderV021501BuildTestParsEnvironment.p
GeminiAutocoderV021501BuildVisualizer.p
GeminiAutocoderV021501DoCpp.p
AutocodersForRomanticLoveRunThis.m (Script)
```

B

```
Name
BM_OdsGetAttribute.p
BM_OdsRead.p
BM_OutText.p
BM_Outtxt.p
BM_ShortName.p
BM_StringToLatex.p
BM_TranslateExcelReference.p
BM_TrimStr.p
BM_WriteParameters.p
BM_WriteVariables.p
BM_XlValidateRange.p
GeminiAutocoderV021501BuildFitEnvironment.p
GeminiAutocoderV021501BuildMatlabTesters.p
GeminiAutocoderV021501BuildModvarsModpars.p
GeminiAutocoderV021501BuildTestParsEnvironment.p
GeminiAutocoderV021501BuildVisualizer.p
GeminiAutocoderV021501DoCpp.p
GeminiAutocoderV021501DoJulia.p
GeminiAutocoderV021501DoLaTeX.p
GeminiAutocoderV021501DoMatlab.p
GeminiAutocoderV021501DoR.p
GeminiAutocoderV021501MultiParExecute.p
GeminiAutocoderV021501ParTableExecute.p
GeminiAutocoderV021501VarTableExecute.p
RomanticLove.MoSpec V01.02.08.xlsx
RomanticLoveIanEdgy.png
RomanticLoveIanIndifferent.png
RomanticLoveIanModeratelyEmpathic.png
RomanticLoveIanNarcissisticJaneEdgyBothMerciless.png
RomanticLoveIanNarcissisticJaneEdgyBothWise.png
RomanticLoveIanStronglyEmpathic.png
RomanticLoveSchemaV010101.png
AutocodersForRomanticLoveRunThis.m (Script)
```

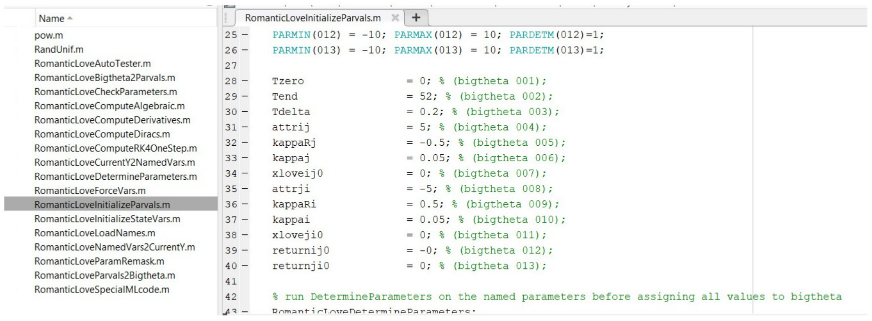
Fig 6. (A-B) Autocoder Matlab files generating model implementation in Matlab, R and Julia programming languages and producing the LaTeX companion file.

https://doi.org/10.1371/journal.pone.0316401.g006

For example, the Matlab implementation of the model is executed by running the "*Project-Name*AutoTester.m" file. The output of the Matlab code is generated in terms of plots of the model variable time courses. The numerical integration method used by the system is the fourth order, fixed-step Runge-Kutta.

C++. The Autocoder also produces model-specific source C++ code implementing the numerical integration of the model and referring to a BioMatLab library of standard mathematical functions (e.g Runge-Kutta integration, global optimization, datafile access,. . .) for eventual compilation within a Microsoft Visual Code C++ Community Edition 2021 environment. Such compiled C++ code takes two forms:

- it produces a self-standing ".exe" file callable from a Windows environment with a string specifying all necessary parameter values, returning a string containing all model predictions at the desired times; this ".exe" implementation of the model is the fast computational engine



```
Name
pow.m
RandUnif.m
RomanticLoveAutoTester.m
RomanticLoveBigtheta2Parvals.m
RomanticLoveCheckParameters.m
RomanticLoveComputeAlgebraic.m
RomanticLoveComputeDerivatives.m
RomanticLoveComputeDiracs.m
RomanticLoveComputeRK4OneStep.m
RomanticLoveCurrentY2NamedVars.m
RomanticLoveDetermineParameters.m
RomanticLoveForceVars.m
RomanticLoveInitializeParvals.m
RomanticLoveInitializeStateVars.m
RomanticLoveLoadNames.m
RomanticLoveNamedVars2CurrentY.m
RomanticLoveParamRemask.m
RomanticLoveParvals2Bigtheta.m
RomanticLoveSpecialMLcode.m
```

RomanticLoveInitializeParvals.m

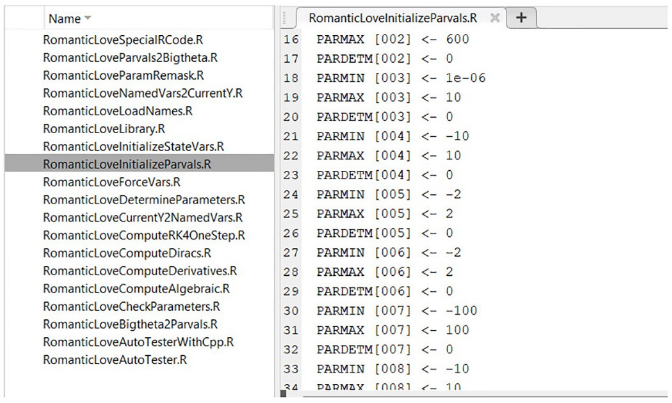
```
    PARMIN(012) = -10; PARMAX(012) = 10; PARDETM(012)=1;
    PARMIN(013) = -10; PARMAX(013) = 10; PARDETM(013)=1;

    Tzero                = 0; % (bigtheta 001);
    Tend                 = 52; % (bigtheta 002);
    Tdelta               = 0.2; % (bigtheta 003);
    attrij               = 5; % (bigtheta 004);
    kappaRj              = -0.5; % (bigtheta 005);
    kappaj               = 0.05; % (bigtheta 006);
    xloveij0             = 0; % (bigtheta 007);
    attrji               = -5; % (bigtheta 008);
    kappaRi              = 0.5; % (bigtheta 009);
    kappai               = 0.05; % (bigtheta 010);
    xloveji0             = 0; % (bigtheta 011);
    returnij0            = -0; % (bigtheta 012);
    returnji0            = 0; % (bigtheta 013);

    % run DetermineParameters on the named parameters before assigning all values to bigtheta
    RomanticLoveDetermineParameters;
```

Fig 7. Matlab model implementation files produced by the Autocoder.

https://doi.org/10.1371/journal.pone.0316401.g007



```
RomanticLoveSpecialRCode.R
RomanticLoveParvals2Bigtheta.R
RomanticLoveParamRemask.R
RomanticLoveNamedVars2CurrentY.R
RomanticLoveLoadNames.R
RomanticLoveLibrary.R
RomanticLoveInitializeStateVars.R
RomanticLoveInitializeParvals.R
RomanticLoveForceVars.R
RomanticLoveDetermineParameters.R
RomanticLoveCurrentY2NamedVars.R
RomanticLoveComputeRK4OneStep.R
RomanticLoveComputeDiracs.R
RomanticLoveComputeDerivatives.R
RomanticLoveComputeAlgebraic.R
RomanticLoveCheckParameters.R
RomanticLoveBigtheta2Parvals.R
RomanticLoveAutoTesterWithCpp.R
RomanticLoveAutoTester.R
```

```
PARMAX [002] <- 600
PARDETM[002] <- 0
PARMIN [003] <- 1e-06
PARMAX [003] <- 10
PARDETM[003] <- 0
PARMIN [004] <- -10
PARMAX [004] <- 10
PARDETM[004] <- 0
PARMIN [005] <- -2
PARMAX [005] <- 2
PARDETM[005] <- 0
PARMIN [006] <- -2
PARMAX [006] <- 2
PARDETM[006] <- 0
PARMIN [007] <- -100
PARMAX [007] <- 100
PARDETM[007] <- 0
PARMIN [008] <- -10
PARMAX [008] <- 10
```

Fig 8. R model implementation files produced by the Autocoder.

https://doi.org/10.1371/journal.pone.0316401.g008

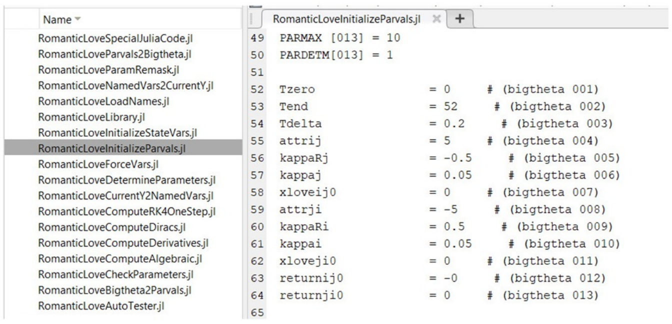
used to deliver model predictions via machine-to-machine PHP SOAP interface or through a human user interface (see below "Online Platform"). The Companion, in pdf format, can also be downloaded from the online platform.

- it produces the necessary compiled Dynamic Link Libraries (".dll") for the numerical integration of the model within the Visualizer model validation tool, see below.

Visualizer. The Autocoder produces the Matlab code supporting a graphical user interface for model prediction visualization: the "Visualizer" (Fig 10), as well as the (.csv format) supporting parameter and variable files (containing default and scenario values, acceptable limits and so on).

The interface is composed of different zones: the parameter section, which allows to modify the parameter values (defaults being automatically extracted from the MoSpec); the variable section, which allows to choose the state variables to visualize in the adjacent plot section; the control section, which presents different output options, one of which the possibility to add and visualize observed data (related to one or more state variables). This allows to visually calibrate the model parameters to explore different scenarios, or before an optimization procedure for parameter estimation is performed.

In this implementation, Matlab only deals with the interface allowing the user to change model parameters or specifying datasets, as well as returning forecast graphics. The model



```
RomanticLoveSpecialJuliaCode.jl
RomanticLoveParvals2Bigtheta.jl
RomanticLoveParamRemask.jl
RomanticLoveNamedVars2CurrentY.jl
RomanticLoveLoadNames.jl
RomanticLoveLibrary.jl
RomanticLoveInitializeStateVars.jl
RomanticLoveInitializeParvals.jl
RomanticLoveForceVars.jl
RomanticLoveDetermineParameters.jl
RomanticLoveCurrentY2NamedVars.jl
RomanticLoveComputeRK4OneStep.jl
RomanticLoveComputeDiracs.jl
RomanticLoveComputeDerivatives.jl
RomanticLoveComputeAlgebraic.jl
RomanticLoveCheckParameters.jl
RomanticLoveBigtheta2Parvals.jl
RomanticLoveAutoTester.jl
```

```
PARMAX [013] = 10
PARDETM[013] = 1

Tzero                    = 0     # (bigtheta 001)
Tend                     = 52     # (bigtheta 002)
Tdelta                   = 0.2     # (bigtheta 003)
attrij                   = 5     # (bigtheta 004)
kappaRj                  = -0.5     # (bigtheta 005)
kappaj                   = 0.05     # (bigtheta 006)
xloveij0                 = 0     # (bigtheta 007)
attrji                   = -5     # (bigtheta 008)
kappaRi                  = 0.5     # (bigtheta 009)
kappai                   = 0.05     # (bigtheta 010)
xloveji0                 = 0     # (bigtheta 011)
returnij0                = -0     # (bigtheta 012)
returnji0                = 0     # (bigtheta 013)

```

Fig 9. Julia model implementation files produced by the Autocoder.

https://doi.org/10.1371/journal.pone.0316401.g009

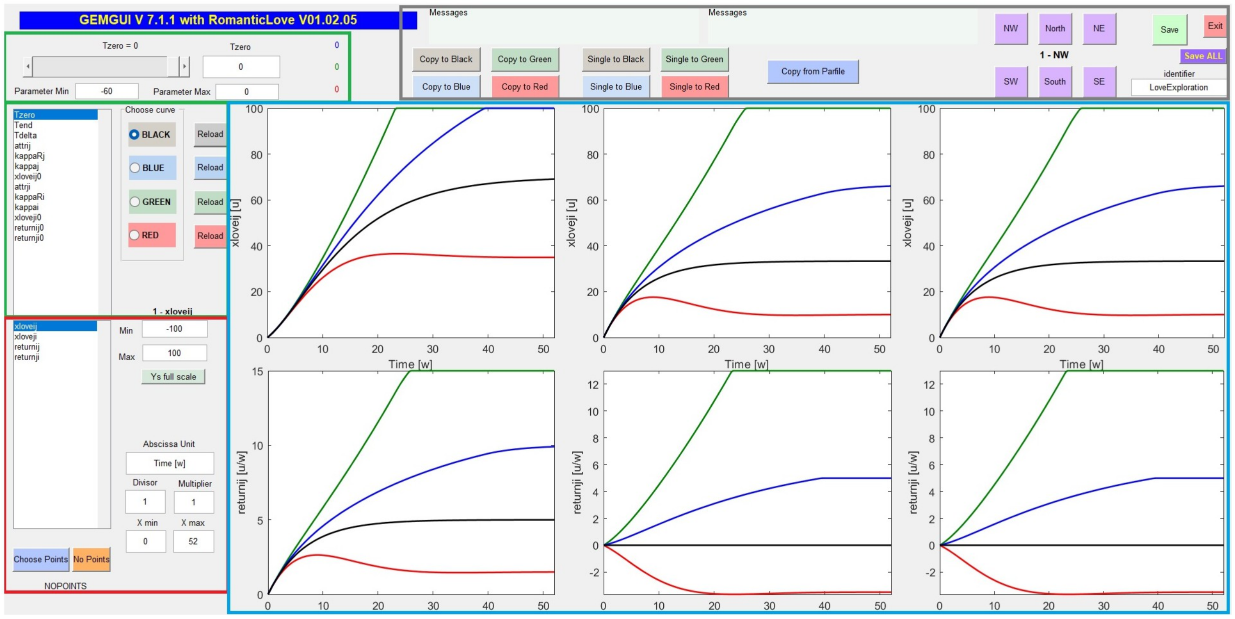


Fig 10. Visualizer interface: Green countered boxes report the model parameters and their control bar for values modification. The chosen configuration (black, blue, green and red) correspond to different parameter values resulting in different (corresponding) model predictions in the adjacent plots; red countered box shows the model variables along with their minimum and maximum allowed. The button "Choose Points", at the bottom of the box, loads external data to be superimposed to the model predictions. Buttons in the grey countered box enable the choice of the quadrant (NW, North, NE, etc.) where to plot the different model variables (light-blue countered box). This grey box contains also buttons for copying a value of a parameter on a specific parameter vector configuration (black, blue, green or red).

https://doi.org/10.1371/journal.pone.0316401.g010

integration is however performed in .dll's obtained from compiling the C++ sourcecode generated by the Autocoder, which makes model integration very fast: for example, by keeping pressed the interface button corresponding to some parameter increment, the time-course graphs of the state variables are seen to move rapidly to reflect successive values of the parameter as it varies in its range of definition.

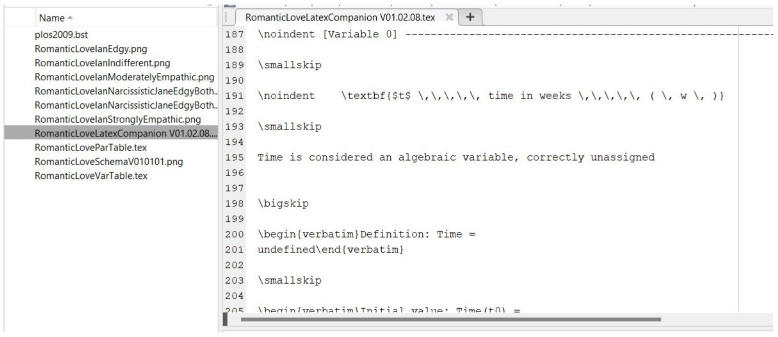
The Visualizer environment represents a robust tool to check the behavior of the model, with respect to all state variables, depending on the parameter choices: this is extremely useful both when first assessing the likely range of values of previously unknown parameters and when validating the model behavior with respect to plausible parameter values.

The LaTeX "Companion". A series of LaTeX files (Fig 11) are also provided as output of the Autocoder: these files are used to generate the Companion, the document containing the mathematical equations, parameter and variable descriptions.

Furthermore, two ".dat" files ("*ProjectName*.modpars.dat" and "*ProjectName*.vars.dat") are also produced by the Autocoder. These files contain a structural representation of all the information related to the model parameters and variables and are exploited by the C++ implementation, the Visualizer and the online platform.

Online platform (HTML, PHP). At CNR-IASI we have developed a standard HTML user interface and a PHP platform resident on our servers, able to accept and satisfy requests for service coming from the Internet.

Any model, successfully implemented, verified and validated, can afterwards be loaded on the CNR server, so that it becomes usable in user-to-machine and machine-to-machine modes. In user-to-machine mode the user can insert or modify the model's parameter values from an HTML interface and see on the same in the same interface page the corresponding



```
Name
plos2009.bst
RomanticLoveIanEdgy.png
RomanticLoveIanIndifferent.png
RomanticLoveIanModeratelyEmpathic.png
RomanticLoveIanNarcissisticJaneEdgyBoth..
RomanticLoveIanNarcissisticJaneEdgyBoth..
RomanticLoveIanStronglyEmpathic.png
RomanticLoveLatexCompanion V01.02.08....
RomanticLoveParTable.tex
RomanticLoveSchemaV010101.png
RomanticLoveVarTable.tex
```

```
RomanticLoveLatexCompanion V01.02.08.tex

\noindent [Variable 0] ----------------------------------------------------------

\smallskip

\noindent    \textbf{$t$ \,\,\,\,\, time in weeks \,\,\,\,\, ( \, w \, )}

\smallskip

Time is considered an algebraic variable, correctly unassigned


\bigskip

\begin{verbatim}Definition: Time =
undefined\end{verbatim}

\smallskip

\begin{verbatim}Initial value: Time(t0) =
```

Fig 11. LaTeX model description files produced by the Autocoder.

https://doi.org/10.1371/journal.pone.0316401.g011

graphical representation of the model forecasts of the time course of the state variables. machine-to-machine mode the user's softare can generate a SOAP call to the server, containing the simulation parameters and the desired evaluation time-points, and the server returns to the calling software the corresponding model forecasts at the desired time-points.

The platform can be found at the address:

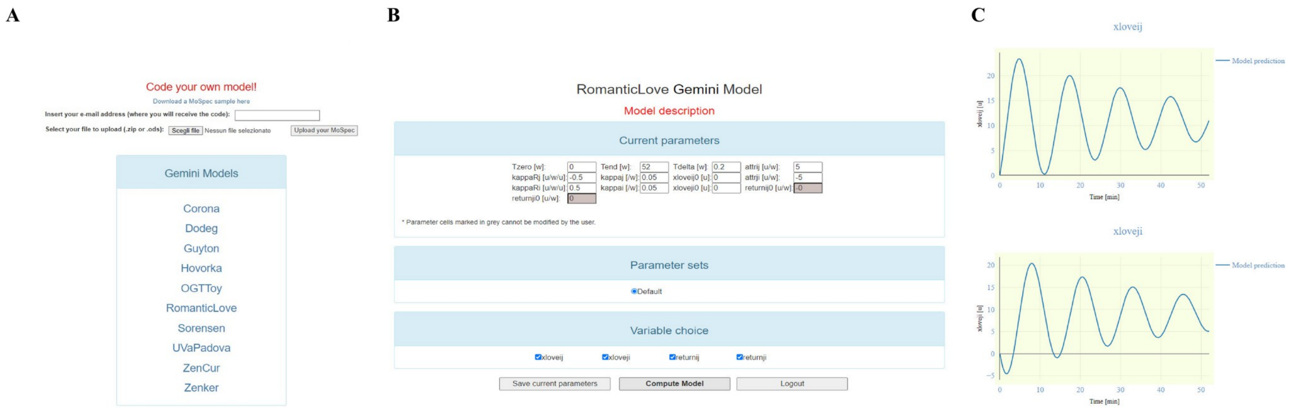
http://biomatlab.iasi.cnr.it/models/login.php

The following is a list of the models currently provided:

- Corona: a multinode generalized Susceptible—Exposed—Infectious—Removed (SEIR) model incorporating major features of the COVID19 pandemic in Italy;
- Guyton: a celebrated maximal model of the vascular system [29];
- OGTToy: a minimal model for the representation of data from an Oral Glucose Tolerance Test (OGTT);
- RomanticLove: a simple model that represents the dynamics of attraction between two people, similar to the Rinaldi romantic relationship models [30, 31];
- Sorensen: a complex, maximal model of the human glucose-insulin system for both normal and diabetic subjects (our implementation from the original Sorensen's PhD dissertation and subsequent corrections) [32];
- UVaPadova: another maximal model of the human glucose-insulin system for both normal and diabetic subjects (or implementation from published equations) [33];
- Dodeg: a simple, realistic mathematical description of cardiovascular dynamics after hemorrhagic shock [34];
- Zenker: a 2007 model portraying the evolution over time after blood loss of clinically relevant variables, such as the mean arterial pressure (MAP), the heart rate (HR), the cardiac output (CO), and the central venous pressure (CVP) [35];
- ZenCur: an upgraded version of the 2007 Zenker model for the representation of the hemorrhagic shock [36].

Any user interested in simulating one of these models can access the platform as "Guest". Following access, a webpage reporting the complete list of the models (Fig 12, panel A) will appear.

After having selected the desired model (for example the RomanticLove model), the user will be sent to the model-specific page showing the lists of model parameters and variables,



Code your own model!

Download a MoSpec sample here

Insert your e-mail address (where you will receive the code):

Select your file to upload (.zip or .ods): Scegli file Nessun file selezionato Upload your MoSpec

Gemini Models

Corona
Dodeg
Guyton
Hovorka
OGTToy
RomanticLove
Sorensen
UVaPadova
ZenCur
Zenker

RomanticLove Gemini Model

Model description

Current parameters

Tzero [w]: 0 Tend [w]: 52 Tdelta [w]: 0.2 attrij [u/w]: 5
kappaRj [u/w/u]: -0.5 kappaj [/w]: 0.05 xloveij0 [u]: 0 attrji [u/w]: -5
kappaRi [u/w/u]: 0.5 kappai [/w]: 0.05 xloveji0 [u]: 0 returnij0 [u/w]: -0
returnji0 [u/w]: 0

* Parameter cells marked in grey cannot be modified by the user.

Parameter sets

●Default

Variable choice

✓xloveij ✓xloveji ✓returnij ✓returnji

Save current parameters | Compute Model | Logout

Fig 12. Screenshots of the online platform. (A) webpage reporting the available models provided by the BioMatlab server; (B) webpage related to the RomanicLove model and showing the list of model parameters and variables, as well as the buttons for model simulation; (C) plots of two of the RomanticLove model variables.

https://doi.org/10.1371/journal.pone.0316401.g012

along with their units of measurement and reference or starting values, respectively (Fig 12, panel B). The user has the possibility to change any parameter value (if the parameter is not determined, see above) and to obtain the corresponding model predictions by clicking the "Compute Model" button (Fig 12, panel C).

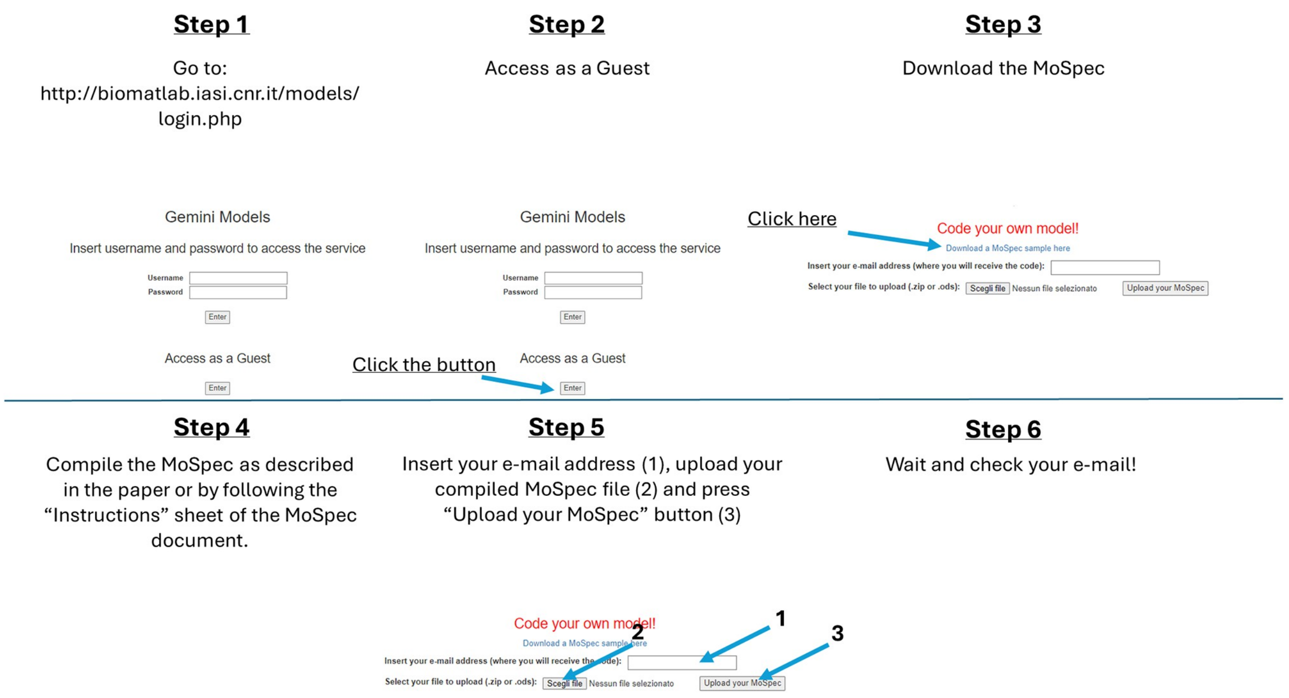
After logging into the platform as a guest, it is also possible to download a MoSpec sample (at the "Download a MoSpec sample here" link at the top of Fig 12). The file can be modified to include structured information about a specific model that the user wants to implement. It can be a model from literature or their own model. The procedure to follow is summarized in the following steps:

- once opportunely filled out with all the necessary information, the ".ods" file must be renamed with the name of the current project (chosen by the user), and saved in ".ods";
- the file can be uploaded as a single ".ods" file or as a ".zip" file if the user wants to include figures into the Companion that will be generated;
- once the file has been uploaded the user will receive a notification that the MATLAB, R and Julia implementation of the model, as well as the model LaTeX description, will be available via email (which must be provided in the "Insert your e-mail address" text box, Fig 12) within 2 working days.

Fig 13 summarizes the above steps.

Discussion

Mathematical models, applied across various medical fields, from basic research to clinical practice, play a crucial role in modern medicine, enabling researchers to better understand, diagnose, treat, and prevent diseases. They are employed to simulate the behavior of different biological systems and can be useful instruments for physicians to make informed decisions regarding diagnosis and prognosis. Mathematical models may also be of great help in optimizing medical treatments by determining the most effective dosage of a drug or in designing novel therapeutic approaches. Furthermore, their ability in providing insights into the underlying mechanisms of pathological conditions appears to be of great importance in



Step 1

Go to: http://biomatlab.iasi.cnr.it/models/login.php

Gemini Models

Insert username and password to access the service

Username

Password

Access as a Guest

Step 2

Access as a Guest

Gemini Models

Insert username and password to access the service

Username

Password

Access as a Guest

Step 3

Download the MoSpec

Code your own model!

Download a MoSpec sample here

Insert your e-mail address (where you will receive the code):

Select your file to upload (.zip or .ods): Nessun file selezionato

Step 4

Compile the MoSpec as described in the paper or by following the "Instructions" sheet of the MoSpec document.

Step 5

Insert your e-mail address (1), upload your compiled MoSpec file (2) and press "Upload your MoSpec" button (3)

Code your own model!

Download a MoSpec sample here

Insert your e-mail address (where you will receive the code):

Select your file to upload (.zip or .ods): Nessun file selezionato

Step 6

Wait and check your e-mail!

Fig 13. Step by step procedure for Gemini online platform access and for MoSpec download and upload.

https://doi.org/10.1371/journal.pone.0316401.g013

understanding disease progression, identifying therapeutic targets, and developing personalized medicine strategies.

Since mathematical models are becoming increasingly sophisticated and accurate due to technological advancements, it is clearly very useful to have available such tools that simplify and accelerate the correct implementation of standard mathematical models.

With the aim of addressing this objective and of rendering the modelling approach more feasible and accessible, the BioMatLab group has developed a system, easy to use, which is the result of a synergistic interaction between different programming languages and a customized spreadsheet, the MoSpec (Model Specification) file. Starting from the spreadsheet template containing different ad-hoc formatted sheets, the system allows automatic code generation for model implementation in different programming languages.

While other tools are available for implementing mathematical models, our system provides a free solution requiring no specific programming knowledge. Users only need to fill out a spreadsheet defining the model equations. This allows even non-programmers to produce and test ODE model implementations, which are automatically verified. Unlike other tools, our system also offers an informative document including tables, figures, and equations for easier scientific paper drafting.

It is to be stressed here that in our daily activity in the support of biomedical investigations our group also develops other kinds of (more mathematically sophisticated) models, based on Stochastic Differential Equations [37], Fractional Differential Equations [38, 39], Delay Differential Equations [40], Partial Differential Equations [41], even Fractional Stochastic Differential Equations [42]: such kinds of models are not easily handled by the current version of the MoSpec/Autocoder system. However, the bread-and-butter of everyday modeling activity still concerns Nonlinear Ordinary Differential Equations (ODE) [6, 9, 32–34, 36, 43–47], and for these the MoSpec represents a huge improvement in rapidity and reliability of implementation. It is also to be stressed that in a multi-disciplinary group not everybody is familiar with all details of coding for ODE models: even relatively junior figures can quickly become effective in contributing to the group's production by using the MoSpec.

It should be noticed that in a multidisciplinary group different professionals tend to prefer different computing languages: R for our statisticians, Matlab for our engineers, C++ for our IT and Julia for those of us more attuned to recent developments: the system allows each of these users to work with their favorite language. The automatically generated LaTeX companion in particular provides equation-by-equation comparison of the mathematical formulation (decided upon at model design phase) with the corresponding computing code, facilitating verification. It also provides the source-code for tables of variable and parameters to be directly incorporated into manuscripts for publication.

By requiring the user to routinely follow specific steps and clear instructions, thus allowing a standardized check for errors, the proposed system has proved to be an extremely useful tool for following all the Verification and Validation phases necessary for the exact implementation of our models. Furthermore, the MoSpec has become a shared language for people working in a group, structuring the development of any new project: the common standards and format simplify and improve communication and the gain in development speed increases the number of new projects that can be simultaneously undertaken.

A first challenge associated with the use of the MoSpec system is the learning curve involved in accurately compiling the necessary spreadsheets. While this may initially require practice, proficient use can quickly be achieved. Although the MoSpec compilation process offers limited flexibility, this constraint actually contributes to its strengths: by requiring users to follow a standardized format, the system ensures implementation consistency and facilitates verification. While this approach may restrict user creativity, it also minimizes the risk of errors commonly encountered in less structured systems. Also, standardization is extremely useful in a group work environment, making it possible for different people to be brought rapidly up-to-date on unfamiliar projects. It's important to note that the Autocoder outputs remain fully editable, providing users with opportunities for customization and tailored code script generation.

In conclusion, the CNR-IASI BioMatLab "Gemini MoSpec/Autocoder" system, which has become the standard procedure of our group for ODE model development, appears to be an extremely useful tool for the implementation, verification and validation of mathematical models.

Author Contributions

Conceptualization: Andrea De Gaetano.

Formal analysis: Andrea De Gaetano.

Methodology: Marcello Pompa, Simona Panunzi, Alessandro Borri, Laura D'Orsi, Andrea De Gaetano.

Software: Marcello Pompa, Simona Panunzi, Alessandro Borri, Laura D'Orsi.

Supervision: Andrea De Gaetano.

Validation: Marcello Pompa.

Visualization: Marcello Pompa, Simona Panunzi, Andrea De Gaetano.

Writing – original draft: Marcello Pompa, Simona Panunzi, Andrea De Gaetano.

Writing – review & editing: Simona Panunzi, Alessandro Borri, Laura D'Orsi, Andrea De Gaetano.

References

1. Mirnezami R, Nicholson J, Darzi A. Preparing for precision medicine. N Engl J Med. 2012; 366(6):489–491. https://doi.org/10.1056/NEJMp1114866 PMID: 22256780
2. Collins FS, Varmus H. A new initiative on precision medicine. New England journal of medicine. 2015; 372(9):793–795. https://doi.org/10.1056/NEJMp1500523 PMID: 25635347
3. De Gaetano A, Batzel J. Mathematical Modeling in Physiology: One Main Road Towards Personalized Medicine; 2018. 11th European Conference on Mathematical and Theoretical Biology, Lisbon, ECMTB 2018.
4. Hovorka R, Shojaee-Moradie F, Caroll P, Chassin L, Gowrie I, Jackson N, et al. Partitioning glucose dristribution/transport, disposal, and endogenous production during IVGTT. Americal Journal of Physiology. 2002; 282:992–1007. PMID: 11934663
5. Panunzi S, De Gaetano A, Mingrone G. Advantages of the single delay model for the assessment of insulin sensitivity from the intravenous glucose tolerance test. Theoretical Biology and Medical Modelling. 2010; 16(9):1–43. https://doi.org/10.1371/journal.pone.0257789 PMID: 20298586
6. Panunzi S, Palumbo P, De Gaetano A. A discrete single delay model for the intra-venous glucose tolerance test. Theoretical Biology and Medical Modelling. 2007; 4:1–16. https://doi.org/10.1186/1742-4682-4-35 PMID: 17850652
7. Palumbo P, Pizzichelli G, Panunzi S, Pepe P, De Gaetano A. Model-based control of plasma glycemia: test on populations of virtual patients. Mathematical Biosciences. 2014; 257:2–10. https://doi.org/10.1016/j.mbs.2014.09.003 PMID: 25223234
8. Hovorka R. Closed-loop insulin delivery from bench to clinical practice. Nature Reviews Endocrinology. 2011; 7:385–395. https://doi.org/10.1038/nrendo.2011.32 PMID: 21343892
9. De Gaetano A, Panunzi S, Matone A, Samson A, Vrbikova J, Bendlova B, et al. Routine OGTT: A Robust Model Including Incretin Effect for Precise Identification of Insulin Sensitivity and Secretion in a Single Individual. Plos One. 2013; 8:1–16. https://doi.org/10.1371/journal.pone.0070875
10. Sorensen JT. A Physiologic Model of Glucose Metabolism in Man and Its Use to Design and Improved Insulin Therapies for Diabetes; 1978.
11. Borri A, Palumbo P, Manes C, Panunzi S, De Gaetano A. Sampled-data observer-based glucose control for the artificial pancreas. Acta Polytech Hungarica. 2017; 14(1):79–94.
12. Borri A, Cacace F, De Gaetano A, Germani A, Manes C, Palumbo P, et al. Luenberger-like observers for nonlinear time-delay systems with application to the artificial pancreas: The attainment of good performance. IEEE Control Systems Magazine. 2017; 37(4):33–49. https://doi.org/10.1109/MCS.2017.2696759
13. Qiu K, Yang J, Gao Z, Xu F. A review of Modelica language in building and energy: Development, applications, and future prospect. Energy and Buildings. 2024; p. 113998. https://doi.org/10.1016/j.enbuild.2024.113998
14. Dabney JB, Harman TL. Mastering simulink. vol. 230. Pearson/Prentice Hall Upper Saddle River; 2004.
15. Adam T, Hashim U. COMSOL multiphysics simulation in biomedical engineering. Advanced Materials Research. 2014; 832:511–516. https://doi.org/10.4028/www.scientific.net/AMR.832.511
16. Villalobos Alva J. Introduction to Mathematica. In: Beginning Mathematica and Wolfram for Data Science: Applications in Data Analysis, Machine Learning, and Neural Networks. Springer; 2024. p. 1–54.
17. Oberkampf WL, Roy CJ. Verification and Validation in Scientific Computing. Cambridge: Cambridge University Press; 2010.
18. Popper KR. The logic of scientific discovery. Central Works of Philosophy v4: Twentieth Century: Moore to Popper. 2015; 4:262.
19. Boehm BW. Verifying and Validating Software Requirements and Design Specifications. IEEE Software. 1984; 1(1):75–88. https://doi.org/10.1109/MS.1984.233702
20. Pierce P. Software verification and validation. In: IEEE Technical Applications Conference. Northcon/96. Conference Record; 1996. p. 265–268.
21. Drug USF. General Principles of Software Validation, Guidance for Industry and FDA Staff; 2002.
22. Hunt BR, Lipsman RL, Osborn JE, Rosenberg JM. Differential equations with MATLAB. John Wiley & Sons; 2019.
23. Johnson B. Professional Visual Studio 2017 (English Edition). Wrox; 2017.

24. Wickham H, Çetinkaya-Rundel M, Grolemund G. R for data science. " O'Reilly Media, Inc."; 2023.

25. OS. Julia 1.9 Documentation; 2023. https://docs.julialang.org/en/v1/.

26. Brooks DR. Programming in HTML and PHP Coding for Scientists and Engineers. Springer; 2017.

27. Haoufadi Y. PHP & MYSQL: Guida alla programmazione PHP e MYSQL. Hacker Italia Editions Srl; 2018.

28. The Document Foundation: LibreOffice24.2.2; 2024. Available from: https://www.libreoffice.org/download/download-libreoffice/.

29. Guyton AC, Coleman TG, Granger HJ. Circulation: overall regulation. Annual review of physiology. 1972; 34(1):13–44. https://doi.org/10.1146/annurev.ph.34.030172.000305 PMID: 4334846

30. Rinaldi S. Love dynamics: The case of linear couples. Applied Mathematics and Computation. 1998; 95 (2-3):181–192. https://doi.org/10.1016/S0096-3003(97)10081-9

31. Gragnani A, Rinaldi S, Feichtinger G. Cyclic dynamics in romantic relationships. International Journal of Bifurcation and Chaos. 1997; 7(11):2611–2619. https://doi.org/10.1142/S0218127497001771

32. Panunzi S, Pompa M, Borri A, Piemonte V, De Gaetano A. A revised Sorensen model: Simulating glycemic and insulinemic response to oral and intra-venous glucose load. Plos one. 2020; 15(8):e0237215. https://doi.org/10.1371/journal.pone.0237215 PMID: 32797106

33. Pompa M, Panunzi S, Borri A, De Gaetano A. A comparison among three maximal mathematical models of the glucose-insulin system. PloS one. 2021; 16(9):e0257789. https://doi.org/10.1371/journal.pone.0257789 PMID: 34570804

34. D'Orsi L, Curcio L, Cibella F, Borri A, Gavish L, Eisenkraft A, et al. A mathematical model of cardiovascular dynamics for the diagnosis and prognosis of hemorrhagic shock. Mathematical Medicine and Biology: A Journal of the IMA. 2021; 38(4):417–441. https://doi.org/10.1093/imammb/dqab011 PMID: 34499176

35. Zenker S, Rubin J, Clermont G. From inverse problems in mathematical physiology to quantitative differential diagnoses. PLoS computational biology. 2007; 3(11):e204. https://doi.org/10.1371/journal.pcbi.0030204 PMID: 17997590

36. Curcio L, D'Orsi L, Cibella F, Wagnert-Avraham L, Nachman D, De Gaetano A. A simple cardiovascular model for the study of hemorrhagic shock. Computational and Mathematical Methods in Medicine. 2020; 2020. https://doi.org/10.1155/2020/7936895 PMID: 33425003

37. Picchini U, GAETANO AD, Ditlevsen S. Stochastic differential mixed-effects models. Scandinavian Journal of statistics. 2010; 37(1):67–90. https://doi.org/10.1111/j.1467-9469.2009.00665.x

38. De Gaetano A, Sakulrang S, Borri A, Pitocco D, Sungnul S, Moore EJ. Modeling continuous glucose monitoring with fractional differential equations subject to shocks. Journal of Theoretical Biology. 2021; 526:110776. https://doi.org/10.1016/j.jtbi.2021.110776 PMID: 34058226

39. Sakulrang S, Moore EJ, Sungnul S, de Gaetano A. A fractional differential equation model for continuous glucose monitoring data. Advances in Difference Equations. 2017; 2017:1–11. https://doi.org/10.1186/s13662-017-1207-1

40. Gaz C, De Gaetano A, Manes C, Palumbo P, Borri A, Panunzi S. Effective control of glycemia using a simple discrete-delay model. IFAC-PapersOnLine. 2017; 50(1):13526–13531. https://doi.org/10.1016/j.ifacol.2017.08.2345

41. Borri A, Panunzi S, De Gaetano A. A glycemia-structured population model. Journal of Mathematical Biology. 2016; 73:39–62. https://doi.org/10.1007/s00285-015-0935-7 PMID: 26440781

42. Panunzi S, Borri A, D'Orsi L, De Gaetano A. Order estimation for a fractional Brownian motion model of glucose control. Communications in Nonlinear Science and Numerical Simulation. 2023; 127:107554. https://doi.org/10.1016/j.cnsns.2023.107554

43. Pompa M, De Gaetano A, Borri A, Farsetti A, Nanni S, Pontecorvi A, et al. A physiological mathematical model of the human thyroid. Journal of Computational Science. 2024; p. 102236. https://doi.org/10.1016/j.jocs.2024.102236

44. Pompa M, Cerasa A, Panunzi S, De Gaetano A. HAPC Model of Crowd Behavior during Crises. Mathematics. 2023; 11(12):2711. https://doi.org/10.3390/math11122711

45. Panunzi S, De Gaetano A, Mingrone G. Approximate linear confidence and curvature of a kinetic model of dodecanedioic acid in humans. American Journal of Physiology-Endocrinology And Metabolism. 2005; 289(5):E915–E922. https://doi.org/10.1152/ajpendo.00503.2003 PMID: 15972268

46. Diego S Rodrigues VAGLATBMJ Guilherme A Soares, Miranda JRA. Accessing the pharmacokinetics of magnetic nanoparticles in cirrhosis-associated hepatocarcinogenesis by ordinary differential equation modeling and AC biosusceptometry. Mathematics in Medical and Life Sciences. 2024; 1 (1):2391739. https://doi.org/10.1080/29937574.2024.2391739

47. Panunzi S, De Gaetano A. A modelling approach to hepatic glucose production estimation. Plos one. 2022; 17(12):e0278837. https://doi.org/10.1371/journal.pone.0278837 PMID: 36542610