Universitat de Lleida

EasyModel: a user-friendly web-based tool for model building, simulation, and analysis in systems and synthetic biology

Jordi Bartolome Tomas

http://hdl.handle.net/10803/693972

Universitat de Lleida

DOCTORAL THESIS

EasyModel: a user-friendly web-based tool for model building, simulation, and analysis in systems and synthetic biology

Jordi Bartolome Tomas

Dissertation to obtain the degree of Doctor by the University of Lleida
Doctoral Program in Engineering and Information Technology

Director
Rui Carlos Vaqueiro de Castro Alves
Francesc Xavier Solsona Tehas

Tutor
Francesc Xavier Solsona Tehas

2024

Life is like riding a bicycle. To keep your balance, you must keep moving.

(Albert Einstein)

To all the beings that made this possible. To you as a reader.

Abstract

Background: Mathematical modeling is a required skill in systems and synthetic biology. However, mastering the creation of mathematical models for molecular systems biology typically involves a steep learning curve. Despite the availability of numerous modeling tools, few incorporate interfaces that expedite this process by enhancing user-friendliness in modeling, simulation and analysis. EasyModel was designed with a user-friendly philosophy to facilitate the learning curve, accommodating both novice and advanced users effectively.

Methods: The main element of this work is the development of the web-based tool EasyModel. EasyModel was designed to offer a user-friendly experience in the scope of modeling, simulation and analysis of biological systems. The tool has been developed using the Java EE technology in conjunction with the Vaadin web UI framework, the Wolfram Mathematica calculus platform, the MySQL database manager and others. Features include: deterministic and stochastic simulations, steady state seeking, gains and sensitivities analysis, parameter scanning analysis, kinetic laws as mathematical formalisms, download of the generated Mathematica notebook, import/export of models in SBML file format and more.

Results: The present work presents the version 2.4 of EasyModel. This version upgrades from Vaadin 8 to 24 offering an up-to-date and sleeker user interface. It also introduces some functional improvements as a simulation job queue system where users can launch simulation jobs at anytime and share the results as well. The execution speed of the stochastic simulation has been improved to approximately four times faster. Tool is available at: https://easymodel.udl.cat.

Conclusions: EasyModel helps the novel user in systems biology modeling by placing a user-friendly web interface layer over the Mathematica calculus engine. Expert users can download the generated Mathematica notebook for further tailoring and performing more advanced simulations and analyses.

Resum

Introducció: La modelització matemàtica és una habilitat necessària per a la biologia de sistemes i la biologia sintètica. No obstant això, dominar la creació de models matemàtics per a la biologia molecular de sistemes sol implicar una corba d'aprenentatge pronunciada. Malgrat la disponibilitat de nombroses eines de modelització, poques incorporen interfícies que agilitin aquest procés al millorar la facilitat d'ús en la modelització, simulació i anàlisi. EasyModel ha estat dissenyat amb una filosofia de fàcil ús per facilitar la corba d'aprenentatge, ajudant tant als usuaris novells com als avançats de manera efectiva.

Mètodes: L'element principal d'aquest treball és el desenvolupament de l'eina web EasyModel. EasyModel ha estat dissenyada per oferir una experiència fàcil d'utilitzar en l'àmbit de la modelització, simulació i anàlisi de sistemes biològics. L'eina s'ha desenvolupat utilitzant la tecnologia Java EE conjuntament amb el framework d'interfície web Vaadin, la plataforma de càlcul Wolfram Mathematica, el gestor de bases de dades MySQL i altres. Les característiques inclouen: simulacions deterministes i estocàstiques, recerca d'estat estacionari, anàlisi de guanys i sensibilitats, anàlisi d'escaneig de paràmetres, lleis cinètiques com a formalismes matemàtics, descàrrega del *notebook* de Mathematica, importació/exportació de models en format de fitxer SBML i més.

Resultats: El present treball presenta la versió 2.4 d'EasyModel. Aquesta versió actualitza la versio de Vaadin de 8 a 24, oferint una interfície d'usuari més actualitzada i refinada. També introdueix algunes millores funcionals, com ara un sistema de cua de treballs de simulació on els usuaris poden llançar treballs de simulació en qualsevol moment i compartir els resultats. La velocitat d'execució de la simulació estocàstica s'ha millorat en aproximadament quatre vegades. L'eina està disponible a: https://easymodel.udl.cat.

Conclusions: EasyModel ajuda als usuaris novells en la modelització de biologia de sistemes mitjançant una capa d'interfície web fàcil d'utilitzar sobre el motor de càlcul de Mathematica. Els usuaris experts poden descarregar el *notebook* de Mathematica generat per personalitzar-lo encara més i realitzar simulacions i anàlisis més avançades.

Resumen

Introducción: La modelización matemática es una habilidad necesaria para la biología de sistemas y la biología sintética. Sin embargo, dominar la creación de modelos matemáticos para la biología de sistemas moleculares suele implicar una curva de aprendizaje pronunciada. A pesar de la disponibilidad de numerosas herramientas de modelización, pocas incorporan interficies que agilicen este proceso al mejorar la facilidad de uso en la modelización, simulación y análisis. EasyModel fue diseñado con una filosofía de fácil uso para facilitar la curva de aprendizaje, ayudando de manera efectiva tanto a usuarios novatos como avanzados.

Métodos: El elemento principal de este trabajo es el desarrollo de la herramienta web EasyModel. EasyModel fue diseñado para ofrecer una experiencia amigable en el ámbito de la modelización, simulación y análisis de sistemas biológicos. La herramienta se ha desarrollado utilizando la tecnología Java EE junto con el framework de interfaz web Vaadin, la plataforma de cálculos Wolfram Mathematica, el gestor de bases de datos MySQL y otros. Las características incluyen: simulaciones deterministas y estocásticas, búsqueda de estado estacionario, análisis de ganancias y sensibilidades, análisis de escaneo de parámetros, formalismos matemáticos, descarga del *notebook* de Mathematica, importación/exportación de modelos en formato SBML, etc.

Resultados: El presente trabajo presenta la versión 2.4 de EasyModel. Esta versión actualiza la versión de Vaadin de 8 a 24, ofreciendo una interfaz de usuario más actualizada y refinada. También introduce algunas mejoras funcionales, como un sistema de cola de trabajos de simulación en el que los usuarios pueden lanzar trabajos de simulación en cualquier momento y compartir los resultados. La velocidad de ejecución de la simulación estocástica se ha mejorado en aproximadamente cuatro veces. La herramienta está disponible en: https://easymodel.udl.cat.

Conclusiones: EasyModel ayuda al usuario novato en la modelización de biología de sistemas al colocar una capa de interfaz web fácil de usar sobre el motor de cálculos de Mathematica. Los usuarios expertos pueden descargar el *notebook* de Mathematica para personalizarlo aún más y realizar simulaciones y análisis más avanzados.

Acknowledgements

Gratitude is not only the greatest of virtues, but the parent of all others.

(Marcus Tullius Cicero)

Firstly, I would like to express my gratitude to my supervisors, Prof. Rui Alves and Prof. Francesc Solsona, whose constant attention and care propelled me diligently and steadily throughout this journey.

I would also like to extend my heartfelt appreciation to my friends and family for their endless support throughout my life, particularly to my family - my parents, Anselm and Maria Dolors, my sibling Eloi, my grandparents and uncles. Their steadfast presence by my side has been an indispensable factor in my journey to reach this point.

Furthermore, I would like to extend my sincere appreciation to all the other individuals who have demonstrated their care and support for me at different times, playing a pivotal role in pushing me towards progress.

Contents

1 Introduction and scope of the research

Mathematics is the language of nature, and biological systems are its expressions waiting to be decoded.

(Albert Einstein)

Over the past few years, we've witnessed a notable uptick in the number of user-friendly tools designed for modeling and simulating biological networks. However, many of these tools are hindered by their limited range of applicability and operational functions. At the same time, math-oriented languages like Wolfram Mathematica (1, 2) offer greater modeling flexibility but necessitate a higher level of user expertise. To address these concerns, this project endeavors to supply a user-friendly web application that's linked to the Mathematica calculation language, thereby encouraging non-expert users to leverage this powerful platform for their biological network modeling needs.

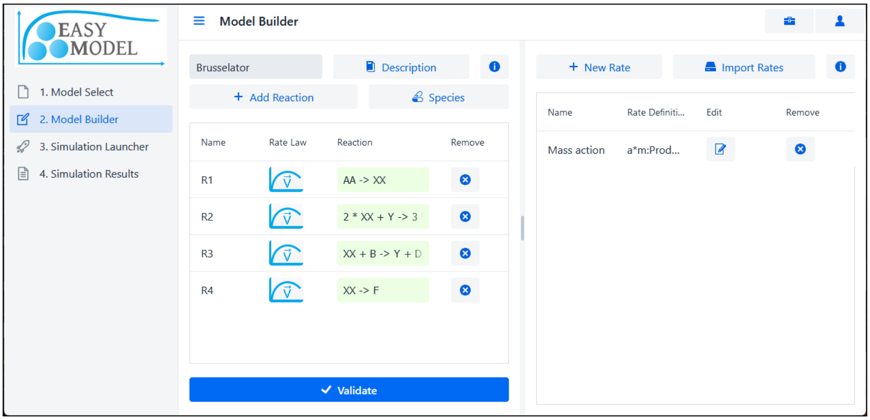
EasyModel (see Figure 1.1) is a user-friendly web application presented in this thesis that empowers researchers to simulate and model their own biological network models. With its intuitive interface and customizable features, this application makes it easy for users to design and refine their models in a streamlined and efficient manner. Overall, EasyModel represents a powerful tool for researchers seeking to analyze and better understand biological networks. The application provides users with the flexibility to design and customize their own models, reactions, and kinetic rate functions. In addition, EasyModel includes numerous predefined and commonly used kinetic rate functions to facilitate the modeling process.

In addition to the user-friendly property, it's also important to note that Easymodel provides value to expert users as well. Specifically, those who possess a higher degree of familiarity with the Mathematica symbolic language can benefit from the option to download and fine-tune the code generated by the application to better suit their individual use cases. This generated Mathematica notebook contains the full model as well as the simulation using the

Mathematica language, so it can be executed in a local Mathematica installation. As a result, this project is well-suited to meet the needs of a broader range of users and has the potential to be a valuable resource for the scientific community as a whole.

We believe this tool will be a very useful service for researchers in Systems and Synthetic Biology that want to create, simulate and analyze mathematical models of biological circuits in a user-friendly way.

The present work discusses the latest version 2.4 of EasyModel.



Name	Rate Law	Reaction	Remove
R1		AA -> XX	
R2		2 * XX + Y -> 3	
R3		XX + B -> Y + D	
R4		XX -> F	

Name	Rate Definiti...	Edit	Remove
Mass action	a*m:Prod...		

Figure 1.1: EasyModel: a user-friendly web-based tool for model building, simulation, and analysis in systems and synthetic biology.

1.1 Background

Molecular systems biology is a quantitative and integrative discipline that heavily relies on the use of mathematical and statistical models. Thus, creating and analyzing mathematical models are important skills for the systems biologist. Acquiring these skills is usually a task with a slow learning curve.

Developing tools for the modeling and analysis of biological systems has always been a challenge. Currently, there is a considerable amount of tools for modeling and simulation of biological systems (3, 4). Many of them are standalone and provide a fixed set of functionalities. Still, a small number of more general platforms for mathematical computation (PMC), such as Mathematica (1) or Maple (5), can also be adapted for systems biology modeling. These PMC offer the possibility of developing code to extend their already impressive set of functionalities. Mathematica and Maple stand aside from other PMCs because they also support symbolic analysis. Users of a PMC for systems biology modeling must become experts in the coding language of the platform. This drawback can be overcome by implementing a user-friendly application that uses the PMC as the motor for calculations, as other modeling tools have already demonstrated (6, 7, 8).

We have developed EasyModel to use the Wolfram Mathematica PMC as the simulation motor and combine it with a user-friendly interface suitable for the beginner mathematical modeler. EasyModel background is the mathematical modeling in systems biology. Specifically, it deals with kinetic modeling of molecular biology networks. This means that the models work with flux reactions. Each reaction determines which substrates and regulators are needed to generate a given set of products. Associated to each flux reaction there is a rate function that determines the rate at which the reaction occurs, as a function of the substrates and regulators.

While EasyModel stands out for its user-friendly graphical user interface (GUI), it also provides the possibility of developing more advanced functionality for expert users that will be described in further chapters.

1.2 Mathematical modeling in biology systems

A biological system is defined as a set of physical entities, usually numerous and diverse, that influence each other and that are physically and functionally separated from their environment (9). The functional separation is a consequence of the fact that biological systems are far from thermodynamic equilibrium, in contrast with the environment. Thus, in order to analyze biological systems, researchers must create a conceptual model of the system, where all the relevant elements are included and unsubstantial elements are omitted. A need for a mathematical model often arises due to the non linear dynamics of biological systems. Such mathematical models can represent these dynamics, thus providing a prediction method for the behavior of the system. Mathematical models of biological systems are designed to allow researchers to attain qualitative and quantitative understanding and control over the systems that the models represent (see Figure 1.2).

Molecular systems biology is a rapidly evolving and interdisciplinary field that employs quantitative and integrative approaches in biomedical and biological research. It seeks to unravel the intricate interactions between the molecular components of cells and gain a comprehensive understanding of their behavior (10). Computational methods play a crucial role in this discipline, enabling researchers to analyze complex biological data and construct mathematical

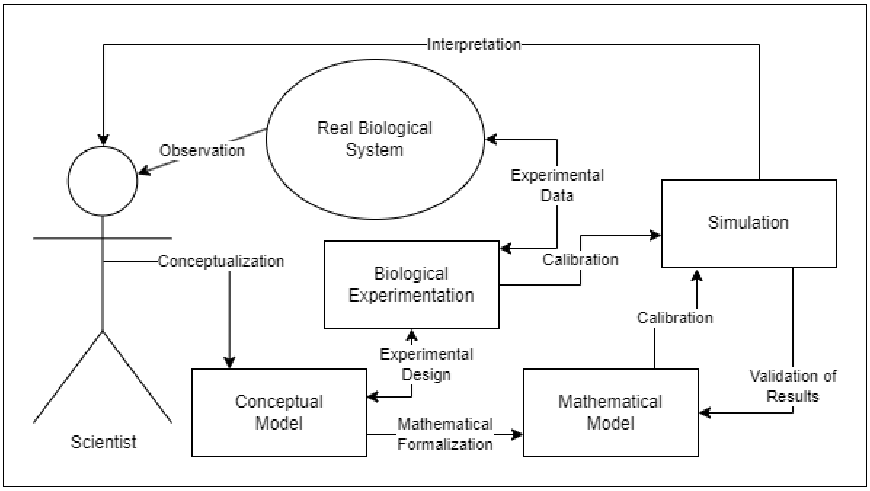


Figure 1.2: Mathematical modeling process in biosciences.

models that provide insights into cellular processes (11).

The integration of computational tools and techniques allows systems biologists to study biological systems at multiple scales, from individual molecules to entire organisms (11). This integrative approach facilitates the identification of key regulatory networks, signaling pathways, and functional relationships between different biomolecules (12). As a result, researchers can elucidate the emergent properties that arise from the collective behavior of these components, shedding light on fundamental biological processes and disease mechanisms (12, 13).

One of the important tools for research in Molecular Systems Biology is the creation and analysis of mathematical models. These models serve as a virtual representation of biological systems and enable researchers to simulate and predict cellular behaviors under different conditions (14). They provide a valuable means to test hypotheses, optimize experimental designs, and guide future investigations. Through model-driven analyses, systems biologists can uncover hidden patterns, validate experimental findings, and generate novel predictions, enhancing the overall understanding of complex biological phenomena.

In addition to traditional experimental techniques, molecular systems biologists employ an array of computational tools, such as data mining, machine learning, network analysis, and statistical modeling (11). These computational methods empower researchers to extract meaningful information from vast datasets and decipher complex biological relationships that may not be readily apparent through conventional approaches.

As a result of its quantitative nature and reliance on computational methods, acquiring skills in molecular systems biology can be a challenging and gradual process (13). Proficiency in these techniques often requires a combination of biological knowledge, mathematical aptitude, and programming expertise. Aspiring systems biologists need to familiarize themselves with diverse disciplines, including bioinformatics, genomics, systems theory, and statistics. However, the growing availability of online courses, workshops, and research-oriented programs can facilitate the learning journey for students and researchers interested in this field.

To stay at the forefront of this rapidly evolving discipline, systems biologists must keep abreast of the latest advancements in computational tools and technologies (11). Moreover, collaboration with experts from various fields, such as biology, computer science, and engineering, can foster a synergistic approach to tackle complex biological questions and promote innovation in molecular systems biology (12).

One of the primary goals of systems biology is to examine the emerging properties that arise from the interactions among molecular components within cells. The ultimate objective is to identify design principles within molecular and cellular circuits. These principles are derived from exploring the topologies (15, 16, 17), parameter ranges (18, 19, 20), and dynamic behaviors (16, 21) of specific biological circuits. By correlating the impact of design variations with the "fitness" of an organism (22, 23) and considering the action of natural selection on alternative circuit designs, these studies help illuminate why diverse designs for the same function can exist in different organisms.

1.3 Objectives

The primary objective of this project has been to develop a software program for modeling and simulation in systems biology that offers a user-friendly experience, especially for newcomers such as students in the field. By simplifying the learning curve, the software aims to make the subject more accessible. The project has been aptly named **EasyModel**, reflecting its emphasis on providing an intuitive and easy-to-use platform for modeling and simulation of biological systems.

Before developing this project, expert users could use Wolfram Mathematica as a modeling, analysis and simulation environment to work with biological systems. Mathematica stands as a formidable engine for mathematical modeling, boasting an expansive collection of pre-built functions that are suitable for biological systems modeling. However, the utilization of Mathematica within the realm of systems biology necessitates a deep understanding of the Mathematica symbolic programming language to manually construct and scrutinize models. Unfortunately, this requirement renders Mathematica less accessible to newcomers and students venturing into the realm of systems biology.

The inception of EasyModel was driven by the intention of bridging the gap between novice users and the potent capabilities of Mathematica, as applied to mathematical modeling in systems biology. Our approach involved crafting an intuitive graphical user interface (GUI) that interfaces seamlessly with the intricate functionalities of Mathematica. This GUI serves as a conduit that simplifies the interaction with the robust Mathematica functions, making them more user-friendly and approachable.

To further enhance the tool's usability, we strategically designed EasyModel as a web application. To do so, we leveraged the Java capabilities to connect to the Mathematica kernel through the specific programming library. This decision ensures accessibility to a broader audience, enabling individuals with nothing more than a browser and internet connection to engage with the application. This inclusive approach transcends operating system constraints and eliminates the prerequisite of possessing a commercial Mathematica license.

Notably, EasyModel isn't solely aimed at novices; it also significantly benefits expert users. It encompasses fundamental operations as well as more advanced features e.g. stochastic simulations and parameter scanning. By providing a more streamlined workflow, it alleviates the potential for errors and time wastage that may arise when manually constructing models within the intricacies of Mathematica programming language code. Furthermore, for those who are proficient in Mathematica, the application offers the choice to download a Mathematica notebook file that encompasses the complete generated Mathematica code, encompassing the model, simulation, and analysis components. This provides experienced users the flexibility to explore more advanced operations by utilizing a locally installed Mathematica.

In essence, EasyModel serves as a conduit between the world of systems biology and the immense capabilities of Mathematica. By fashioning a user-friendly entry point and harnessing the web application framework, we've fostered an environment where learners, students, and experts can get access to mathematical modeling with ease.

1.4 Thesis contribution

Although there is already a considerable amount of tools for modeling and simulation of biological systems (3, 4), many of them take a scientific user interface approach, thus hindering its use to novel users. The main contribution of this thesis has been the development of the EasyModel tool, which cares for the initiating user into this field of tools.

EasyModel has made a notable contribution to the systems biology community by providing an intuitive and user-friendly platform for modeling and simulating biological systems. It simplifies the process of constructing and analyzing models, enabling users to simulate both deterministic and stochastic behaviors with ease. EasyModel allows users to perform dynamic and stochastic simulations, making it suitable for studying complex molecular interactions, gene regulation, and cellular processes.

A key feature of EasyModel is its use of Wolfram Mathematica to perform the underlying calculus operations, similar to other advanced tools in the field. This integration allows for robust computation and accurate simulations, leveraging Mathematica's powerful engine to handle complex biological systems efficiently. As EasyModel allows advanced users to download the generated Mathematica notebooks, it enables them to further leverage the capabilities of the Mathematica program. This feature is relatively unique in the field of systems biology software, as it provides users with the opportunity to directly interact with and modify the underlying Mathematica code used in the simulations.

In addition to dynamic simulations, EasyModel supports parameter scanning and sensitivity analysis, giving researchers deeper insights into model behavior under various conditions. Its ability to handle parallel simulations helps reduce computation time, particularly when running stochastic simulations that require multiple replicates.

By supporting standard formats like SBML, EasyModel promotes collaboration, data sharing, and reproducibility, making it a valuable tool for the systems biology research community. Its versatility, combined with the computational power of Mathematica, allows researchers to tackle both simple and large-scale models effectively.

EasyModel development has focused primarily on providing a user-friendly experience. The current version 2.4 also caters to advanced users by enabling some advanced simulations and functionalities. While other simulation tools may offer a broader range of simulation options, this was not the main objective of EasyModel.

We believe this web application will meet the needs of both novice and expert users and provide a valuable addition to the current array of tools in the field of modeling and simulation in systems biology. A comparison with other tools will be discussed in the following Chapter 2, Related Work.

1.5 Publications

The following publications have been derived from the work on this thesis.

1.5.1 Journal publications

- Jordi Bartolome, Rui Alves, Francesc Solsona, Ivan Teixido, EasyModel: user-friendly tool for building and analysis of simple mathematical models in systems biology, Bioinformatics, Volume 36, Issue 3, February 2020, Pages 976–977, https://doi.org/10.1093/bioinformatics/btz659.

1.5.2 Conference publications and attendance

- Bartolome, J.; Alves, R. and Solsona, F. (2018). EasyModel: building and analysis of mathematical models for biological systems with a wide range of target users. In Proceedings of the Jornada de Benvinguda Doctorands UdL (2018), Universitat de Lleida, Lleida, Spain, 2018.

- Bartolome, J.; Alves, R. and Solsona, F. (2019). User friendly tool for building and analysis of simple mathematical models in systems biology. In Proceedings of the 19th International Conference Computational and Mathematical Methods in Science and Engineering (CMMSE 2019), Rota, Spain, June 30 - July 6, 2019.

- Bartolome, J.; Alves, R. and Solsona, F. (2019). EasyModel: user friendly tool for building and analysis of simple mathematical models in systems biology. In Proceedings of Workshop project IMPACTS (2019), Universitat de Lleida, Lleida, Spain, 2019.

- Bartolome, J.; Alves, R. and Solsona, F. (2020). EasyModel 1.1: User-friendly Stochastic and Deterministic Simulations for Systems Biology Models. In Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2020) - BIOINFORMATICS; ISBN 978-989-758-398-8; ISSN 2184-4305, SciTePress, pages 145-149. DOI: 10.5220/0008966001450149.

1.6 Availability and requirements

EasyModel is freely available on the web. No user registration is required. All that is needed to use it is a standard web browser.

- **Project name:** EasyModel.

- **Project home page:** https://easymodel.udl.cat/.
- **Source code:** https://github.com/jordibart/easymodel/.
- **Operating system(s):** Platform-independent. Tested on Linux, Windows, macOS and Android.
- **Programming language:** Java, Mathematica and SQL.
- **Other requirements:** Any standard web browser. Tested on Mozilla Firefox, Google Chrome, Safari and Microsoft Edge.
- **License:** GNU GPL.
- **Any restrictions to use by non-academics:** None.

2 Related work

The true sign of intelligence is not knowledge but imagination.

(Albert Einstein)

A comparison between 12 software packages was carried out on June 2006 (3). The comparison highlighted which applications were more appropriate to use in a given task requirements. It was aimed for non-expert users who needed to choose in an easy way a package by taking into account all the required points of their kinetic modeling. The key points compared in this comparison were: running operating system, system requirements, cross-compatibility, open sourced, type of user interface, compartmentalization support, type of simulation engine, functionality for model analysis and others. Overall, the study concluded that there was a fair amount of overlap in the functionality of the software packages but that each of them was best suited for specific classes of problems or systems. Certain software problems were uncovered in this study which encouraged tool developers to improve their software. Despite this, study stated that the available tools at that moment were a formidable array of tools for applying mathematical models in biological research. The fact that this kind of tools become one of the usual laboratory tools primarily depends on how the software improve and on how the mainstream molecular biology shifts its focus to understand how molecular processes work within the larger context of biological systems.

Furthermore, another study was carried out on December 2015 (24). The study concluded that Systems Biology Toolbox and COPASI could be a good choice for academics. On the other side, Cellware, COPASI and Virtual Cell could be a more appropriate choice for non-academic users.

Yet another study simulated the same systems in various software and then compared the obtained simulating results (25). It was observed that there was almost no differences in the accuracies of the simulated systems. The slight changes in results were caused by the algorithms that were used for solving the ordinary differential equations that were computed by distinct optimization methods. The study also observed that each tool demanded different computational power, specially when the simulated system dimensions were increased.

Also, authors observed that tools written in C++ were faster in execution time since they used a compiled language.

2.1 Comparing EasyModel with other stand-alone software

Stand-alone software refers to applications designed to operate independently, without relying on other software or external services for their core functions. In the context of modeling and simulation in systems biology, stand-alone tools are particularly valuable because they offer self-contained environments that include all the necessary components, libraries, and resources within a single package. These applications typically run locally on a user's device, such as a desktop or laptop computer, and are capable of performing complex simulations and analyses without requiring continuous network connectivity or additional external modules.

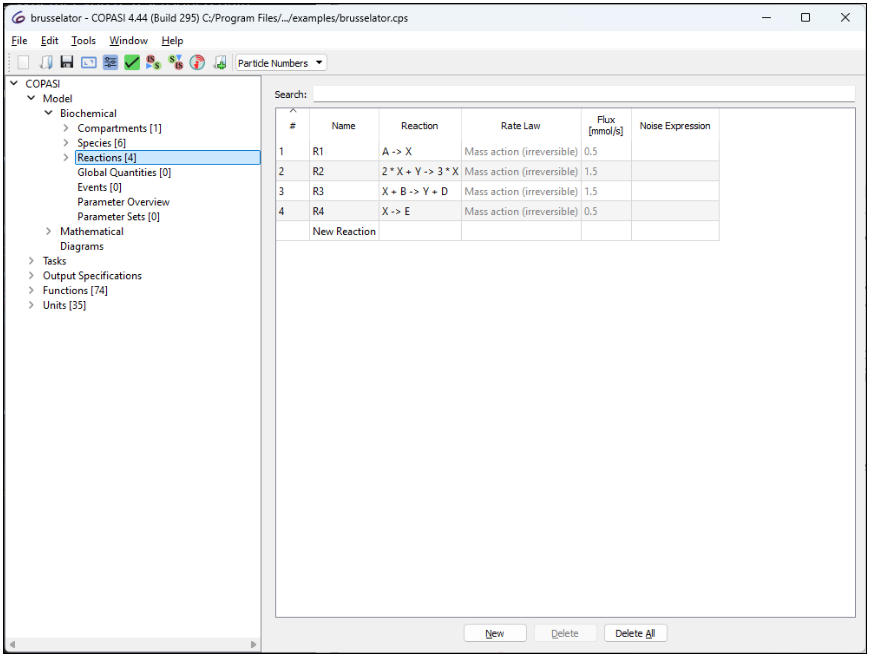
For modeling and simulation tasks in systems biology, stand-alone software often integrates advanced computational algorithms, extensive biological databases, and user interfaces—whether graphical or text-based—into a cohesive system. This integration enables researchers to build, simulate, and analyze biological models effectively, even when offline. The absence of dependencies on external services ensures that users can rely solely on the software's built-in features and tools without needing to access remote servers or cloud-based resources.

In the following paragraphs, EasyModel will be compared to other stand-alone software with similar and related features.

A similar stand-alone software is COPASI (26, 27) (see Figure 2.1)(available for Windows, Linux, and Mac OS X), which is the successor to Gepasi (28) (available for Windows). COPASI excels in handling both simple and complex models, offering a user-friendly interface and advanced features. These include stochastic simulation methods, essential for studying systems with inherent randomness, and multicompartmentalization.

Randomness in biological systems often arises from the stochastic nature of molecular interactions, especially when dealing with small numbers of molecules. Stochastic simulations, such as the Gillespie algorithm, allow researchers to capture this randomness and observe how it influences system dynamics. In contrast to deterministic models, which provide the same result given the same initial conditions, stochastic models generate different outcomes with each simulation run. This is crucial for studying processes like gene expression, where randomness can lead to significant variation in cellular behavior.

Multicompartmentalization refers to the modeling of biological systems as divided into distinct compartments, each with its own specific environment or set of conditions. This is particularly important when simulating processes that occur in different cellular locations, such as the nucleus, cytoplasm, or extracellular space. Multicompartmental models allow for the movement of substances between these compartments, simulating more realistic biological behaviors, such as diffusion, transport, and compartment-specific reactions. This feature enables the accurate modeling of spatial heterogeneity, which can significantly affect system behavior.


brusselator - COPASI 4.44 (Build 295) C:/Program Files/.../examples/brusselator.cps
File
Edit
Tools
Window
Help
Particle Numbers
New
Delete
Delete All

- COPASI
  - Model
    - Biochemical
      - Compartments [1]
      - Species [6]
      - Reactions [4]
      - Global Quantities [0]
      - Events [0]
      - Parameter Overview
      - Parameter Sets [0]
    - Mathematical
    - Diagrams
  - Tasks
  - Output Specifications
  - Functions [74]
  - Units [35]

Search:

#	Name	Reaction	Rate Law	Flux [mmol/s]	Noise Expression
1	R1	A -> X	Mass action (irreversible)	0.5	
2	R2	2 * X + Y -> 3 * X	Mass action (irreversible)	1.5	
3	R3	X + B -> Y + D	Mass action (irreversible)	1.5	
4	R4	X -> E	Mass action (irreversible)	0.5	
	New Reaction				

Figure 2.1: COPASI: A stand-alone software in systems biology.

EasyModel draws inspiration from various software tools, including COPASI, but it distinguishes itself through several key differences. While COPASI offers a robust platform for simulating and analyzing biochemical models, EasyModel places a stronger emphasis on user-friendliness, particularly through its web-based interface. This design choice makes EasyModel more accessible to a broader audience, especially those who may not have extensive experience with traditional simulation software.

One of the primary distinctions is that EasyModel leverages the Mathematica kernel for performing complex calculus operations, which allows for more sophisticated mathematical modeling. This integration with Mathematica not only enhances the computational power of EasyModel but also provides users with the unique ability to download the generated Mathematica code. This feature is particularly valuable for advanced users who wish to customize their simulations and analyses beyond the standard capabilities of the software. In contrast, COPASI does not offer this level of flexibility, as it does not support exporting or modifying the underlying simulation code in a similar manner.

Thus, while both tools are powerful in their own right, EasyModel is specifically designed to combine ease of use with the advanced analytical capabilities provided by Mathematica, making it a versatile tool for both novice and expert users.

Another noteworthy software is Virtual Cell (29), which also connects to an Internet simulation server. Unlike EasyModel, Virtual Cell is a desktop application (Windows/Linux/Mac OS X) that must be installed locally, though the actual simulations are performed remotely on a server. It supports multi-compartmentalization, offers a diagrammatic user interface for graphical model definition, and provides features like stochastic simulation. Virtual Cell offers free user accounts for storing biological models and facilitates collaboration through shared project access. In contrast, EasyModel does not require local installation, sharing the remote server simulation approach and user account access. Additionally, EasyModel offers more comprehensive assistance throughout the modeling process.

PLAS (30) (Power Law Analysis and Simulation)) is a simulation tool for Windows that offers similar functionality to EasyModel. It features a plain text editor where users input the various stages of model construction, including transformations (dynamic variables and constants definition) and differential equations, across multiple lines. Like EasyModel, PLAS is well-suited for educational purposes, as it requires users to define and understand each step of the modeling process. However, EasyModel enhances usability by providing a graphical user interface (GUI) instead of a text editor, making the modeling process more intuitive and accessible.

The applications mentioned above are standalone and do not depend on powerful calculation engines like Mathematica (1), MAPLE (5), or MATLAB (31). This limitation restricts the level of customization available in these tools. In contrast, applications integrated with Mathematica, MAPLE, or MATLAB can leverage the advanced programming languages and predefined calculation engines provided by these platforms, allowing for greater flexibility in model building and analysis.

For example, the Systems Biology Toolbox (24) for MATLAB facilitates the analysis and simulation of biological and biochemical systems. The user interface is primarily text-based, relying on a command line or text editor. The toolbox includes features such as network identification, sensitivity analysis, and bifurcation analysis. Researchers can also extend its capabilities by writing custom scripts in MATLAB. It supports various methods for deterministic simulations as well as exact and approximate stochastic simulations. However, the Systems Biology Toolbox requires users to have proficiency in MATLAB programming. Similar tools are available for Mathematica and Maple, but they share the same limitations.

In contrast, EasyModel focuses on delivering a user-friendly interface, making it accessible even to those without prior knowledge of Mathematica.

2.2 Comparing EasyModel with other web-based tools

In contrast to stand-alone applications, web applications in the realm of modeling and simulation in systems biology are specifically designed to operate within a web browser and generally depend on server-side components or cloud services to function effectively. Unlike stand-alone applications, which are self-contained and run locally on a user's device, web applications rely on continuous network connectivity to access and interact with server-based resources.

In the context of systems biology, web-based modeling and simulation tools often utilize server-side components to perform complex computations, manage large datasets, and handle intricate simulations that might be too resource-intensive for local machines. These server-side components are responsible for various tasks, including data processing, storage, and user authentication, all of which are crucial for the web application's core functionality. For instance, these components may process and analyze large-scale biological data, store simulation results, and maintain user accounts and permissions.

In the following paragraphs, EasyModel will be compared to other web-based tools with similar and related features.

A notable tool in systems biology worth highlighting is JWS Online (6, 32) (see Figure 2.2). Similar to EasyModel, JWS Online uses Mathematica as its underlying calculus engine and supports a range of simulations and analyses. The tool offers comprehensive features, including databases for users, models, simulations, and manuscripts, and it supports SBML, just like EasyModel. Its user interface is robust and user-friendly, though it can be overwhelming for beginners. The availability of online documentation is a significant advantage.

JWS Online also includes features that EasyModel currently lacks, such as

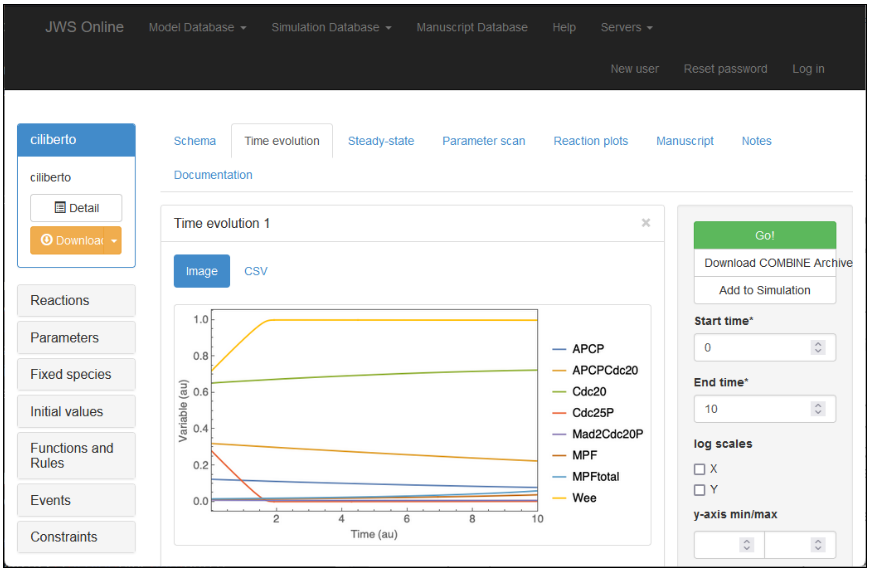


Figure 2.2: JWS Online: A web-based application in systems biology.

visual representation of model schemas and reaction plots. It is an incredibly valuable resource for biologists. However, EasyModel offers unique advantages, particularly for novice users, by easing and accelerating the learning curve. Additionally, EasyModel offers advanced features not available in JWS Online, such as stochastic simulations, kinetic laws derived from mathematical formalisms, and the ability to modify and simulate models from its public database. While JWS Online provides a public database of curated models, it does not allow users to modify these models for subsequent simulations, a capability that EasyModel supports.

Overall, both JWS Online and EasyModel are powerful tools, each catering to different audiences and needs in the field of systems biology.

Another noteworthy simulation tool is the APMonitor Optimization Suite (33, 34, 35), which adopts a different approach to solving problems. Unlike EasyModel, APMonitor uses its own modeling language, allowing users to define models in a plain text editor, similar to the previously discussed PLAS software. This approach provides significant flexibility, enabling programmers to introduce new features more easily and focus heavily on expanding the tool's capabilities through coding.

Such tools are highly beneficial for expert users, offering a broad range of solutions. However, this flexibility comes at the cost of accessibility; the complexity and extensive feature set result in a steeper learning curve compared to a tool like EasyModel. Despite this, APMonitor compensates with comprehensive online documentation and a series of webinars designed to educate its users over time. The software is freely available online and can also be accessed as a MATLAB toolbox and a Python package. Additionally, the developers offer the GEKKO Python package for dynamic optimization.

Both EasyModel and APMonitor serve distinct user bases effectively. EasyModel excels in user-friendliness and accessibility, making it ideal for users with less programming experience, while APMonitor provides powerful, flexible tools for advanced users who require a more customizable environment.

Cellware (36) is a versatile simulation tool designed for modeling and simulating both deterministic and stochastic cellular events. It effectively handles the complexity of cellular processes by offering a broad range of simulation methods. The tool features an intuitive diagrammatic graphical user interface, making it accessible to users at various expertise levels. Cellware's computational tasks are managed through a grid computing environment, leveraging network resources rather than local hardware. Developed in Java, it is compatible with Windows, UNIX, and macOS. Its last release dates back to 2005, which may lead to compatibility issues with modern operating systems. Additionally, the official website is no longer available, but we wanted to mention Cellware due to its relevance as a comparative tool to EasyModel.

3 Methods

Technology is best when it becomes invisible, empowering us to achieve what once seemed impossible without being noticed.

(Alan Kay)

This chapter will provide an in-depth overview of EasyModel, including its main features, application capabilities, technical implementation details, mathematical modeling notation, version history, availability and more.

3.1 EasyModel features

The most important implemented features of EasyModel are summarized in Table 3.1.

These features have been formulated following the objectives that we set for this project after analyzing the state of the art of similar applications, conducted in the chapter 2, Related Work.

Global Features	
Type of application	Web-based application. Users do not need to install any application on their computers. More convenient for users.
User Interface	Graphical User Interface. Modern and sleek UI powered by Vaadin web framework. User-friendly experience for both novel and expert users. Low number of clicks to get to any functionality. Step-oriented—from modeling to simulation—design. Tutorial for beginner users. Help button available across the application.
Calculus engine	Wolfram Mathematica PMC Downloadable Mathematica notebook of model and simulation. Only one Mathematica kernel can be executed concurrently, so a simulation job queue is used. Mathematical formalisms can be defined within the Mathematica language.
Models storage	MySQL database. Stores pre-implemented models and registered users information. Predefined formulas available.
Tool Views	
Tutorial	A tutorial explaining all the basic use steps.
Model selector	Starting page and first step. New users are advised to read the tutorial. Select from public-available models or create a new one. SBML file format model import.
Model builder	Definition of model name and description. Definition of reaction processes. Definition of species and variable types. Definition of kinetic rate laws. Import predefined rate laws into the current model. Assign rate laws with parameters to reactions.
Simulation configurator	Dynamic simulation. Species evolution over a period of time. Steady State simulation. Seek of system's equilibrium. Stochastic simulation. Species evolution over a period of time. Analyses. Gains, sensitivities, stability and parameter scan. Configuration of plot views based on dependent variables. Plot settings to define general graphics configuration.
Simulation results	Results displayed in real-time. Simulation cancelable at any time. Launched simulations control panel. Simulations downloadable as graphics and text files. Download of Mathematica notebook and model SBML file. Results can be shared via hyperlinks.
Administration panel	Restricted to administrators. Administration of user, model and formula data entries.

Table 3.1: EasyModel's main features.

3.2 EasyModel implementation design

Figure 3.1 summarizes the overall EasyModel architecture.

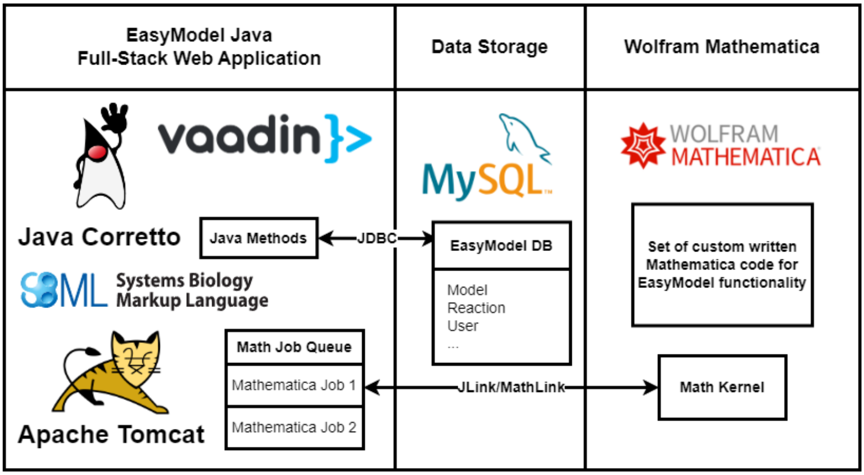


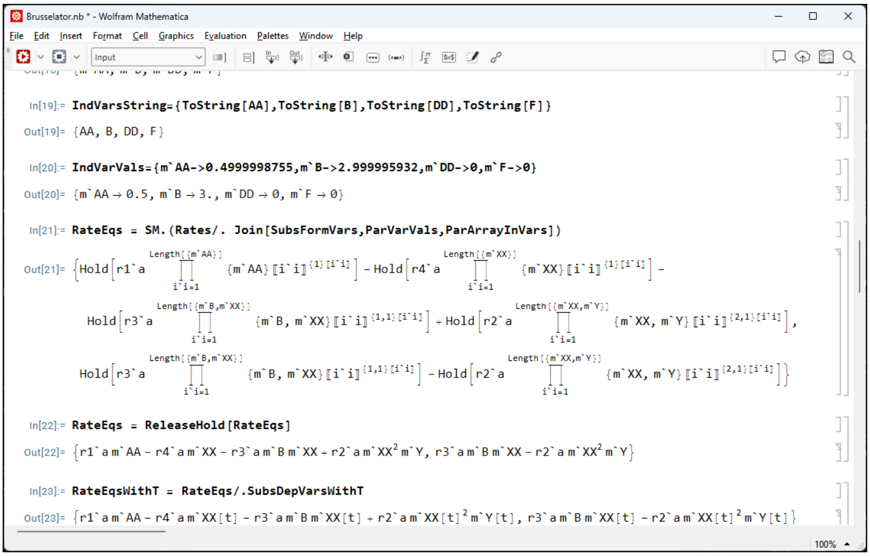
Figure 3.1: EasyModel implementation architecture.

EasyModel has been built mainly over the Jakarta EE (37) programming environment. Jakarta EE is the evolution of Java EE (Java Platform, Enterprise Edition), a set of specifications that extend the Java SE (Standard Edition) platform to provide a framework for developing robust, scalable, and secure enterprise-level applications. After Oracle donated Java EE to the Eclipse Foundation in 2017, the platform was rebranded as Jakarta EE, marking a new phase in its development with a focus on community-driven innovation.

To build the web user interface, we chose the Vaadin (38) Flow web user framework for Java. Vaadin Flow is a modern web framework for building rich, interactive web applications entirely in Java. Unlike traditional web development approaches that require developers to work with multiple languages (such as HTML, CSS, and JavaScript) for the front end and Java for the back end, Vaadin Flow allows developers to create the entire web application using only Java. This approach simplifies the development process and enables developers to focus on writing business logic rather than dealing with the complexities of front-end development.

Vaadin Flow is considered a Full Stack Web Framework. A full stack application is one that encompasses both the front-end (client-side) and back-end (server-side) aspects of a web application. This means it handles everything from the user interface and experience (UI/UX) to the underlying server logic, database interactions, and server-side processing.

For performing the calculations for the simulations and analyses we chose the Wolfram Mathematica (1) PMC (see Figure 3.2). Mathematica is a proprietary software system developed by Wolfram Research, designed for symbolic computation, numerical analysis, data visualization, and more. It is widely used in various scientific, engineering, mathematical, and computing fields for complex calculations and simulations. Mathematica offers a vast range of built-in functions and algorithms that cover areas such as algebra, calculus, graph theory, machine learning, and statistical analysis.


Brusselator.nb * - Wolfram Mathematica
File Edit Insert Format Cell Graphics Evaluation Palettes Window Help
Input
100%

```
In[19]:= IndVarsString={ToString[AA],ToString[B],ToString[DD],ToString[F]}
Out[19]= {AA, B, DD, F}

In[20]:= IndVarVals={m`AA->0.4999998755,m`B->2.999995932,m`DD->0,m`F->0}
Out[20]= {m`AA → 0.5, m`B → 3., m`DD → 0, m`F → 0}

In[21]:= RateEqs = SM.(Rates/. Join[SubsFormVars,ParVarVals,ParArrayInVars])
                  Length[{m`AA}]                                   Length[{m`XX}]
Out[21]= {Hold[r1`a     ∏        {m`AA}[[i`i]]^{1}[[i`i]]] - Hold[r4`a     ∏        {m`XX}[[i`i]]^{1}[[i`i]]] -
                       i`i=1                                            i`i=1
                 Length[{m`B,m`XX}]                                       Length[{m`XX,m`Y}]
         Hold[r3`a     ∏        {m`B, m`XX}[[i`i]]^{1,1}[[i`i]]] + Hold[r2`a     ∏        {m`XX, m`Y}[[i`i]]^{2,1}[[i`i]]],
                      i`i=1                                                     i`i=1
                 Length[{m`B,m`XX}]                                       Length[{m`XX,m`Y}]
         Hold[r3`a     ∏        {m`B, m`XX}[[i`i]]^{1,1}[[i`i]]] - Hold[r2`a     ∏        {m`XX, m`Y}[[i`i]]^{2,1}[[i`i]]]}
                      i`i=1                                                     i`i=1

In[22]:= RateEqs = ReleaseHold[RateEqs]
Out[22]= {r1`a m`AA - r4`a m`XX - r3`a m`B m`XX + r2`a m`XX^2 m`Y, r3`a m`B m`XX - r2`a m`XX^2 m`Y}

In[23]:= RateEqsWithT = RateEqs/.SubsDepVarsWithT
Out[23]= {r1`a m`AA - r4`a m`XX[t] - r3`a m`B m`XX[t] + r2`a m`XX[t]^2 m`Y[t], r3`a m`B m`XX[t] - r2`a m`XX[t]^2 m`Y[t]}
```

Figure 3.2: Wolfram Mathematica frontend on Microsoft Windows.

One of the key features of Mathematica is its ability to perform symbolic computations, which allows it to manipulate mathematical expressions in symbolic form, such as solving equations analytically or simplifying complex expressions. This makes it a powerful tool for both theoretical research and practical applications. In the case of EasyModel, this was a crucial factor, as it enabled us to execute a wide range of complex mathematical operations essential for the advanced calculations typically required in systems biology. Despite not being open-source software, we chose it because it offers extensive functionality and is widely recognized and utilized in the academic community we aim to serve.

We chose Vaadin because it is ideal for building user-friendly web interfaces and integrates seamlessly into our Java application, which needs to connect with Mathematica. Opting for a web application approach offers significant advantages: It frees users from needing a personal Mathematica license, which would be required if this were a locally installed application. Additionally, it alleviates the need for users to own a powerful computer capable of handling Mathematica's computational demands, although this concern is diminishing as technology becomes more affordable.

However, this approach comes with a notable limitation—only one user can run simulations at a time. While EasyModel's current user base is not large enough for this to be a pressing issue, it could become problematic if the user base grows. Addressing this limitation would require an expensive license, but for now, our current setup is adequate.

It's important to note that we strictly adhere to Wolfram's policies; users are not allowed to execute arbitrary code on our Mathematica license. They are only permitted to run predefined Mathematica code for simulations. The connection between the Java application and Mathematica is facilitated using Wolfram's proprietary JLink library, which enables communication with the Mathematica kernel for both sending and receiving data. To optimize data exchange between Java and Mathematica, we batch data into large chunks, which helps to improve execution speed.

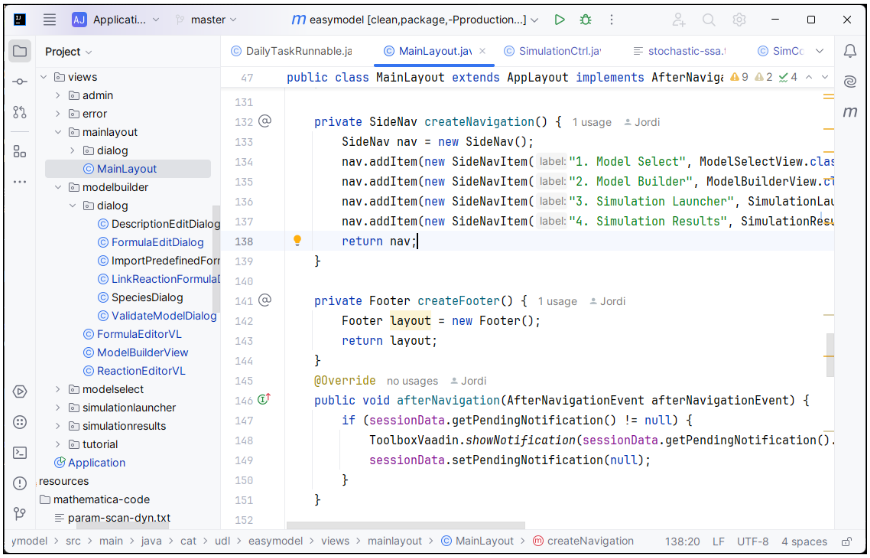
To address the limitation of having only one concurrent Mathematica kernel, we developed a simulation job queue system to enhance user experience by eliminating the need to wait for kernel availability. This issue was a significant drawback in previous versions, impacting the tool's effectiveness. In the latest version, we implemented a simulation job queue system within the Java code. This system operates through a dedicated Java thread running parallel to the main web application execution thread.

While the main application processes user web requests, the simulation job queue independently handles incoming jobs. When a new job is detected, the system spawns a new thread to execute the simulation on the Mathematica kernel. Although this solution is complex, it was deemed essential for improving application usability. The implementation has proven to be highly effective, significantly enhancing the tool's performance and user experience.

To store the model and user data, we utilized the MySQL Community (39) database manager. This open-source database manager is renowned for its reliability and performance, offering robust support for a range of data storage and retrieval needs. Its compatibility with various platforms and programming languages makes it a versatile choice for managing the application's data, ensuring efficient handling of user interactions and model storage.

The application has been primarily coded in Java, though other languages have also been utilized, including Mathematica's symbolic language, MySQL query language, HTML for markup, and CSS for styling. These languages collectively support the development of EasyModel.

The development environment chosen for this project was the JetBrains IntelliJ IDEA (40), which provided a powerful and versatile Integrated Development Environment (IDE) for Java programming (see Figure 3.3). Coupled with the Amazon Corretto 17 OpenJDK Java Platform (41), this setup ensured a stable and up-to-date Java development environment. Maven (42) was used as the dependency manager for the web application. For web application deployment, we utilized the Apache Tomcat (43) web application server. To test the Mathematica code, we used the Wolfram Mathematica (1) version 14 in graphical mode. To test the database environment we used the Laragon (44) universal development environment which includes a MySQL server and the possibility to deploy a *phpmyadmin* local website to manage the MySQL databases.



```java
public class MainLayout extends AppLayout implements AfterNaviga

    private SideNav createNavigation() {  1 usage  Jordi
        SideNav nav = new SideNav();
        nav.addItem(new SideNavItem( label: "1. Model Select", ModelSelectView.clas
        nav.addItem(new SideNavItem( label: "2. Model Builder", ModelBuilderView.cl
        nav.addItem(new SideNavItem( label: "3. Simulation Launcher", SimulationLau
        nav.addItem(new SideNavItem( label: "4. Simulation Results", SimulationResu
        return nav;
    }

    private Footer createFooter() {  1 usage  Jordi
        Footer layout = new Footer();
        return layout;
    }
    @Override  no usages  Jordi
    public void afterNavigation(AfterNavigationEvent afterNavigationEvent) {
        if (sessionData.getPendingNotification() != null) {
            ToolboxVaadin.showNotification(sessionData.getPendingNotification().
            sessionData.setPendingNotification(null);
        }
    }
```

Figure 3.3: JetBrains IntelliJ IDEA on Microsoft Windows: The current Java IDE used in the development of EasyModel.

In the initial stages of developing EasyModel, we used the Eclipse Java EE IDE (45) instead of IntelliJ IDEA and employed Vaadin 6 rather than the current Vaadin 24. The migration from Vaadin 8 to Vaadin 24 required a substantial amount of work, as it involved a complete revision and rewriting of the UI code to accommodate the updated Vaadin 24 coding syntax. Despite the significant investment of time and effort, we consider the upgrade to Vaadin 24 to have been worthwhile, as it greatly improved the application's functionality and user experience.

To implement the SBML (46, 47) file format specification for model exchange, the JSBML (48) Java library was used. SBML will be explained in the following sections.

To deploy the EasyModel web application, the following requirements must be met: a Java Runtime Environment, a MySQL Database Server, and Wolfram Mathematica software with a valid license. The application has been successfully tested for deployment on both Linux and Microsoft Windows operating systems.

3.3 Modeling

EasyModel employs a conceptual representation of molecular biology networks, enabling users to model and analyze complex biological systems effectively. In this representation, networks are broken down into individual reactions or processes that involve the consumption of substrates, the generation of products, and the modulation of reaction rates by modifiers such as activators or inhibitors. Each reaction's speed rate is governed by an associated kinetic rate law, which governs the rate at which the reaction proceeds. The integration of all these elements conforms a mathematical model of the biological network of interest (see Figure 3.4).

Users provide the program with the individual reactions using the notation from the Equation 3.1.

$$n_1 * S_1 + \ldots - > m_1 * P_1 + \ldots ; M_1; \ldots \tag{3.1}$$

In the Equation 3.1, S_i refer to the substrate species (left-hand side of the arrow). n_i and m_i refer to the stoichiometric coefficients for the substrates and products, respectively. Finally, M_i refer to the modifier species (found at the end and separated by a semicolon character).

A kinetic rate law needs to be associated to each of the reactions defined for the model. These rate laws are mathematical functions that define the pace at which reactions will occur. When associating rate laws to reactions, rate parameters must be defined. Rate parameters can take either numerical values or be linked to model species that behave like mathematical variables.

EasyModel offers predefined standard formalisms (Mass Action, Power Law, Saturating, Saturating Cooperative, Henri-Michaelis Menten, Hill Cooperativity, Catalytic Activation and Competitive Inhibition and more (49)) that users

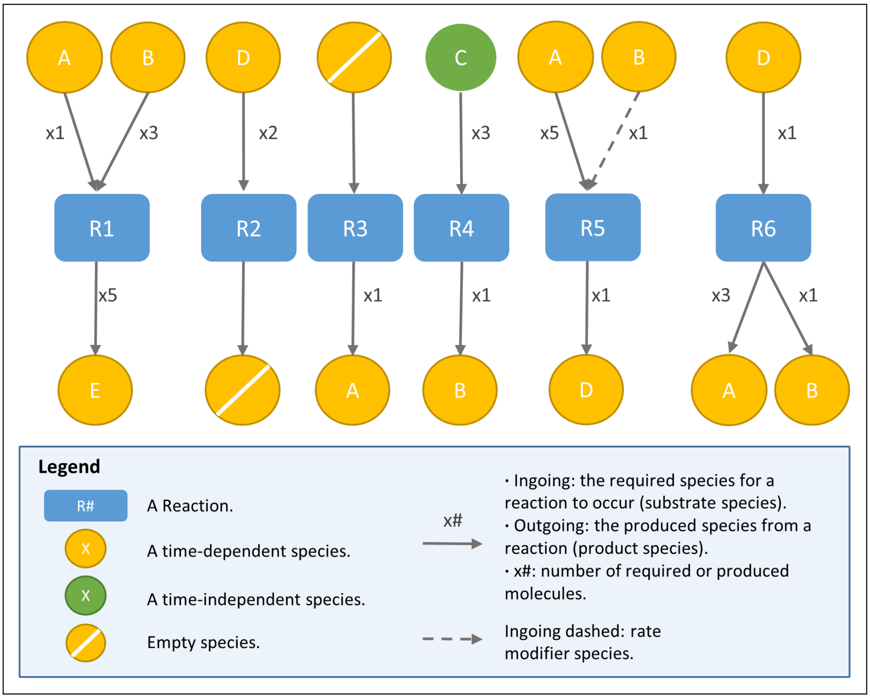

A
B
D
C
A
B
D
x1
x3
x2
x3
x5
x1
x1
R1
R2
R3
R4
R5
R6
x5
x1
x1
x1
x3
x1
E
A
B
D
A
B
Legend
R#
A Reaction.
X
A time-dependent species.
X
A time-independent species.
Empty species.
x#
• Ingoing: the required species for a reaction to occur (substrate species).
• Outgoing: the produced species from a reaction (product species).
• x#: number of required or produced molecules.
Ingoing dashed: rate modifier species.

Figure 3.4: A generic reactions diagram of a systems biology model.

can import into the model and use without the need to write functional formulas.

Users can also create custom made formulas through the user interface. To define custom rates and mathematical formalisms, see the following "How to define rate expressions" text outline.

```
How to define rate expressions
  Usable operators: +-/*^()
  Reserved symbols:
    Mathematica functions/constants: m:<Mathematica function>
    Mathematica function indexes: i:<index>
    Special variables:
      b:t -> Time.
      b:X[] -> Mathematica substrate list.
      b:A[] -> Mathematica substrate coefficient list.
      b:M[] -> Mathematica modifier list.
      b:XF -> First substrate.
      b:MF -> First modifier.
  Example: m:Product[b:X[[i:j]]^g[[i:j]],{i:j,1,m:Length[b:X]}]
```

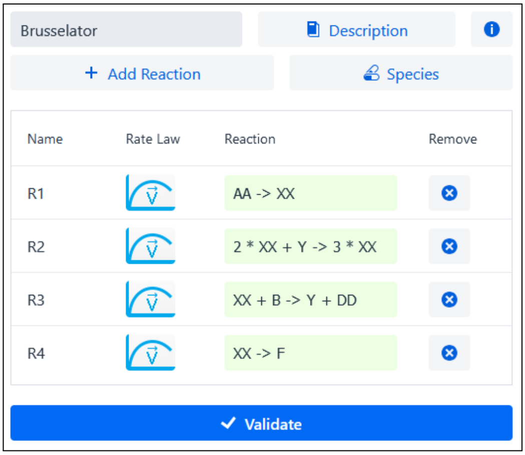
Formulas definition also allow to set up reaction linking restrictions. This means the formula will not be able to be bound to a reaction that doesn't comply with the formula restrictions. Possible restrictions that can be applied to formulas are described in the next text outline.

```
Optional formula restrictions
  -One substrate only.
  -No products.
  -One modifier only.
```

The species of a model can either be time-dependent (dependent variables) or time-independent (independent or control variables). By default EasyModel makes every variable dependent and attributes it an initial value of 1 concentration units. Users can then change the initial value and type of the variables.

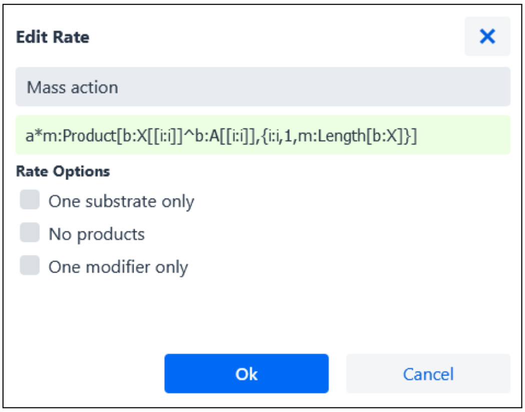
EasyModel doesn't so far support multi-compartmentalization, so it is responsibility of the user to adjust the models manually if more than one compartment is considered.

Regarding the EasyModel model editor user interface (see Figure 3.5): It is a combination of explicit textual interface and dialog boxes. Models are defined in text mode, that is, users define reactions line by line using the reaction syntax. Rate laws are also defined in text mode (see Figure 3.6).



Name	Rate Law	Reaction	Remove
R1		AA -> XX	
R2		2 * XX + Y -> 3 * XX	
R3		XX + B -> Y + DD	
R4		XX -> F	

Figure 3.5: EasyModel: Defining reactions in a model.


Edit Rate

Mass action

```
a*m:Product[b:X[[i:i]]^b:A[[i:i]],{i:i,1,m:Length[b:X]}]
```

Rate Options

- [ ] One substrate only
- [ ] No products
- [ ] One modifier only

Figure 3.6: EasyModel: Defining a custom formula in a model.

3.4 Types of simulation and analysis

Table 3.2 shows the different simulations and analysis supported by EasyModel.

Deterministic Simulations	
Dynamic simulation	Time evolution of the system.
Steady State simulation	Search for the steady state of the system. Stability analysis available.
Gains and Sensitivities	Calculated for rate parameters and independent variables. Available for dynamic and steady state simulations.
Parameter Scan	Calculated for rate parameters and independent variables. Available for dynamic and steady state simulations.
Stochastic Simulations	
SSA method	Exact numerical solution of the time evolution of the system. Not available for all models.
τ-leaping method	Approximated solution to reduce the CPU time of the SSA method. Not all models can benefit from it.

Table 3.2: EasyModel's types of simulations.

3.4.1 Deterministic regime

Deterministic simulations numerically simulate ODE systems using deterministic algorithms (50). These simulations provide accurate descriptions of systems that have a large amount of molecules.

An ODE, or Ordinary Differential Equation, is a mathematical equation that describes the relationship between a function and its derivatives. In simpler terms, it relates a dependent variable (often representing a physical quantity like position, velocity, or concentration) to its rate of change with respect to an independent variable, typically time. In the case of EasyModel, we take the dependent variable to be the species concentrations and the independent variable to be time.

Performing **dynamic** simulations generates concentration plots describing the time course of the various dependent variables of the system from an initial time (usually zero) to a final simulation time decided by the user.

Performing a **steady state** simulation directly calculates the non-trivial equilibrium steady state of the system. This is done using Mathematica's default algorithms to find the roots of the ODE system. To account for pathological failures of the root finding algorithms, if no roots are found, a dynamic simulation with a final time of 500000 time units is performed. The final concentrations for this simulation are then inserted into the rate equations and tested to see if a steady state is reached (see Algorithm 1).

Algorithm 1 Steady State Algorithm

```
1: **Input:** Rate Equations and Model ODE
2: **Output:** Steady State Concentrations
3: procedure CalculateSteadyState(RateEquations, ODE)
4:     $Root = FindRoot(RateEquations = 0, Random)$
5:     if $Root \leq threshold$ then
6:         $SteadyState = Root$
7:     else
8:         $t = 500000$
9:         $SolvedODE = SolveODE(ODE, t-1, t+1)$
10:        $Root = FindRoot(RateEquations = 0, SolvedODE(t))$
11:        if $Root \leq threshold$ then
12:            $SteadyState = Root$
13:        else
14:            $SteadyState =$ empty
15:        end if
16:    end if
17:    return $SteadyState$
18: end procedure
```

To perform steady state **stability analysis**, EasyModel determines the local stability of a steady state by calculating the eigenvalues for the Jacobean matrix of the system at steady state and testing to see if all real parts of those eigenvalues are negative (stable steady state) or not (51).

Local **gains and sensitivities** analyses are calculated as described in (52). In short, this analyses measures the effect of changing a control variable (gains) or parameter (sensitivities) on the dynamic or steady state behavior of the system without changing those parameter directly. Both magnitudes can be calculated for Dynamic and Steady State simulations.

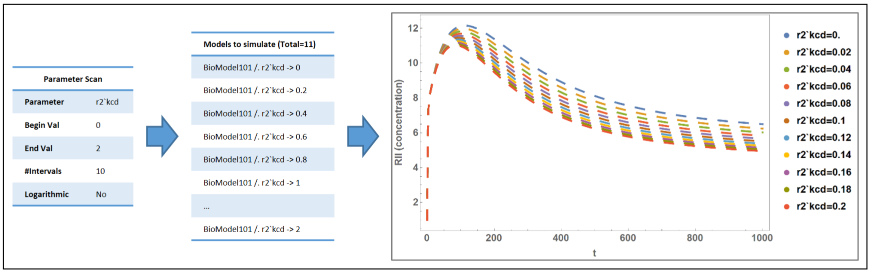
Parameter scan allows to conduct a global sensitivity analysis (see Figure 3.7). This functionality allows the user to directly observe the effect of changing a parameter or control variable on the dynamic behavior of the system. To do so the user selects the parameter (or control variable) that will be analyzed and defines the minimum and maximum values that it can assume. When running the simulation, EasyModel will perform the selected basic simulation (e.g. Dynamic simulation) and after that it will perform the Parameter Scan by systematically launching multiple versions of the same base simulation. These simulations will be performed individually by changing the parameter value between its minimum and maximum values. The range of values is divided into a user-defined number of intervals, using either a linear or a logarithmic scale. The user may select multiple parameters to scan at once and they will be displayed in the results separated by two factors: by dependent variables and by selected parameters to scan. This display separation makes it easier for the user to identify the changes in the model.

Parameter scanning is available for both dynamic and steady-state simulations in the deterministic regime. The Parameter Scan is executed in parallel across CPU cores for each parameter value, helping to reduce overall execution time. Implementing Parameter Scan for stochastic simulations is feasible but would require significantly more time compared to deterministic simulations.

3.4.2 Stochastic regime

Stochastic simulations were introduced into EasyModel after the deterministic simulations. This type of simulation still numerically solves ODE systems that describe the dynamic behavior of molecular networks. However, the numerical solution relies on using a stochastic approximation instead of a deterministic one. In very simple terms, only one process or reaction is executed in each simulation step. The decision regarding which process is fired relies on a random number that is generated in each integration step, used in combination with the rate functions. This is a more accurate simulation procedure for molecular systems with a small number of molecules (53). Still, it requires significantly more computational time to be performed than the equivalent deterministic simulation. Not all models can be simulated in the stochastic regime, EasyModel will warn the user if that is the case.

The simulation engine uses deterministic simulation algorithms to solve



Parameter Scan	
Parameter	r2`kcd
Begin Val	0
End Val	2
#Intervals	10
Logarithmic	No

Models to simulate (Total=11)
BioModel101 /. r2`kcd -> 0
BioModel101 /. r2`kcd -> 0.2
BioModel101 /. r2`kcd -> 0.4
BioModel101 /. r2`kcd -> 0.6
BioModel101 /. r2`kcd -> 0.8
BioModel101 /. r2`kcd -> 1
...
BioModel101 /. r2`kcd -> 2

Figure 3.7: An example of parameter scanning using *BioModel 101*. We change the value of parameter *kcd* of the second reaction between 0 and 2, at intervals of 0.2. EasyModel performs a dynamic simulation for each single value in the scanning range, enabling our understanding of how changing the parameter value will change systemic behavior. The plot on the right side of the figure illustrates this effect for variable *RII*.

differential equation models. This kind of simulation is less computational-demanding than the ones done using more realistic stochastic simulation algorithms such as the Gillespie algorithm. However, as a rule of thumb one should be aware that differences in the results of the simulations are only noticeable when less than 1000 molecules are involved in the simulation.

Stochastic simulation algorithms were implemented by us, as no such predefined algorithm is available in Mathematica. First, we implemented and thoroughly tested the **Gillespie's Stochastic Simulation Algorithm (SSA)** (54, 55) by comparing the results of simulating the same model with deterministic and stochastic simulation algorithms (see Figure 3.8). This algorithm provides accurate simulations at the cost of an increase of the computational time.

In deterministic simulations, a system starting from a specific set of initial conditions will always follow the same dynamic behavior, regardless of how many times the simulation is repeated. In contrast, each run of a stochastic simulation produces a unique time trajectory. Due to this variability, stochastic simulations must be repeated multiple times—across several replicates—to ensure that none of the individual trajectories deviate significantly from the median behavior of the system. To reduce execution time, EasyModel executes these independent replicates using parallel computing threads. The more replicates are performed, the more precise and smooth the plot becomes (see Figure 3.9).

Typically, the more replicates are plotted, the closer the plotted median trajectory aligns with its deterministic counterpart. However, stochastic simu-

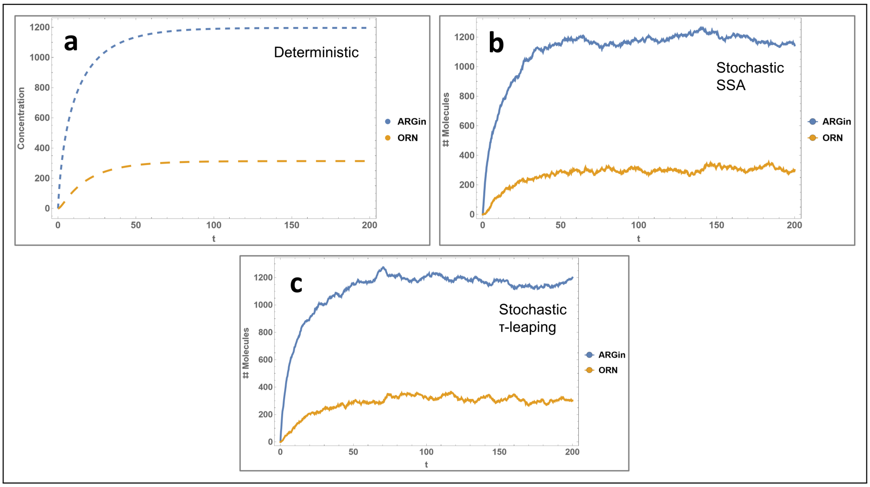


Figure 3.8: A comparison of the results produced by deterministic and stochastic simulations on the *BioModel 191 Arginine catabolism* model. In this model, species follow a similar trajectory in all the different simulation methods. **a** deterministic ODE solve method. **b** stochastic SSA method. One stochastic replicate.**c** stochastic τ-leaping method. One stochastic replicate.

lations do not always reproduce the same deterministic behavior. In fact, they can provide different and potentially more accurate representations, especially in simulations involving fewer than 1000 molecules.

Deterministic simulations rely on the assumption that the number of molecules involved is sufficiently large, allowing stochastic fluctuations to be ignored and concentrations to be treated as continuous variables. It is also important to note that EasyModel does not yet support spatially non-homogeneous models, which would require the use of partial differential equations (PDEs).

EasyModel works by default with concentrations. This is fine to operate within the deterministic regime. However, stochastic simulations operate using number of molecules. To account for this, EasyModel performs a numerical translation based on a selectable cell size, which can be configured before starting the simulation.

Regarding plotting of stochastic simulations, individual plots are generated for each dependent variable, displaying the time course for all replicate curves of that variable.

In addition, EasyModel implements **stochastic simulation linear noise**

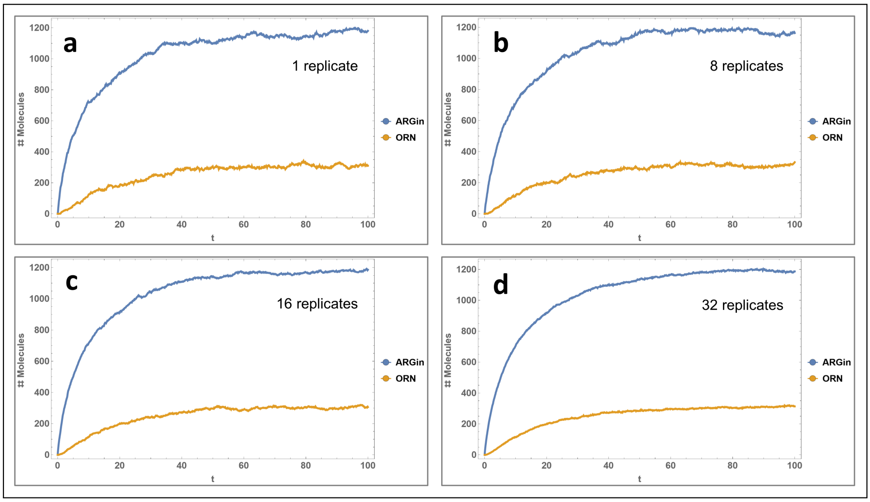


Figure 3.9: A comparison of the results produced by the SSA stochastic method on the *BioModel 191 Arginine catabolism* model using different number of replicates. **a** Plot produced with one stochastic replicate. **b** Plot produced with eight stochastic replicates. **c** Plot produced with sixteen stochastic replicates. **d** Plot produced with thirty two stochastic replicates.

analysis (56) (see Figure 3.10). This analysis of the intrinsic noise of the system is a more appropriate tool than sensitivity analysis to understand the limitations and regulation of a molecular system working in the stochastic regime. EasyModel displays this analysis in the form of two graphics: One of them shows the 0.25, the median and the 0.75 quantile coefficient of dispersion, while the other one calculates the coefficient of variation according to parametric (standard deviation/median) and non-parametric ((Q0.75-Q0.25)/Median) approaches. EasyModel automatically generates separated noise analysis for each dependent variable of the system.

EasyModel is not the only tool capable of performing linear noise analysis in the stochastic regime. Other simulation applications, such as NetBioDyn (57) and StochPy (53), offer similar functionalities. However, these alternatives either lack the user-friendly features that EasyModel provides or employ different modeling methodologies. For example, NetBioDyn is a user-friendly agent-based simulator that uses different mathematical approaches than EasyModel. In contrast, StochPy is a Python package that requires programming

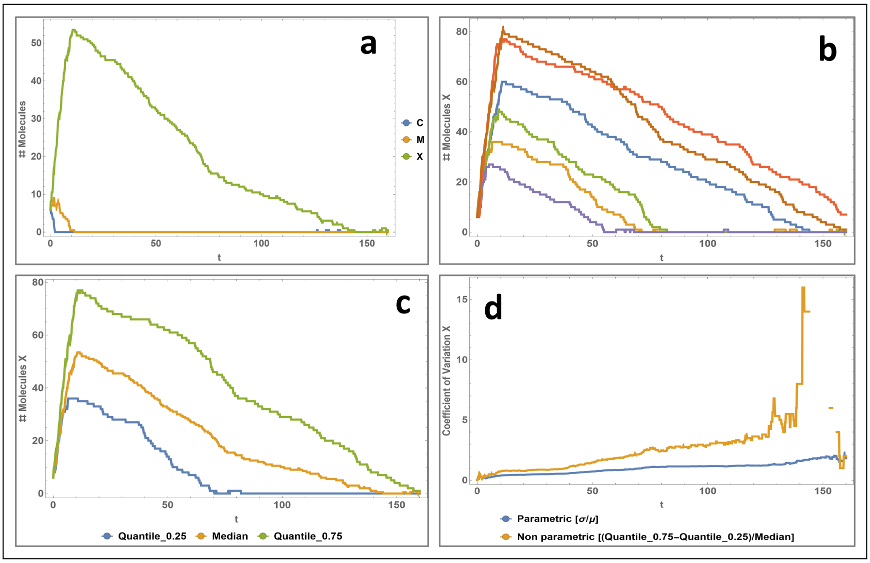


Figure 3.10: Example of a stochastic simulation run on *BioModel 3*. Simulation performs 6 replicates on the model using the SSA method. **a** medians of all the dependent variables of the model. **b** The 6 trajectories calculated for the time-dependent variable X. **c** Stochastic noise analysis for variable X: Quantile coefficients of dispersion. **d** Stochastic noise analysis for variable X: Coefficients of variation.

skills, making it less accessible to users without a coding background.

We also implemented a more efficient stochastic simulation algorithm known as the **τ-leaping method** (58, 59, 60, 61, 62, 63, 64). This algorithm potentially increases the simulation speed at the cost of an acceptable accuracy loss in the simulation results. Not all models can benefit from the increased efficiency of this algorithm compared to the SSA algorithm. This algorithm has undergone multiple revisions, resulting in several different versions. We implemented the "Efficient step size selection for the tau-leaping simulation method" version of the τ-leaping method (64).

3.4.2.1 Stochastic algorithms

The **Gillespie's Stochastic Simulation Algorithm (SSA)** (see Algorithm 2) operates by simulating the occurrence of individual reaction events in a system where randomness is inherent, typically in systems with low molecular counts. The algorithm calculates the time until the next reaction and determines which reaction will occur based on their respective probabilities (54, 55).

The SSA follows these steps:

1. **Calculate propensities**: For each reaction, compute the propensity function, which reflects how likely the reaction is to occur in a small time interval. This depends on the number of molecules and the reaction rates.

2. **Generate random numbers**: Two random numbers, r_1 and r_2, are generated from a uniform distribution between 0 and 1. These random numbers are used to determine the next event.

3. **Time step calculation**: The time to the next reaction event is calculated using the first random number r_1 and the total sum of propensities. The time step is computed as:

$$\Delta t = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$$

   where a_0 is the sum of all reaction propensities.

4. **Select reaction**: The second random number r_2 is used to select which reaction will occur, by comparing r_2 to cumulative propensities.

5. **Update system**: Once a reaction is selected, the state of the system is updated to reflect the occurrence of that reaction (e.g., molecule counts are modified).

6. **Repeat**: The process is repeated until the simulation reaches the desired final time or the system reaches equilibrium.

This algorithm captures the inherent randomness of molecular interactions in biochemical systems, making it especially useful for simulating biological reactions with small molecule counts where deterministic methods fail to capture the stochastic effects.

The second implemented stochastic algorithm is the **τ-leaping method** (see Algorithm 3), which improves computational efficiency by allowing multiple reactions to occur within a single time step, τ. Unlike the original Gillespie SSA, which simulates one reaction at a time, the τ-leaping method simulates several reactions simultaneously, significantly reducing computational cost.

The version implemented in EasyModel follows the improved τ-leaping method described in *Efficient step size selection for the τ-leaping simulation method* (64).

Algorithm 2 Stochastic Simulation Algorithm (SSA) - Gillespie Algorithm

```
1: Input: Set of $N$ molecular species $S = (S_1, \dots, S_N)$, $M$ chemical reaction channels $R = (R_1, \dots, S_M)$, initial number of molecules $X(0) = (X_1(0), \dots, X_N(0))$, final time $T_{max}$
2: Output: System state at each time step described by the vector $X(t) = (X_1(t), \dots, X_N(t)$ where $X_i(t)$ is the number of molecules of species $S_i$ in the system at time $t$
3: $t = 0$
4: while $t < T_{max}$ do
5:     Compute propensity functions $a_j(X)$ for each reaction $R_j$, $j = 1, \dots, M$, where X is the state of the system in the current time step
6:     Compute total propensity $a_0(X) = \sum_{j=1}^{M} a_j(X)$
7:     Generate two random numbers $r_1, r_2$ uniformly distributed in $(0, 1)$
8:     Compute time to next reaction $\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right)$
9:     Determine which reaction will occur: Find $j$ such that $\sum_{k=1}^{j-1} a_k(X) < r_2 \cdot a_0(X) \leq \sum_{k=1}^{j} a_k(X)$
10:    Update system state: $X = X + \Delta X_j$, where $\Delta X_j$ is the change vector for reaction $R_j$
11:    Update time: $t = t + \tau$
12: end while
13: return $X(t)$
```

This version dynamically adjusts the step size, τ, to optimize both speed and accuracy.

The algorithm operates as follows:

1. **Calculate propensities**: As in SSA, the propensity functions for each reaction are computed based on the current state of the system.

2. **Estimate the leap condition**: For each reaction, the algorithm estimates whether a leap step (with multiple reactions occurring) is valid, based on the system's current state. The key goal is to avoid taking too large of a leap, which would violate the assumptions about how propensities change.

3. **Determine optimal time step τ**: The time step τ is selected dynamically to ensure stability and accuracy. The efficient step size selection algorithm adjusts τ to balance between accuracy and speed, based on how rapidly the propensities are changing. This ensures that the leap condition holds within the time step.

4. **Poisson sampling**: The number of occurrences for each reaction within the time step τ is sampled from a Poisson distribution with a mean equal to the product of the reaction's propensity and τ:

$$k_i \sim \text{Poisson}(a_i \cdot \tau)$$

   where k_i is the number of times reaction i occurs, and a_i is the propensity of reaction i.

5. **Update system**: The system's state is updated by applying the effects of all reactions sampled during the time step τ. This results in a leap forward in time where multiple reactions occur simultaneously.

6. **Adjust τ and repeat**: Based on the updated system state, a new τ is calculated for the next step. The process repeats until the desired simulation end time is reached or the system reaches a steady state.

This improved version of the τ-leaping method offers an optimal trade-off between speed and accuracy, allowing larger time steps when possible, while dynamically reducing τ when necessary to maintain the validity of the simulation. It is particularly useful for systems where the reactions have large disparities in their propensities.

Algorithm 3 Efficient Step Size Tau-Leaping Algorithm

```
1: Input: Set of molecular species $S = (S_1, \dots, S_N)$, set of chemical reaction channels $R = (R_1, \dots, S_M)$, set of initial number of molecules $X(0) = (X_1(0), \dots, X_N(0))$, final time $T_{max}$
2: Output: System state at each time step described by the vector $X(t) = (X_1(t), \dots, X_N(t)$
3: Initialize time $t = 0$
4: while $t < T_{max}$ do
5:     Compute propensity functions $a_j(X)$ for each reaction $R_j$, $j = (1, \dots, M)$ and $a_0(X) = \sum_{j=1}^{M} a_j(X)$
6:     (1) Determine $J_c$ for critical reactions and $J_{ncr}$ for non-critical reactions. Any reaction $R_j$ with $a_j(x) > 0$ is deemed critical if $\min_{i\in[1,N];v_{ij}\leq 0}[\frac{x_i}{|v_{ij}|}] < 10$
7:     if (2) $J_{noncritical} \neq \emptyset$ then
8:         Compute $\tau' = \min_{i\in I_{noncritical}}\{\frac{max\{\in x_i/g_i,1\}}{|\hat{\mu}_i(x)|}, \frac{max\{\in x_i/g_i,1\}^2}{|\hat{\sigma}_i^2(x)|}\}$
9:     else
10:        Set $\tau' = \infty$
11:    end if
12:    if (3) $\tau' < \frac{10}{a_0(x)}$ then
13:        Abandon temporally the tau leaping method and execute 100 single-reactions SSA steps and return to step (1)
14:    end if
15:    (4) Compute $a_0^{critical}(x)$ and generate a second candidate time leap $\tau''$ as a sample of Exp($\frac{1}{a_0^{critical}(x))}$
16:    if (5a) $\tau' < \tau''$ then
17:        Set $\tau = \tau'$ and generate $k$ to indicate reactions $R_j$ that will be fired during this time leap. For critical reactions, set $k_j = 0$. For non-critical reactions, generate $k_j \sim$ Poisson$(a_j(x) \cdot \tau)$
18:    else if (5b) $\tau'' \leq \tau'$ then
19:        Set $\tau = \tau''$ and generate $k$. Set $k_j = 0$ for all critical reactions except for one selected with Random$(a_j(x)/a_0^c(x))$ set as $k_j = 1$. For non-critical reactions, generate $k_j \sim$ Poisson$(a_j(x) \cdot \tau)$
20:    end if
21:    if (6) any negative value in $x + \sum_j k_j v_j$ then
22:        Set $\tau' = \frac{\tau'}{2}$ and return to step (3)
23:    else
24:        Leap by replacing $t \leftarrow t + \tau$ and $x \leftarrow x + \sum_j k_j v_j$
25:    end if
26: end while
27: Return $X(t)$
```

3.4.2.2 Automatic stochastic method chose

Following EasyModel's user-friendly philosophy, we implemented a quick-to-execute internal control test that tentatively determines if a model can be simulated using stochastic algorithms or not. Then, if the system can be simulated using stochastic algorithms, EasyModel further tentatively determines if the model is more efficiently simulated using the τ-leaping algorithm or not.

This control test begins by executing 1000 simulation steps of the SSA algorithm. If the system does not move away from its initial conditions, the stochastic simulation is deemed not possible. If the simulation moves away from its initial condition, EasyModel calculates the total SSA execution time and the reached final simulation time after performing 1000 simulation steps.

Then, another stochastic simulation is performed using the τ-leaping method until it reaches the simulation final time reached during the SSA simulation. Once the τ-leaping pass is completed, we calculate the τ-*leaping boost* using the Equation 3.2.

$$\tau\text{-leaping}_{\text{boost}} = (\text{SSA}_{\text{execution time}})/\tau\text{-leaping}_{\text{execution time}}) - 1 \tag{3.2}$$

This rate value determines the gain in execution speed that the τ-leaping method has over the SSA method on a particular model. If the calculated gain is greater than zero, then EasyModel automatically deems the τ-leaping algorithm as the recommended stochastic simulation method.

A limitation of this test is that it only analyses the beginning part of the simulation. If performance gains are only achieved at later stages of the simulation, EasyModel may end up choosing the least efficient algorithm. Still, in all our tests with these and other models we found that usually τ-leaps are distributed along the simulation time course, which implies that such a selection procedure will choose the most efficient algorithm most of the time.

To improve efficiency, this test is only executed during the first time the model is simulated on the system. The result that indicates if the model can be simulated using the SSA or the τ-leaping method is stored in the database to avoid future calculations. Once the value is on the database, EasyModel uses this value to inform the user about the model's compatibility with the stochastic regime when selecting this option on the simulation configurator.

3.5 Mathematica

Mathematica is considered one of the leading tools in the field of computational mathematics and symbolic computation. Developed by Wolfram Research, it is widely used for mathematical modeling, scientific computing, data visualization, and algorithm development. Its comprehensive set of features and powerful capabilities make it a top choice for researchers, engineers, and scientists in various disciplines.

EasyModel benefits greatly from Mathematica's extensive capabilities. Mathematica's advanced computational engine enables EasyModel to handle complex mathematical operations and large-scale simulations with ease. Its symbolic computation abilities are crucial for manipulating and simplifying mathematical expressions analytically, which aids in deriving insights and performing precise calculations.

Despite the benefits, there are some drawbacks to using a platform like Mathematica for a project like this, which we have addressed:

- Cost: Mathematica requires a paid license, making it relatively expensive compared to other mathematical and computational tools. To address this, we developed the tool to be accessed through a network using a web application server. This approach ensures that only the owner of the web server needs to handle the licensing cost. However, there are certain limitations with this solution. For instance, only one Mathematica kernel can be executed concurrently, meaning that any additional requests must be placed in a waiting queue. To comply with Mathematica's licensing policies, EasyModel does not allow users to execute arbitrary Mathematica code to be executed on the server. Users can only execute a set of pre-established commands that have been programmed into EasyModel, specifically for the scope of our work.

- Steep learning curve: While powerful, Mathematica can be challenging to learn, especially for beginners who are not familiar with its syntax and programming style. We address this by implementing a user-friendly interface on the web application with the open-source Vaadin web user interface framework.

- Resource-intensive: Mathematica can be demanding on system resources, particularly for large-scale computations, which may require a powerful computer to run efficiently. We address this by deploying the application on a remote web server, so the user doesn't have to worry about this aspect.

- Closed ecosystem: Mathematica is a proprietary software, meaning users are limited to its ecosystem and may find it difficult to integrate with other open-source tools or languages. Fortunately, the Mathematica platform can be accessed through various libraries, such as the Java Link library, which we conveniently used to connect with our program written in Java. This library enables a Java program to connect with a Mathematica kernel—essentially a Mathematica command prompt console—that can exchange data in both text and binary formats.

As outlined, we successfully addressed the limitations associated with using a PMC like Mathematica, which reinforced its suitability as a viable option for the development of EasyModel.

EasyModel provides the capability to download the generated Mathematica code, enabling further customization with a local Mathematica instance. The code is downloaded as a Mathematica notebook and includes both the model implementation and the associated simulation.

3.6 Simulation job queue system

In EasyModel 2.4, we introduced the Simulation Job Queue System, a feature that was critical for improving the application's usability. Although previous versions of EasyModel were user-friendly, they had a significant limitation: Due to licensing restrictions, we could only use a single Mathematica kernel at a time. Moreover, user access to the Mathematica kernel was unmanaged, meaning only one user could execute a simulation at any given time. If a user attempted to run a simulation while the kernel was in use, they would receive a message like "Mathematica is busy, please try again later." This was highly inconvenient, as users had to manually retry launching the simulation without any guarantee of success.

To address this issue, we implemented a queue system (see Figure 3.11) that manages all simulation job requests to ensure none are left out. This solution required expertise in Java threading as well as a deep understanding of Vaadin's asynchronous *push* user interface updates.

The Simulation Job Queue System operates on a continously running thread. This thread continuously monitors the state of a *LinkedList* data structure which holds the received pending jobs submitted by the users. When a pending job is detected in the list, the system moves it to the *Running* state and removes it from the queue. Once the Mathematica task is completed, the queue system removes the job from the *Running* state and checks the list for any additional pending jobs. This ensures a smooth and efficient process for managing multiple simulation requests, significantly enhancing the overall user experience.

Each simulation job is managed by a dedicated thread manager. This thread manager is responsible for setting a timeout period and tracking the completion status of the simulation job. The actual thread that performs the Mathematica operations is launched last. This thread executes the mathematical operations by interacting with the Mathematica kernel through the JLink Java library. Simulation jobs can be completed either by the successful end of the task, by user cancellation, or by timeout.

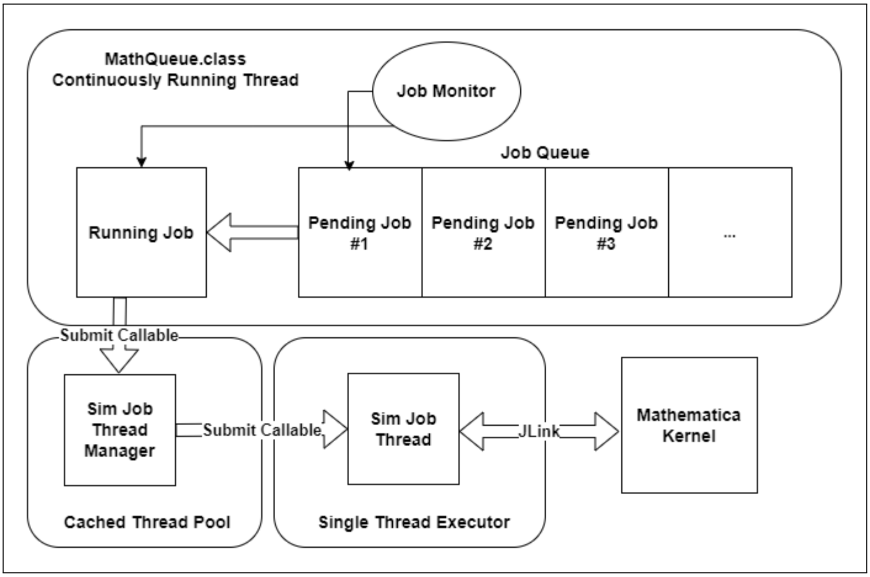


Figure 3.11: EasyModel's Simulation Job Queue System diagram.

3.7 Data storage

To store the application's persistent data, we employed the widely used MySQL database management system. We designed the EasyModel database using a traditional relational database schema, which currently comprises 10 tables interrelated by foreign keys (see Figure 3.12). To facilitate the connection between the Java application and the database, we utilized the popular JDBC driver library for Java.

Each Java entity class that needs to be stored in the database includes specific methods for various database operations: inserting new entities, updating existing ones and deleting entities. For entities like *Model*, which have several associated sub-entities, any database operation cascades down to the underlying classes, ensuring data consistency across related tables. The database is primarily divided into two main areas: model data and user data.

To enhance connectivity and optimize data transmission speed, a single database connection is established during the EasyModel startup process and remains active throughout the application's runtime. This connection is then properly closed during the shutdown process, ensuring efficient resource management.

During the application's design, security was a key consideration. User passwords are securely stored using a strong cryptographic hash algorithm, and the login dialog is specifically designed to protect user information from potential attacks. We have also implemented measures to prevent database query injection by utilizing the latest methodologies throughout the application, safeguarding it against common vulnerabilities that hackers may exploit for data collection. By prioritizing user privacy, EasyModel ensures that sensitive information and user data remain confidential and safeguarded from unauthorized access. This focus on data protection not only fosters a sense of trust and confidence among users but also encourages open sharing of models and collaborative research without concerns about data misuse.

3.8 SBML model exchange format

The Systems Biology Markup Language (SBML) (46, 47) is an open standard format for representing computational models in systems biology. Written in XML, it enables the exchange and storage of biochemical network models, such as metabolic networks and cell signaling pathways, across different software tools. SBML supports modular and hierarchical model structures, allowing complex systems to be broken down into manageable sub-models. It is versatile, accommodating various types of biological models, including deterministic and stochastic models. SBML also facilitates interoperability between software tools, includes support for annotations and metadata, and can be extended through custom packages to meet specific modeling needs. This makes SBML a crucial tool for sharing, simulating, and analyzing biological models in the systems biology community.

EasyModel currently utilizes the SBML Level 3 Version 2 specification for importing and exporting biochemical network models. This is accomplished using the JSBML (48) Java library which allows developers to read and write SBML model files from within the Java developing environment.

Models are imported by uploading an XML file in the *Model Select* view of the application, and exported by downloading the generated XML file from the *Simulation Results* view after performing a simulation.

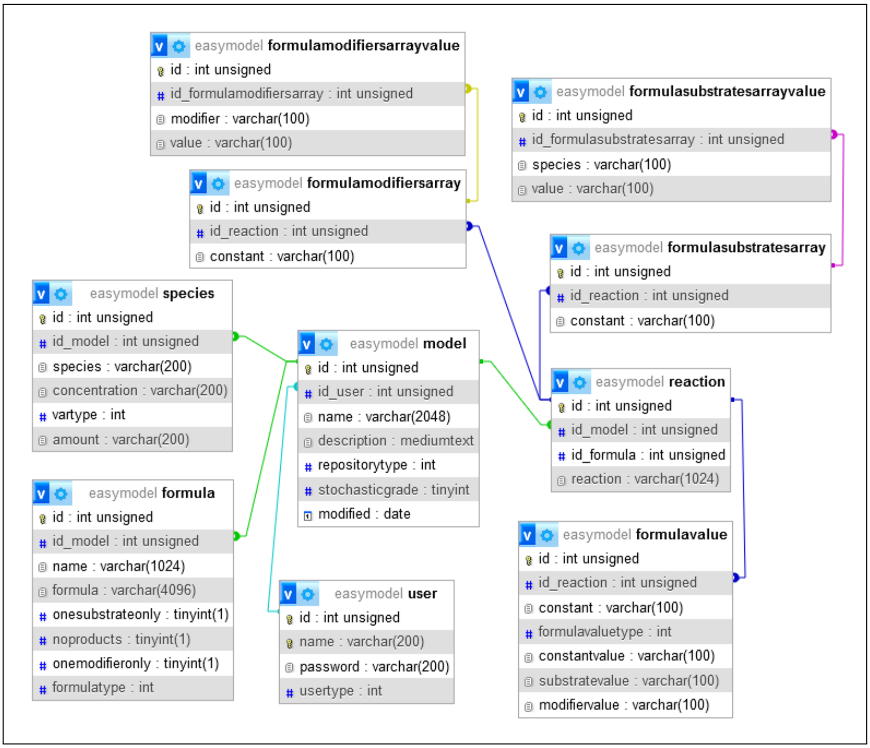


Figure 3.12: EasyModel's MySQL database table schema produced using *phpmyadmin*.

3.9 BioModels

To offer a range of well-known models to the EasyModel's user base, we decided to import models from the BioModels Database (65, 66, 67, 68).

BioModels is a repository and an open-source resource for storing, sharing, and retrieving computational models of biological processes. Hosted on the EBI (European Bioinformatics Institute) website, BioModels provides access to peer-reviewed, published models that describe a wide range of biological phenomena, such as biochemical networks, signaling pathways, and systems biology.

To achieve the import of the models, we designed a method to scrape all the manually-curated models in SBML/XML format from the BioModels website. This was easily achievable as all the models are named after the pattern

"BIOMDXXXXXXXXXX", e.g. "BIOMD0000000363". We designed a script that generated a download URL for each model number, starting from 1 to the total number of models. Then we downloaded the models using a download manager. Afterwards, we imported the downloaded XML models into EasyModel trough an automated batch process. This special process is activated through a private URL in the EasyModel web application. When the URL is requested, EasyModel goes through all the XML files located in a specific directory which contains all the downloaded BioModels files. For each XML file, EasyModel tries to import the model using the already-implemented SBML import method. Some BioModels cannot be imported into EasyModel as EasyModel doesn't support some of the features of the SBML specification. These models are skipped and not imported. To deem a model as valid, EasyModel performs a series of checks on the model which includes trying to perform a basic simulation on the Mathematica kernel. Once the model is deemed to be compatible, it is saved into the public repository of the EasyModel's models database. Thanks to this, EasyModel has an hefty dataset of models to start from.

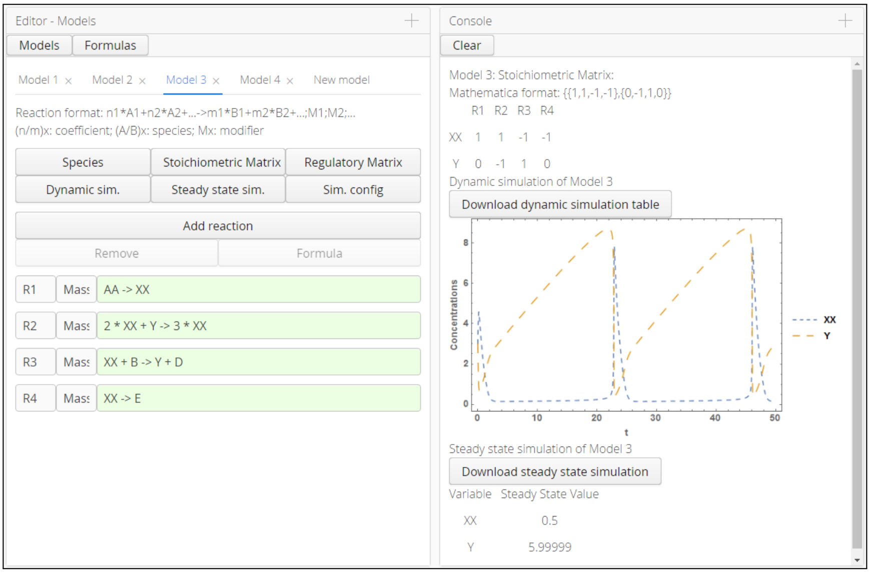
In the meantime, this process must be performed manually. We are considering developing an unattended process to periodically update the database with new available models.

3.10 EasyModel version history

The first version of EasyModel was the **EasyModel 1.0 alpha** (see Figure 3.13). This version was not released to the public, and its main objective was to tentatively assess the feasibility of creating a Java web application using the Vaadin web user interface framework in conjunction with Wolfram Mathematica to perform the underlying mathematical calculations. The version featured a simple user interface, as its focus was primarily on evaluating the technical capabilities of the programming environment.

The user interface consisted of a single view that included all functionality—modeling editor, formula editor, simulation configurator, and simulation results—within the same space. The set objectives were successfully achieved, proving that EasyModel had potential under the chosen technologies such as Java EE 7, Vaadin 7, and Mathematica 10. This success encouraged further development of the tool, leading to subsequent versions with more advanced features and a refined user experience.

The first public release of EasyModel was the **EasyModel 1.0** version (see Figure 3.14). It was centered around providing fundamental yet comprehensive functionality for a simulation software, inclusive of a user-friendly web-



Reaction format: n1*A1+n2*A2+...->m1*B1+m2*B2+...;M1;M2;...
(n/m)x: coefficient; (A/B)x: species; Mx: modifier

R1	Mass	AA -> XX
R2	Mass	2 * XX + Y -> 3 * XX
R3	Mass	XX + B -> Y + D
R4	Mass	XX -> E

Model 3: Stoichiometric Matrix:
Mathematica format: {{1,1,-1,-1},{0,-1,1,0}}

	R1	R2	R3	R4
XX	1	1	-1	-1
Y	0	-1	1	0

Dynamic simulation of Model 3

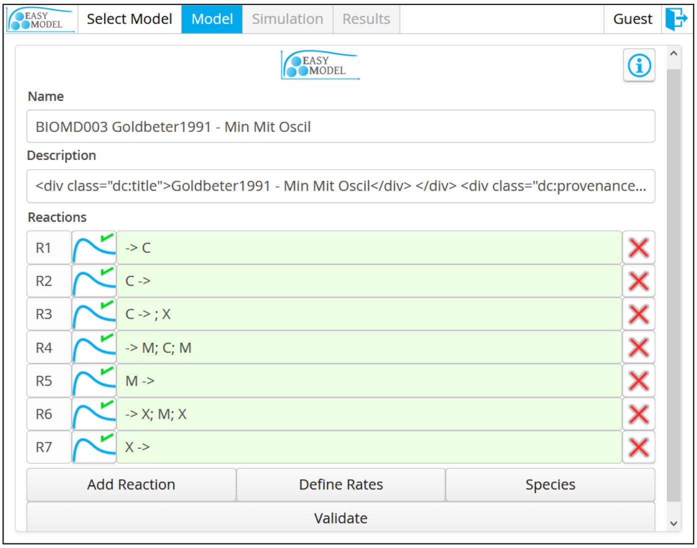
Steady state simulation of Model 3

Variable	Steady State Value
XX	0.5
Y	5.99999

Figure 3.13: EasyModel version 1.0 alpha web user interface.

based GUI, capable of utilizing a Mathematica calculus core. The software additionally featured a model editor, simulation configuration interface, and database for models and users, among other features such as compatibility with SBML (46) Level 3 Version 2 specifications, and Mathematica notebook generation. The focus was on systems of ordinary differential equations and deterministic algorithms, as detailed in (50). An article publication was made for this version which can be found under (69).

The next version **EasyModel 1.1** (70), introduced stochastic simulations and analysis of models in systems biology (see Figure 3.15). EasyModel 1.0 focused on the simulation of systems of ordinary differential equations using deterministic algorithms. Nevertheless, when the systems being modeled are composed of a small number of molecules, stochastic algorithms are more accurate (53), and linear noise analysis is a more appropriate tool than sensitivity analysis to understand the limitations and regulation of the system (56). While stochastic simulation and linear noise analysis are available in several simulation applications (53, 57), they lack the usability characteristics EasyModel provides to new systems biologists. Because of the importance of stochasticity in molecular systems biology, it was important that EasyModel also provided this functionality to its users.



Name

BIOMD003 Goldbeter1991 - Min Mit Oscil

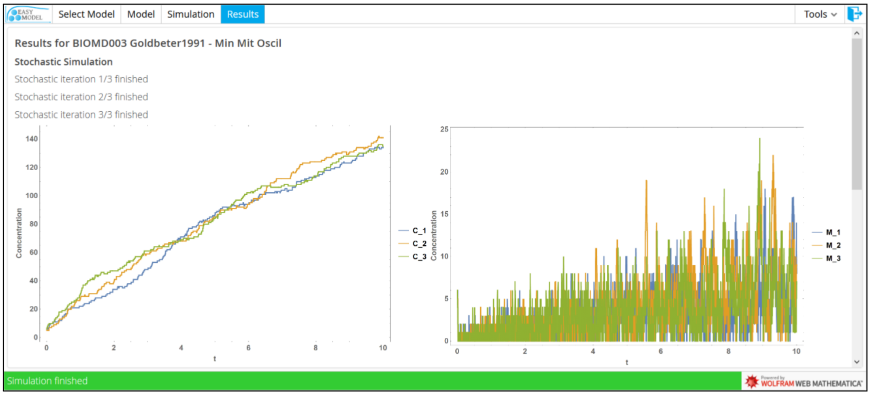
Description

<div class="dc:title">Goldbeter1991 - Min Mit Oscil</div> </div> <div class="dc:provenance...

Reactions

R1	-> C
R2	C ->
R3	C -> ; X
R4	-> M; C; M
R5	M ->
R6	-> X; M; X
R7	X ->

Figure 3.14: EasyModel version 1.0 web user interface.



Results for BIOMD003 Goldbeter1991 - Min Mit Oscil

Stochastic Simulation

Stochastic iteration 1/3 finished

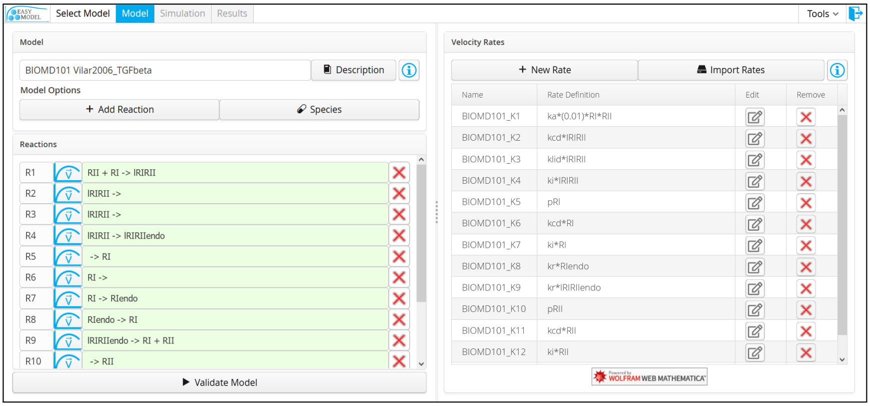
Stochastic iteration 2/3 finished

Stochastic iteration 3/3 finished

Simulation finished

Figure 3.15: EasyModel version 1.1 web user interface.

In the development of **EasyModel 2.0**, we introduced a suite of exciting new features designed for systems biologists and scientists (see Figure 3.16). These features included a new stochastic simulation method known as τ-leaping and parameter scanning for the deterministic regime. The τ-leaping method performs stochastic simulations more efficiently than the traditional SSA method by optimizing execution time, though it produces approximated results compared to SSA. This approximation is typically acceptable, making τ-leaping a valuable addition.



R1	RII + RI -> lRIRII
R2	lRIRII ->
R3	lRIRII ->
R4	lRIRII -> lRIRIIendo
R5	-> RI
R6	RI ->
R7	RI -> RIendo
R8	RIendo -> RI
R9	lRIRIIendo -> RI + RII
R10	-> RII

Name	Rate Definition	Edit	Remove
BIOMD101_K1	ka*(0.01)*RI*RII		
BIOMD101_K2	kcd*lRIRII		
BIOMD101_K3	klid*lRIRII		
BIOMD101_K4	ki*lRIRII		
BIOMD101_K5	pRI		
BIOMD101_K6	kcd*RI		
BIOMD101_K7	ki*RI		
BIOMD101_K8	kr*RIendo		
BIOMD101_K9	kr*lRIRIIendo		
BIOMD101_K10	pRII		
BIOMD101_K11	kcd*RII		
BIOMD101_K12	ki*RII		

Figure 3.16: EasyModel version 2.0 web user interface.

Additionally, we incorporated parameter scanning, which enables a global sensitivity analysis. This analysis is more accurate than the local sensitivity analysis previously available, providing more detailed insights at the cost of extended execution time.

For the latest version, **EasyModel 2.4**, our primary focus was on enhancing the user experience (see Figure 1.1). We introduced a simulation job queue system to address the inconvenience of manually relaunching simulations when the Mathematica kernel was occupied by another user. Implementing this feature was complex, requiring a deep understanding of Java threads and Vaadin's asynchronous user interface updates. Nonetheless, we believe the effort was well worth it.

Another significant change was the upgrade from Vaadin 8 to Vaadin 24. This upgrade was far from straightforward, as it necessitated a complete rewrite of the user interface to accommodate the new syntax system. Although time-consuming, this update was crucial for the project's long-term viability, ensuring it remains future-proof by staying aligned with the latest Vaadin version. Additionally, the new interface is more modern and sleek, while still adhering to the user-friendly philosophy that defines the EasyModel project.

Alongside these major updates, we also made general improvements and fixed minor bugs to further enhance the overall performance and stability of the application.

4 Results

Continuous effort, not strength or intelligence, is the key to unlocking our potential.

(Winston Churchill)

This chapter will showcase the results obtained from the development of the EasyModel web application.

4.1 EasyModel usage

This section provides a user manual for potential EasyModel users.

New users are advised to read the tutorial via the welcome dialog that appears when first visiting the web application.

The application can be used either as a guest or as a logged-in user.

4.1.1 Usage design

The guiding principle in the design of EasyModel's user interface was that it should be easy to use by both novel and expert users. In consequence, we minimized the number of screen changes required to use the application as much as possible.

The interface guides the user through a series of conceptual steps in the modeling and simulation process. The initial screen prompts the user to select the model to work on or to create a new model from scratch. The next screen deals with model implementation. This entails defining model name, model description, reaction list, formula list, formula binding, etc. The third screen deals with model simulation and analysis settings and the fourth and final screen presents the simulation results. While it would be possible to put all functionality in one screen, we believe this approach would make EasyModel less user-friendly, because it would contain too much information and options to select from simultaneously.

Hence, the application has been meticulously crafted for utilization in four consecutive steps (elaborated individually in the following sections):

1. Model Select.
2. Model Builder.
3. Simulation Launcher.
4. Simulation Results.

4.1.2 Tutorial

A short tutorial is provided to new users in order to facilitate usage. When the user lands on the website for the first time, a dialog greets him/her (see Figure 4.1) to advise to read the tutorial first before using the application. The tutorial opens in a new browser tab (see Figure 4.2), allowing users to read it while simultaneously using the application. The tutorials consist in a series of slides depicting screenshots and text explanations about how to navigate through the application. The Tutorial can be consulted anytime through the *Tools* menu located in the top menu bar. Furthermore, EasyModel has *Information* buttons all across the application to keep the user informed on what to do next.

4.1.3 User login

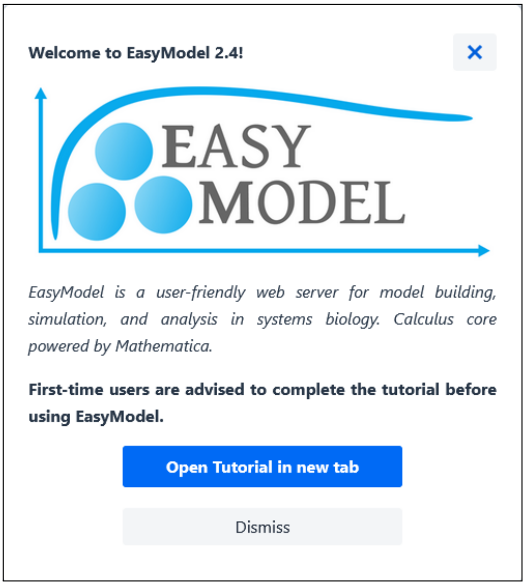
Users can access the application without the need to log in or register.

Non-registered users can fully utilize the application without worrying about causing any permanent change. Their actions will not affect the database or the application itself.

Registered users, on the other hand, can create, store, and share models within the EasyModel community. This feature allows users to contribute to the growth of the tool with their models. Registered users can access their models through the private repository available in the *Model Select* screen 4.1.4.

To log in with a username and password, use the *Login* option from the *User* menu located in the top menu bar (see Figure 4.3).

To register a new account, use the *Register* option from the *User* menu located in the top menu bar. Currently, to facilitate the process, creating a new account only requires entering a username and a password (see Figure 4.3).


Welcome to EasyModel 2.4!

EasyModel is a user-friendly web server for model building, simulation, and analysis in systems biology. Calculus core powered by Mathematica.

First-time users are advised to complete the tutorial before using EasyModel.

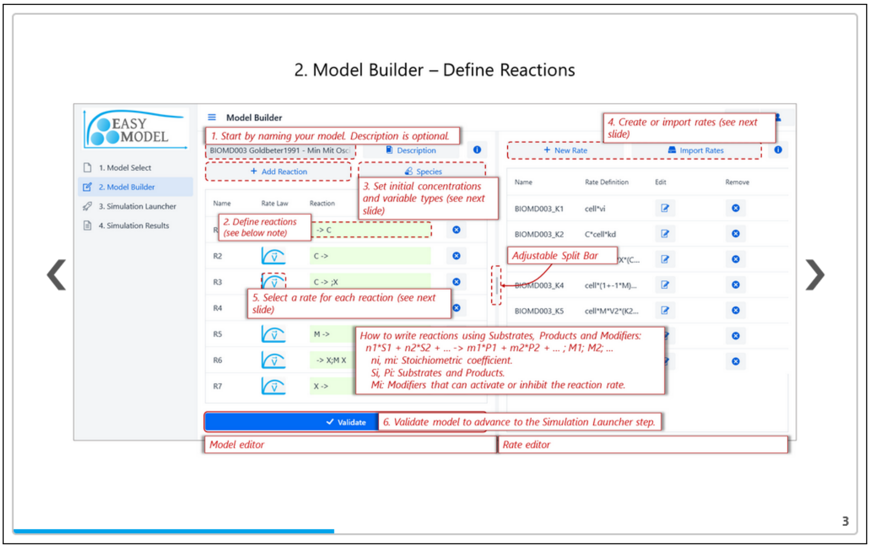
Figure 4.1: EasyModel: First visit greeting dialog.

4.1.4 Model select

This is the first step in the modeling process. User begins by selecting one of the three possible model sources as described in the following list (see Figure 4.4).

- Database model: A model can be chosen from the database. There are 2 repositories: the public and the private. The public repository houses a collection of pre-built models, sourced from various places, including the renowned BioModels Database (65, 68), as well as models developed and shared by other users on the platform community. These readily available models offer a convenient starting point for users. On the other hand, logged users can access their private repository. This offers users the flexibility to store their own created models or continue working on existing models that are still in progress. This private space allows users to experiment and iterate freely, without affecting the shared public models. The core concept of the private repository is for users to prepare models for publication, enabling other users to utilize them as well.

- Create a new model: Alternatively, you have the option to create a brand-new model. This involves crafting a model from scratch, tailored


2. Model Builder – Define Reactions

1. Start by naming your model. Description is optional.

2. Define reactions (see below note)

3. Set initial concentrations and variable types (see next slide)

4. Create or import rates (see next slide)

5. Select a rate for each reaction (see next slide)

How to write reactions using Substrates, Products and Modifiers:
n1*S1 + n2*S2 + ... -> m1*P1 + m2*P2 + ... ; M1; M2; ...
ni, mi: Stoichiometric coefficient.
Si, Pi: Substrates and Products.
Mi: Modifiers that can activate or inhibit the reaction rate.

6. Validate model to advance to the Simulation Launcher step.

Adjustable Split Bar

Model editor

Rate editor

Name	Rate Definition
BIOMD003_K1	cell*vi
BIOMD003_K2	C*cell*kd
BIOMD003_K4	cell*(1+-1*M)...
BIOMD003_K5	cell*M*V2*(K2...

Name	Reaction
R1	-> C
R2	C ->
R3	C -> ;X
R4	
R5	M ->
R6	-> X;M X
R7	X ->

Figure 4.2: EasyModel: Tutorial presented in slides.

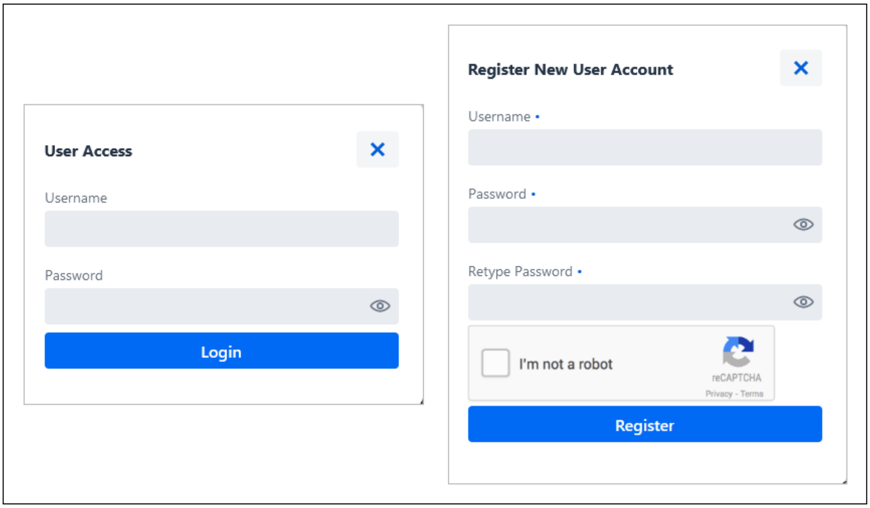


Figure 4.3: EasyModel: Login and register dialogs.

to your specific needs and objectives.

- Import from a local SBML file: You can import a model from a locally stored SBML (46) file. SBML files contain a standardized representation of biological models, making it an efficient method for transferring models between different platforms or software.

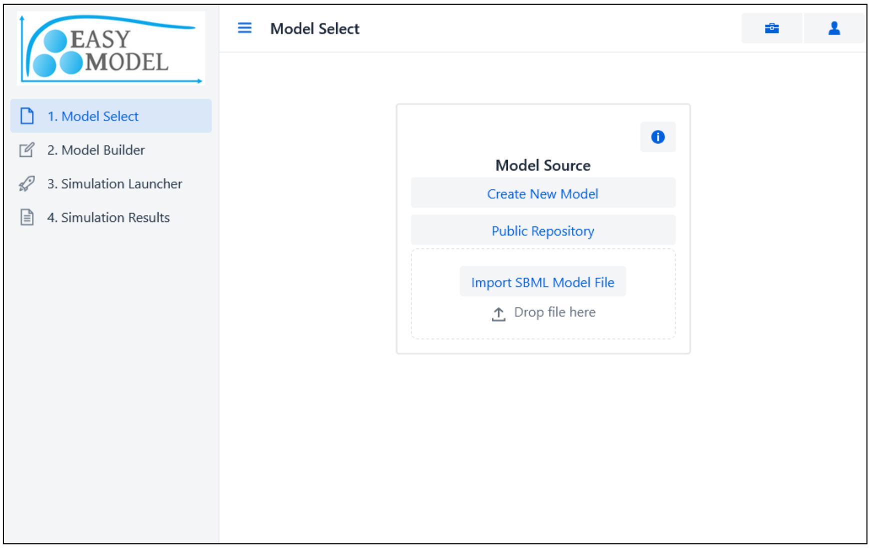
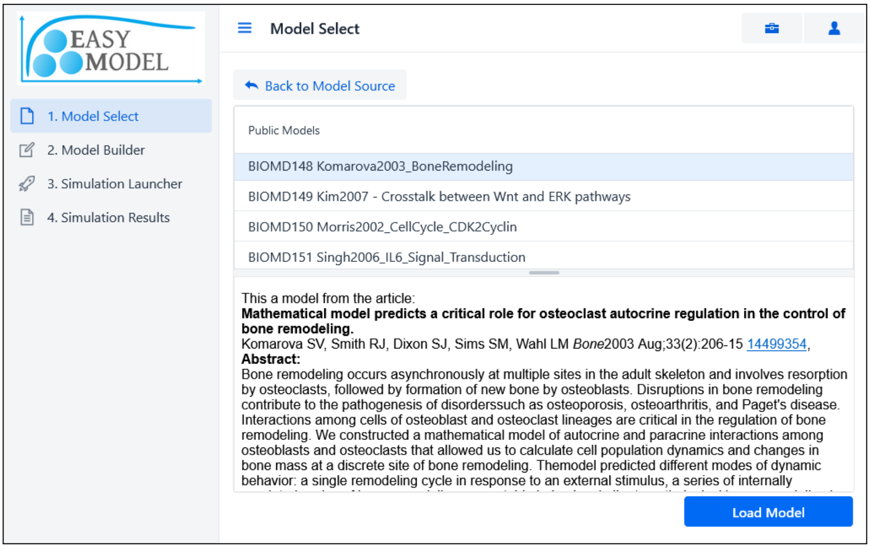


Figure 4.4: EasyModel: Model source. Model can be selected from the database, created from scratch, or imported from a local SBML file.

Afterwards, if the user has selected a database repository as model source, he/she is prompted to select a model from the repository list (see Figure 4.5).

Guest users, although unable to save their created models in the system, can still harness the modeling capabilities to generate new models. These models can be downloaded in the SBML format, providing a way for guest users to preserve their work externally. Subsequently, they have the option to re-upload their models into EasyModel at a later time by importing the SBML file, enabling continued development without the need of registering a user account.



BIOMD148 Komarova2003_BoneRemodeling
BIOMD149 Kim2007 - Crosstalk between Wnt and ERK pathways
BIOMD150 Morris2002_CellCycle_CDK2Cyclin
BIOMD151 Singh2006_IL6_Signal_Transduction

This a model from the article:
Mathematical model predicts a critical role for osteoclast autocrine regulation in the control of bone remodeling.
Komarova SV, Smith RJ, Dixon SJ, Sims SM, Wahl LM *Bone*2003 Aug;33(2):206-15 14499354,
Abstract:
Bone remodeling occurs asynchronously at multiple sites in the adult skeleton and involves resorption by osteoclasts, followed by formation of new bone by osteoblasts. Disruptions in bone remodeling contribute to the pathogenesis of disorderssuch as osteoporosis, osteoarthritis, and Paget's disease. Interactions among cells of osteoblast and osteoclast lineages are critical in the regulation of bone remodeling. We constructed a mathematical model of autocrine and paracrine interactions among osteoblasts and osteoclasts that allowed us to calculate cell population dynamics and changes in bone mass at a discrete site of bone remodeling. Themodel predicted different modes of dynamic behavior: a single remodeling cycle in response to an external stimulus, a series of internally

Figure 4.5: EasyModel: Model select list from the public database repository.

4.1.5 Model builder

In this step, the user defines the model to be simulated or analyzed. The user interface is divided in two panels by a vertical splitter (see Figure 4.6). The left side focus on reaction definition and global settings. The right side focuses on formula definition. Formulas are also known as rate laws or velocity rates.

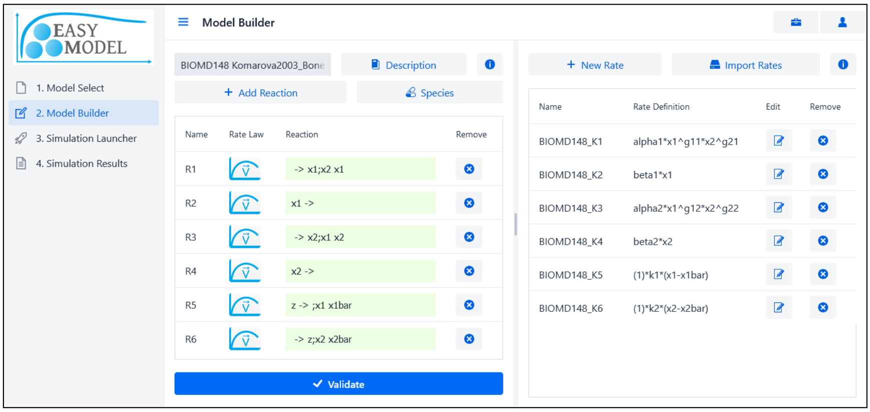
First step should be to define the model name and description on the left panel of the model builder.

4.1.5.1 Defining reactions

Next, reactions should be defined line by line using the following syntax.

```
Reaction definition: Substrates -> Products ; Modifiers
How to write: n1*A1 + n2*A2 + ... -> m1*B1 + m2*B2 + ... ; M1
    M2 ...
Legend: nX,mX: coefficient; AX,BX: species; MX: modifier
Example: 3*x1 + 4*x2 -> x3 + 4*x4 ; x5 : x6
```

To facilitate writing, users can press the enter key at the end of a reaction to continue writing a new reaction like he/she would in a text editor. When typing a reaction, the system continuously checks if the reaction is valid and indicates



BIOMD148 Komarova2003_Bone

Name	Rate Law	Reaction	Remove
R1		-> x1;x2 x1	
R2		x1 ->	
R3		-> x2;x1 x2	
R4		x2 ->	
R5		z -> ;x1 x1bar	
R6		-> z;x2 x2bar	

Name	Rate Definition	Edit	Remove
BIOMD148_K1	alpha1*x1^g11*x2^g21		
BIOMD148_K2	beta1*x1		
BIOMD148_K3	alpha2*x1^g12*x2^g22		
BIOMD148_K4	beta2*x2		
BIOMD148_K5	(1)*k1*(x1-x1bar)		
BIOMD148_K6	(1)*k2*(x2-x2bar)		

Figure 4.6: EasyModel: Model builder interface.

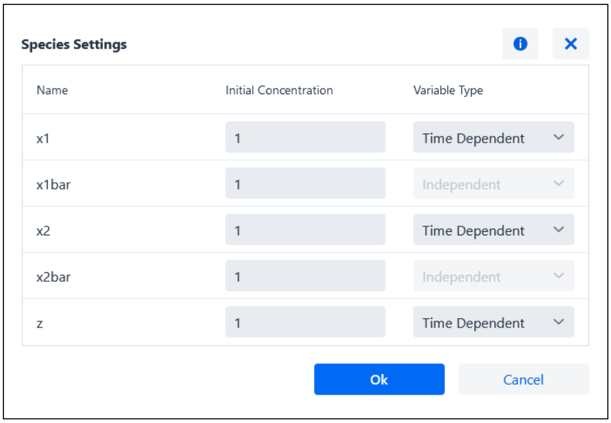
this by changing the background color style of the reaction text field. Green shade indicates a valid reaction while red shade indicates an invalid reaction.

After writing a reaction, the system recognizes the variable names as species and integrates them into the system. After defining the reactions, species settings can be changed by clicking the *Species* button. In the *Species Settings* dialog (see 4.7), user must define the initial concentrations and the variable type for the species of the selected model. Concentrations are real numerical values. Variable type can be either *Time dependent* (concentration values evolve over time) or *Constant* (concentration values remain constant over time). In case these initial concentrations are not explicitly provided, EasyModel automatically sets them to a default value of 1, ensuring that the model is initialized correctly.

4.1.5.2 Defining rate laws

For accurate representation, every reaction within the system must be associated with a rate law which governs the rate at which the reaction will occur during the simulation. Before assigning rate laws, these have to previously been defined in the editor.

EasyModel offers users a variety of rate law options with predefined formalisms, including Power Law, Mass Action, Saturating, and Cooperative (49). These formalisms provide a convenient way to model a wide range of biochemical and regulatory processes commonly encountered in molecular biology. Additionally, EasyModel stores all the previously defined rate laws in a generic

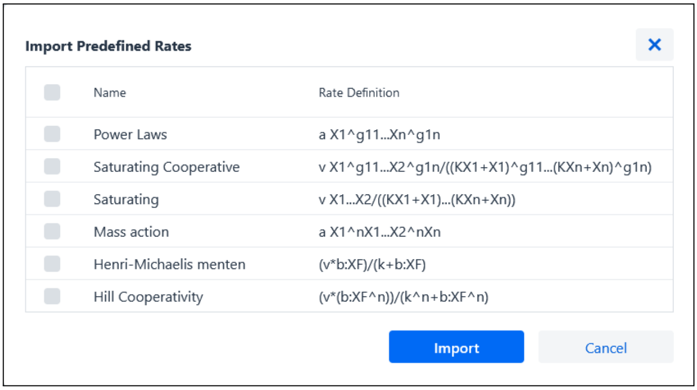


Species Settings

Name	Initial Concentration	Variable Type
x1	1	Time Dependent
x1bar	1	Independent
x2	1	Time Dependent
x2bar	1	Independent
z	1	Time Dependent

Figure 4.7: EasyModel: Model species settings.

format, creating a rate law database for future reference. There rate laws are available in the *Import Rates* button from the rate laws editor (see Figure 4.8).



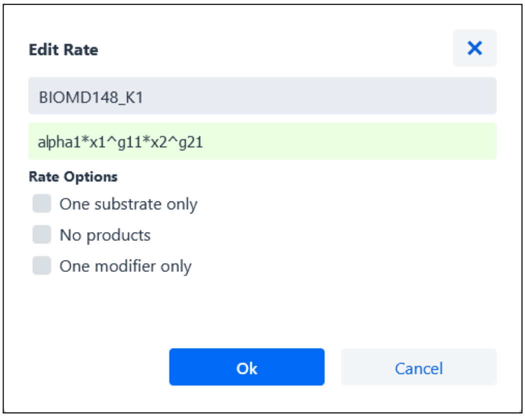
Import Predefined Rates

Name	Rate Definition
Power Laws	a X1^g11...Xn^g1n
Saturating Cooperative	v X1^g11...X2^g1n/((KX1+X1)^g11...(KXn+Xn)^g1n)
Saturating	v X1...X2/((KX1+X1)...(KXn+Xn))
Mass action	a X1^nX1...X2^nXn
Henri-Michaelis menten	(v*b:XF)/(k+b:XF)
Hill Cooperativity	(v*(b:XF^n))/(k^n+b:XF^n)

Figure 4.8: EasyModel: Import of predefined rate laws.

Users have also the flexibility to create custom-made rate law formulas, enabling them to capture specialized or unique kinetic behaviors specific to their research (see Figure 4.9). To define new formulas users must adhere to the following guidelines.

```
How to define rate expressions
  Usable operators: +-/*^()
```


Edit Rate

BIOMD148_K1

alpha1*x1^g11*x2^g21

Rate Options

- [ ] One substrate only
- [ ] No products
- [ ] One modifier only

Figure 4.9: EasyModel: Defining a new rate law.

```
Reserved symbols:
  Mathematica functions/constants: m:<Mathematica function>
  Mathematica function indexes: i:<index>
  Special variables:
    b:t -> Time.
    b:X[] -> Mathematica substrate list.
    b:A[] -> Mathematica substrate coefficient list.
    b:M[] -> Mathematica modifier list.
    b:XF -> First substrate.
    b:MF -> First modifier.
Example: m:Product[b:X[[i:j]]^g[[i:j]],{i:j,1,m:Length[b:X]}]
```

Rate law definition also allow to set up reaction assigning restrictions. This means the formula will not be able to be bound to a reaction that doesn't comply with the rate law restrictions. Available restrictions are as follows.

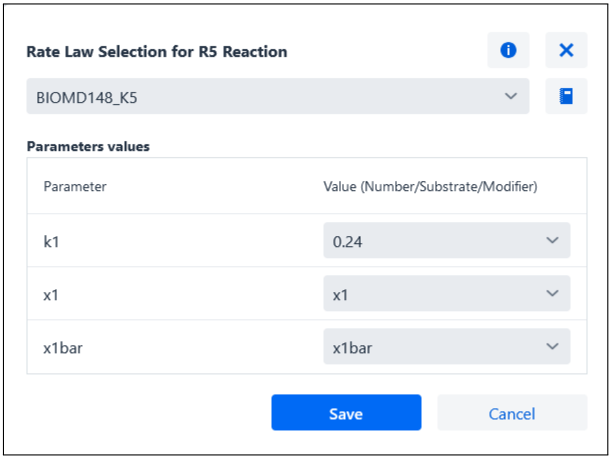
- One substrate only. Reaction must have only one substrate, and nothing more (e.g. $X1 \to$).
- No products. Reaction must not have any product (e.g. $X1 + X2 \to$)
- One modifier only. Reaction must have only one modifier, and nothing more (e.g. $\to; M1$).

4.1.5.3 Assigning rate laws

Once reactions and rate laws are defined, rate laws must be assigned to reactions. Each reaction must have one rate law assigned to it which describes the reaction speed. Rate laws contain parameters and these must be fulfilled during the reaction assignment. These parameters must be filled with one of the three sources described below.

- A numerical value: A real number.
- A substrate: One of the substrates of the selected reaction.
- A modifier: One of the modifiers of the selected reaction. Can also be any species within the model that will act as a modifier of the selected reaction.

The rate law assignment is done within the rate law selection dialog after clicking the *Rate Law selection* button found in the reaction rows (see Figure 4.10). Within this dialog, the user must select a compatible rate law and fill the parameter values with either numerical or text values. Text values must refer to species names within the model.


Rate Law Selection for R5 Reaction

BIOMD148_K5

Parameters values

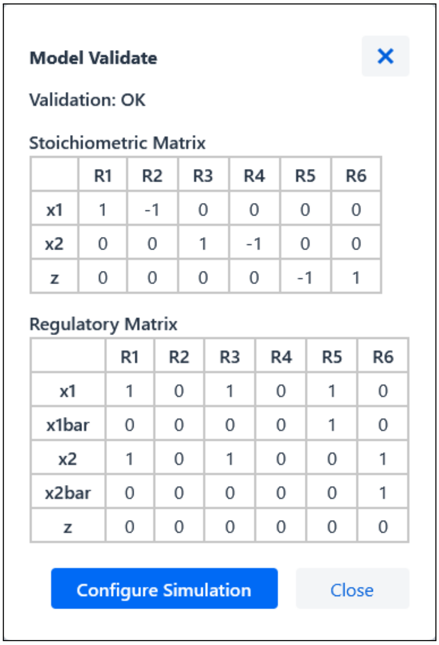
Parameter	Value (Number/Substrate/Modifier)
k1	0.24
x1	x1
x1bar	x1bar

Figure 4.10: EasyModel: Linking a rate law to a reaction with definition of parameter values.

Some mathematical formalisms require to fulfill a special type of parameters named *Parameters dependent on substrates/modifiers*. This parameters require a numerical value for each substrate or modifier present in the selected reaction. For example, in the Power Law formalism, the g parameter represents exponential values dependent on substrates of the selected reaction. Same applies for the Km parameter in the Michaelis-Menten rate law. Formally, these parameters are defined as array type variables that require a numerical value for every substrate or modifier found in the selected reaction.

4.1.6 Validating the model

After following the previous steps, the model should be able to get verified. The *Validate* button from the *Model Builder* will display a dialog indicating the validation status (see Figure 4.11). If the model contains errors, a list of errors will be displayed. If the model is valid, EasyModel will display both stoichiometric and regulatory matrices and suggest to proceed to simulation configuration.



Model Validate

Validation: OK

Stoichiometric Matrix

	R1	R2	R3	R4	R5	R6
x1	1	-1	0	0	0	0
x2	0	0	1	-1	0	0
z	0	0	0	0	-1	1

Regulatory Matrix

	R1	R2	R3	R4	R5	R6
x1	1	0	1	0	1	0
x1bar	0	0	0	0	1	0
x2	1	0	1	0	0	1
x2bar	0	0	0	0	0	1
z	0	0	0	0	0	0

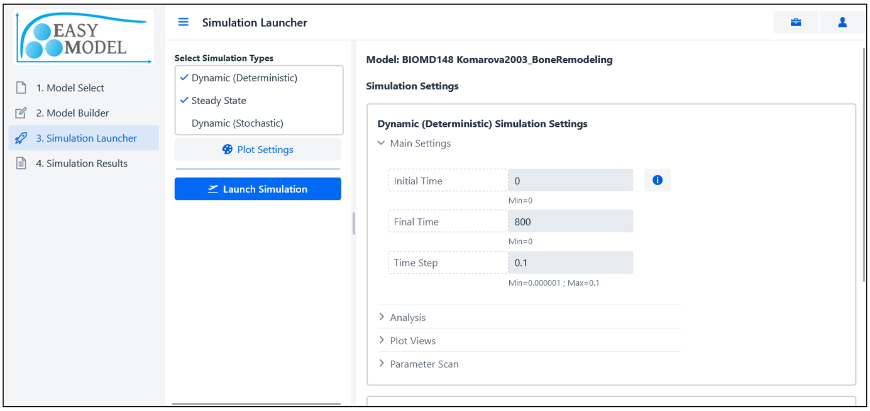
Figure 4.11: EasyModel: Model validation. Stoichiometric and regulatory matrices are displayed.

A stoichiometric matrix is a mathematical representation used in systems biology, biochemistry, and chemical engineering to describe the relationships between substrates and products in a set of chemical reactions. It is a fundamental concept in metabolic network analysis and other fields where chemical reactions are modeled.

A regulatory matrix is a conceptual tool used in systems biology and gene regulatory network analysis to represent the relationships between regulatory elements, such as genes, proteins, or other molecules, and their targets. It helps in understanding how different modifiers influence the expression of genes or the activity of proteins within a biological system.

4.1.7 Simulation launcher

The next step is to configure the simulation before launching it into the job queue system for execution. Once the model has been successfully validated, users are given the opportunity to choose the specific simulations and analyses to be performed (see Figure 4.12). For a deeper understanding of the simulations, please refer to section 3.4.



Select Simulation Types
✓ Dynamic (Deterministic)
✓ Steady State
Dynamic (Stochastic)

Model: BIOMD148 Komarova2003_BoneRemodeling

Simulation Settings

Dynamic (Deterministic) Simulation Settings

Main Settings

Initial Time	0	Min=0
Final Time	800	Min=0
Time Step	0.1	Min=0.000001 ; Max=0.1

Figure 4.12: EasyModel: Simulation configuration view.

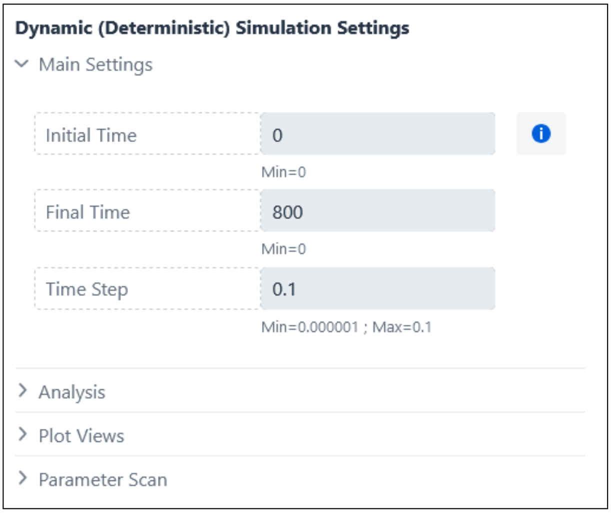
Simulations can be performed under two types of regimes: deterministic and stochastic. If the system under study comprises a large number of molecules, users can opt for a deterministic simulation, which provides a more efficient and computationally feasible approach for such cases. Otherwise, the stochastic simulation can be chosen as it will provide more accurate results.

4.1.7.1 Deterministic regime

For deterministic simulations, users can conduct dynamic time course simulations, which track the system's behavior over time, and steady state simulations, which focus on identifying stable equilibrium points.

For **dynamic** simulations, users can configure the following *Main Settings* (see Figure 4.13):

- Initial time: A non-negative real numerical value that defines the starting point of the simulation. Time units must be the same as used in velocity functions.
- Final time: A positive real numerical value that defines the end point of the simulation. Value must be greater than the initial time value.
- Time step: A positive real numerical value that defines the level of detail in the simulation's sampling. Smaller values produce more detailed plots but increase computational cost.



Dynamic (Deterministic) Simulation Settings

Main Settings

Initial Time	0
	Min=0
Final Time	800
	Min=0
Time Step	0.1
	Min=0.000001 ; Max=0.1

Figure 4.13: EasyModel: Dynamic (deterministic) simulation settings.

Analysis available for dynamic simulations include:

- Gains analysis: Provides detailed insights into how changes in control variables (or constants) affect the model's behavior. This analysis is local and doesn't make any actual change in the values of the control variables.
- Sensitivities analysis: Provides detailed insights into how changes in rate parameters affect the model's behavior. This analysis is local and doesn't make any actual change in the values of the rate parameters.

Another feature of dynamic simulations is **Plot Views**. This allows users to filter which variables are plotted during the simulation by creating multiple views, each with different assigned variables. All views will be generated during the same simulation process. This feature is particularly useful for clarifying situations where several variables overlap, providing users with a clearer understanding of the simulation results.

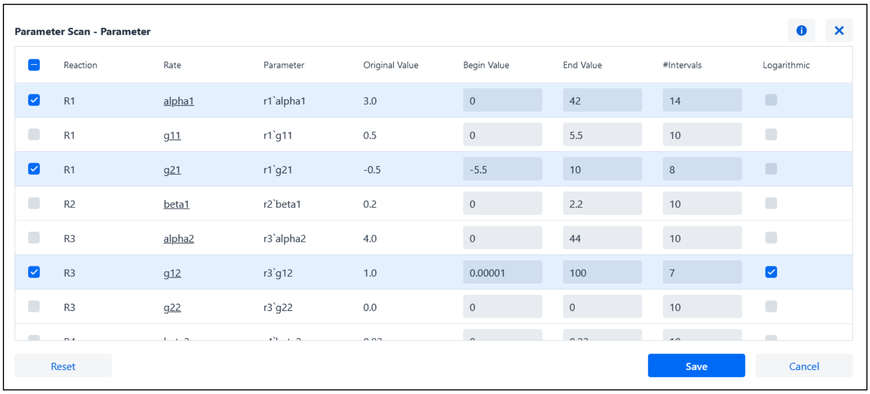
Parameter scan is another available feature (see Figure 4.14). Parameter scanning can be performed on both rate parameters and control variables. Parameter scan allows to conduct a global sensitivity analysis. Unlike standard sensitivity analysis, parameter scanning does actually change the values of the parameters to be studied. This provides a more accurate study at the cost of longer execution time. EasyModel allows to define numerical intervals to be analyzed for different rate parameters. These are the settings that can be configured:

- Parameters: Users can select multiple parameters to be analyzed during the same simulation process.
- Begin value: A real numerical value that defines the initial value of the parameter to be analyzed.
- End value: A real number that defines the final parameter value of the analysis. Value must be greater than the begin value.
- Number of intervals: A natural number specifying how many divisions to create between the *begin value* and the *end value*. By default, these divisions are generated on a linear scale. The resulting values will be used in the parameter scan.
- Logarithmic toggle: Indicates whether to use a logarithmic scale for calculating the interval points.

EasyModel will run a separate simulation for each interval point, allowing users to observe how changes in the parameter value affect the system's behavior. The results of the parameter scan are organized by two factors: dependent variables and the selected parameters being scanned. This separation helps users more easily identify how changes in each parameter affect the model's behavior.

For **steady state** simulations, the main settings are (see Figure 4.15):

- Threshold: A numerical value used to determine whether a result qualifies as a valid steady state. This value is typically a small number close to zero. Mathematica uses the built-in function `FindRoot[]` to search for

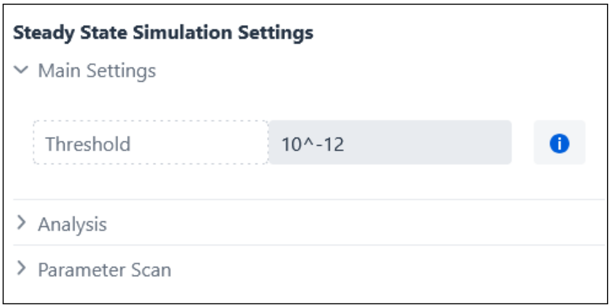


Parameter Scan - Parameter

	Reaction	Rate	Parameter	Original Value	Begin Value	End Value	#Intervals	Logarithmic
✓	R1	alpha1	r1`alpha1	3.0	0	42	14	
	R1	g11	r1`g11	0.5	0	5.5	10	
✓	R1	g21	r1`g21	-0.5	-5.5	10	8	
	R2	beta1	r2`beta1	0.2	0	2.2	10	
	R3	alpha2	r3`alpha2	4.0	0	44	10	
✓	R3	g12	r3`g12	1.0	0.00001	100	7	✓
	R3	g22	r3`g22	0.0	0	0	10	

Figure 4.14: EasyModel: Parameter scan configuration. Multiple parameter can be selected to be performed during the same simulation.

steady states. While we generally aim to find steady states where rate laws equal zero, the function may return approximate solutions that are close to zero. To account for these results and avoid discarding them, the user can set a threshold value.



Steady State Simulation Settings

Main Settings

Threshold	10^-12

Figure 4.15: EasyModel: Steady state simulation settings.

EasyModel allows to perform analyses for steady state simulations: linear stability analysis, gains/sensitivities analysis and parameter scan. Linear stability analysis enables users to assess the stability of equilibrium points and understand the system's response to perturbations.

4.1.7.2 Stochastic regime

The **stochastic** simulation is particularly valuable when dealing with systems involving low molecule counts and intrinsic randomness, characteristics commonly encountered in biological systems. By incorporating stochastic simulations, EasyModel enables users to capture the inherent uncertainty and fluctuations that play a critical role in biological processes.

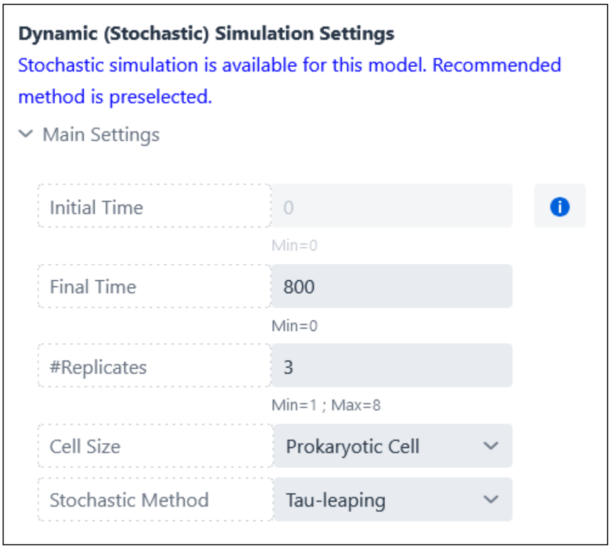
The users can configure the following stochastic simulation settings (see Figure 4.16):

- Initial time: A non-negative real number for defining beginning time of simulation. Due to the nature of stochastic simulations, this value is initialized to zero and can't be changed afterward.
- Final time: A non-negative real number for defining end time of simulation. Value must be higher than initial time.
- Replicates: A natural number that defines the number of times the simulation will be repeated. Performing several equal stochastic simulations allows to assess the system's behavior with statistical significance, capturing the inherent randomness in biological processes. In this case, the program sets the default number of repeated simulations to 3 to ensure a reliable statistical representation.
- Cell size: It allows for changing between concentrations and number of molecules. In cases where users do not have a specific cell size preference, EasyModel considers a default cell size, which emulates that of a *Prokaryotic cell*.
- Stochastic method: This option allows you to choose between the standard SSA method and the optimized τ-leaping method. The SSA method provides more accurate results but requires more computational time, whereas the τ-leaping method reduces processing time at the expense of some loss in accuracy.

4.1.7.3 Plot settings

Use plot settings to configure how the plots will be generated (see Figure 4.17). The available settings are:

- Line thickness: A non-negative real number for defining the thickness of the graphs generated during simulation.



Dynamic (Stochastic) Simulation Settings

Stochastic simulation is available for this model. Recommended method is preselected.

Main Settings

Initial Time	0	Min=0
Final Time	800	Min=0
#Replicates	3	Min=1 ; Max=8
Cell Size	Prokaryotic Cell	
Stochastic Method	Tau-leaping	

Figure 4.16: EasyModel: Dynamic (stochastic) simulation settings.

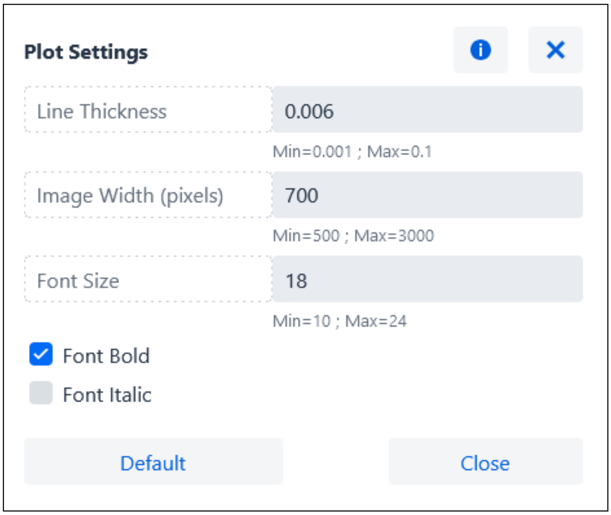
- Image width: A natural number that defines the resolution of the images by defining their width in pixels. Aspect ratio of images is maintained.
- Font size: A natural number that regulates the text size found in plots.
- Font bold toggle: Allows to show text found in plots in bold style.
- Font italic toggle: Allows to show text found in plots in italic style.

Once all actions are configured, the user presses the "Launch Simulation" button to submit the simulation job into the simulation job queue system (see Section 3.6) which will process it as soon as conditions are favorable.

4.1.8 Simulation results

4.1.8.1 Individual simulation results

After launching a simulation, the user is redirected to the individual simulation results view of the launched simulation job (see Figure 4.18). In this view, the user must wait for the simulation job to reach the execution status. The position in the simulation job queue is displayed in the top status bar. Once the simulation starts, its status will change from *In queue* to *Running*.


Plot Settings

Line Thickness	0.006
	Min=0.001 ; Max=0.1
Image Width (pixels)	700
	Min=500 ; Max=3000
Font Size	18
	Min=10 ; Max=24

[✓] Font Bold

[ ] Font Italic

Figure 4.17: EasyModel: Plot settings. Settings will be applied to the simulation plots.

As the simulation progresses, results are added to the results view in real-time. The platform ensures a seamless experience by providing individual results as soon as they are computed. This approach keeps the user continuously informed about the simulation's status as it advances toward completion.

The simulation outcomes are presented on the web page in multiple formats:

- Graphical representations.
- Textual form.
- Tabular data.
- Downloadable files.

For user convenience, EasyModel offers the flexibility to cancel simulations at any point during the process. By simply clicking a button, the system halts the simulation after the completion of the current Mathematica command being evaluated. This feature empowers users to efficiently manage and control their simulations, saving time and computational resources.

Moreover, in addition to graphical representations, users have the option to download the generated Mathematica notebook and the model in SBML format (see Figure 4.19).

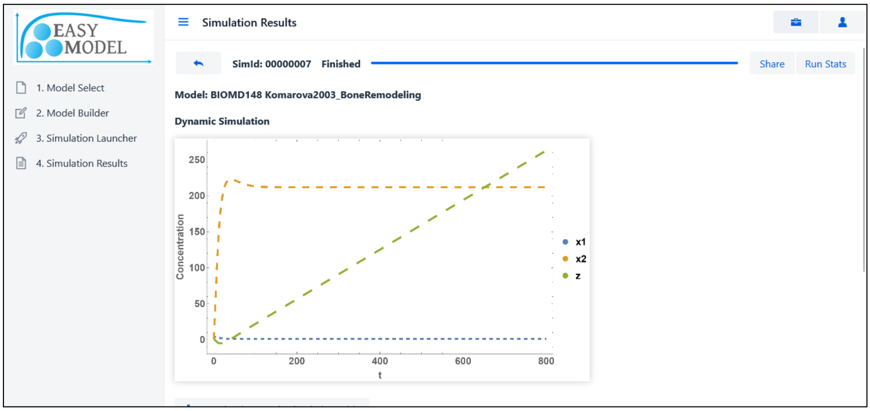


Figure 4.18: EasyModel: Simulation results view. Simulation results are updated in real-time. Users can monitor the execution status and share the results directly from this interface.

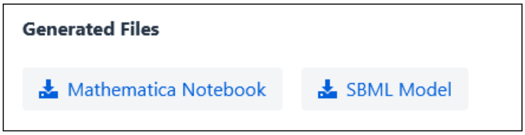


Figure 4.19: EasyModel: Generated files available for download.

A Mathematica notebook is a file type designed for use within the graphical interface of Wolfram Mathematica, allowing users to perform computations, visualize data, and write dynamic content. The downloadable Mathematica notebook includes meticulously crafted code that defines the entire model, along with the selected simulations and analyses. Users can examine and modify this code to run custom simulations and analyses. This feature empowers users by offering flexibility and control over the model's behavior.

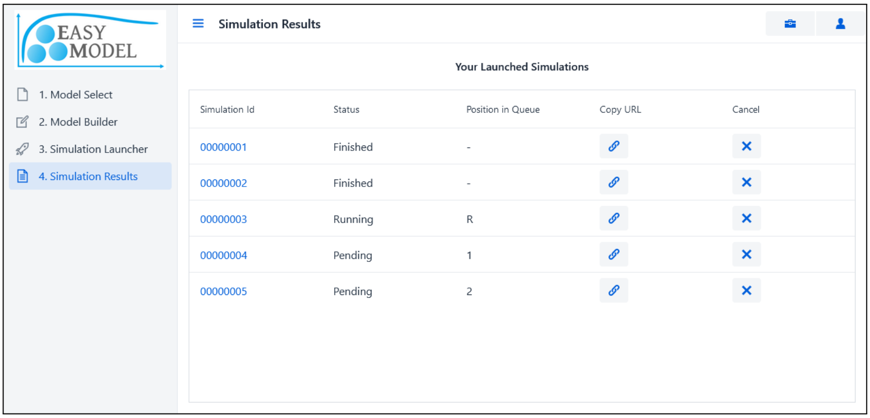
The SBML downloadable files contain the full definition of the model, including all its parameters, variables, and interactions. These files can be re-uploaded to EasyModel at a later time, allowing users to resume their work without needing to redefine the model. Additionally, the SBML format is a widely recognized standard in systems biology, which means that these files can also be imported into other compatible tools, such as simulation platforms or bioinformatics software. This promotes seamless collaboration across different research groups and ensures that experiments can be reproduced and validated independently.

The simulation results can be shared via a hyperlink generated when the user clicks the share button. These shared simulation results are stored temporarily in the server's memory, rather than being saved to the database. This approach helps prevent the database from growing uncontrollably due to the storage of multiple simulation results, ensuring that the system remains efficient and scalable. Since the data is stored in-memory, it is accessible as long as the server is running, but it is not permanently persisted, further reducing the load on the database.

By streamlining the simulation process, providing real-time results, and offering versatile download options, EasyModel creates a user-friendly environment that caters to researchers' needs for swift, accurate, and insightful analysis of biological models.

4.1.8.2 Simulation results list

When users click the back button from an individual simulation results view or select the *4. Simulation Results* button from the main menu, they are redirected to the simulation results list (see Figure 4.20). This view displays all simulations launched during the current session. Users can review the status and queue position of each simulation job. Additionally, they have the option to share or cancel pending simulation jobs.



Your Launched Simulations

Simulation Id	Status	Position in Queue	Copy URL	Cancel
00000001	Finished	-	🔗	✕
00000002	Finished	-	🔗	✕
00000003	Running	R	🔗	✕
00000004	Pending	1	🔗	✕
00000005	Pending	2	🔗	✕

Figure 4.20: EasyModel: Simulation results list. This view displays the simulations that had been launched during the current user session. Users can access the individual simulation results as well as share and cancel them.

The simulation job queue system continuously monitors the job queue to dispatch new jobs to the execution manager (see Section 3.6). The execution manager leverages the Mathematica calculus engine to generate the simulation results.

4.2 Benchmarking EasyModel

In this section we present the benchmark results of the implemented stochastic algorithms.

4.2.1 Benchmarking the stochastic algorithms

We chose two representative models from our database and use them to benchmark the stochastic algorithms. First, we benchmarked the accuracy of the stochastic algorithms, by running deterministic and stochastic simulations of the same model independently. If the stochastic algorithms are working properly, the results should be in agreement between the stochastic and deterministic simulations provided that the model is mono stable i.e. the model has only one steady state.

4.2.1.1 Benchmarking BioModel 101 TGF-β

First we chose EasyModel's local version of BioModel 101 TGF-β (71), imported from the BioModels database. This system models the pathway that responds to changes in the extracellular concentration of Transforming growth factor-β (TGF-β) in mammalian cells. The pathway helps cells interpret extracellular TGF-β into appropriate cell fate decisions.

The model contains 6 dependent variables and 13 reactions, all of them following mass action kinetics. It has one stable steady state i.e. the model is mono stable.

Using EasyModel, we performed three independent simulations using the deterministic method (see Figure 4.21A), the SSA method (see Figure 4.21B) and the τ-leaping method (see Figure 4.21C). Selected final time was three thousand seconds to reach the steady state. For the deterministic simulation we used the default 0.1 time step. For the stochastic simulation we set it to perform 4 replicates i.e. stochastic simulation runs, and the cell size was set to the Prokaryotic Cell size. We see that the deterministic and stochastic algorithms follow a similar time course, indicating that our implementation of the stochastic methods works properly (see Section 4.2.1.3 below for more details).

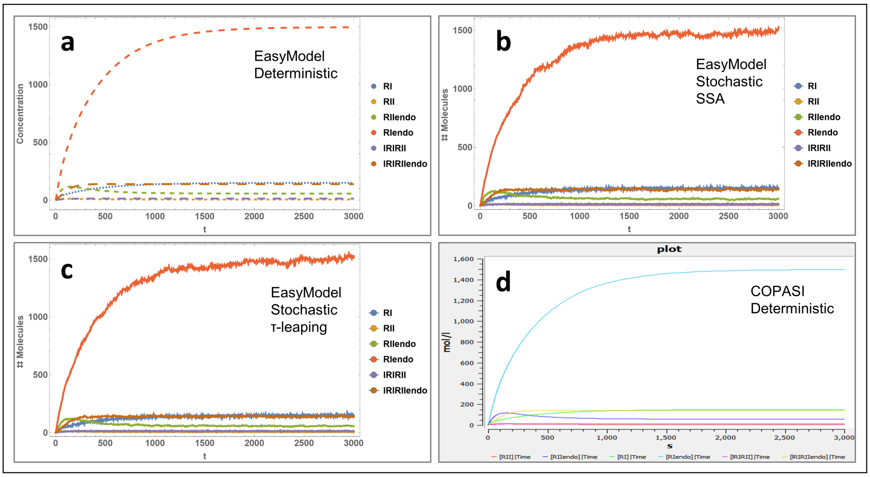


Figure 4.21: Studying the dynamic behavior of BioModel 101 TGF-β in a comparative benchmark between EasyModel and COPASI. The interplay of these variables allows the model to simulate the TGF-β signaling process, from ligand binding to gene expression, capturing both activation and regulation steps. **a** EasyModel deterministic dynamic simulation. **b** EasyModel stochastic SSA simulation. Median values for eight stochastic replicates. **c** EasyModel stochastic τ-leaping simulation. Median values for eight stochastic replicates. **d** COPASI deterministic dynamic simulation using the LSODA method.

Then we exported the model as a SBML file and imported it into COPASI to try the same types of simulations performed on EasyModel. The deterministic simulation performed in COPASI using the LSODA method resulted in a similar graph to the one performed by EasyModel (compare Figure 4.21A and Figure 4.21D).

We attempted to perform stochastic simulations on this model in COPASI but were unsuccessful. For the stochastic SSA method, we used the *Stochastic (Direct method)* option, and for the τ-leaping method, we selected the *Stochastic (Adaptive SSA/τ-Leap)* option. Performing the SSA method resulted in the exception *"CTrajectoryMethod (12): Internal step limit exceeded,"* and performing the τ-leaping method resulted in the exception *"CTrajectoryMethod (26): A tau-Leap step encountered numerical problems,"* thus preventing the model from being simulated. For more information about the typical errors encountered during our testing of stochastic simulations, refer to the text block below. To account for the possibility of an error in the exported SBML file, we imported it back into EasyModel, and repeated the simulations successfully.

Some of the errors we encountered trying to perform stochastic simulations in COPASI:

```
EXCEPTION: CTrajectoryMethod (12): Internal step limit
    exceeded.
The program has the option to increase the internal step
    limit. However, doing so often fails to resolve the issue,
    as the problem typically stems from the simulation
    becoming trapped in an infinite loop during the initial
    step calculation. This is likely due to a flaw in either
    the model specification (or SBML file) or the simulation
    algorithm.
--------------------------------------------------------
ERROR: At least one particle number in the initial state is
    too big.
At first glance, this may seem like a minor issue that could
    be resolved by adjusting the initial particle numbers of
    the species. However, significantly reducing the initial
    particle numbers can drastically alter the system's
    behavior, making this approach unsuitable.
--------------------------------------------------------
ERROR: At least one reaction is reversible. That means
    stochastic simulation is not possible.
This error can sometimes be addressed using a COPASI built-in
    function, though its applicability depends on the specific
    model and may not always be effective.
--------------------------------------------------------
EXCEPTION: CTrajectoryMethod (26): A tau-Leap step encountered
    numerical problems.
When encountering this error, it mostly indicates that the
    model is not compatible with the $\tau$-leaping method.
```

It is worth noting that although we were unable to simulate certain models in the stochastic regime in COPASI, we successfully simulated other models that were more appropriate for this type of simulation.

4.2.1.2 Benchmarking BioModel 148 Bone Remodeling

To further confirm the correct functioning of the stochastic algorithms implemented in EasyModel we benchmarked them using BioModel 148 (Bone Remodeling) (72). This system models the molecular network that regulates bone remodeling in vertebrates. This model was originally used to study two modes of bone remodeling. On the one hand the model correctly simulates healthy bone remodeling in response to different external stimulus. On the other it also correctly reproduces an unstable behavior similar to that of pathological bone remodeling in Paget's disease.

The model contains 3 dependent variables, 2 control variables and 6 reactions, all of them following mass action kinetics. The model doesn't reach steady state.

Using EasyModel, we performed three independent simulations using the deterministic method (see Figure 4.22A), the SSA method (see Figure 4.22B) and the τ-leaping method (see Figure 4.22C). Selected final time was ten thousand seconds. For the deterministic simulation we used the default 0.1 time step. For the stochastic simulation we set it to perform 4 replicates i.e. stochastic simulation runs, and the cell size was set to the Prokaryotic Cell size. We see that the deterministic and stochastic algorithms (both SSA and τ-leaping) follow a similar time course, indicating that our implementation of the stochastic algorithms works properly. In the EasyModel's stochastic simulations, the variable *x2* exhibits an initial upward movement, followed by a downturn, before eventually aligning with the trend observed in the deterministic algorithm. This behavior is due to the conversion from concentration numbers to molecule quantities that must be applied prior to running stochastic simulations. In fact, in the deterministic simulations, a closer look reveals that *x2* also shows a small initial upward movement followed by a slight downturn, before continuing to level off horizontally.

Then we exported the model as a SBML file and imported it into COPASI to try the same types of simulations performed on EasyModel. The deterministic simulation was similar to the one obtained in EasyModel (compare Figure 4.22A and Figure 4.22D).

We tried to simulate this model on COPASI under the stochastic regime and we encountered the same problems as previously described in the BioModel 101 Section 4.2.1.1. As before, to account for the possibility of an error in the exported SBML file, we imported it back into EasyModel, and repeated the simulations successfully.

4.2.1.3 Further validation of the stochastic τ-leaping implementation

A simulation that uses the τ-leaping algorithm mixes SSA and τ-leaping simulation time steps. In extreme cases, a simulation may only use τ-leaping or SSA time steps. This comes as a consequence that τ-leaps can only be made under specific numerical conditions, that are checked at every time step in the simulation by resolving auxiliary numerical tasks. If the numerical conditions for performing a τ-leap are met, the simulation time jumps ahead by a certain amount, bypassing many single SSA steps. The amount of time to leap depends on the auxiliary calculations. As an overall consequence of exten-

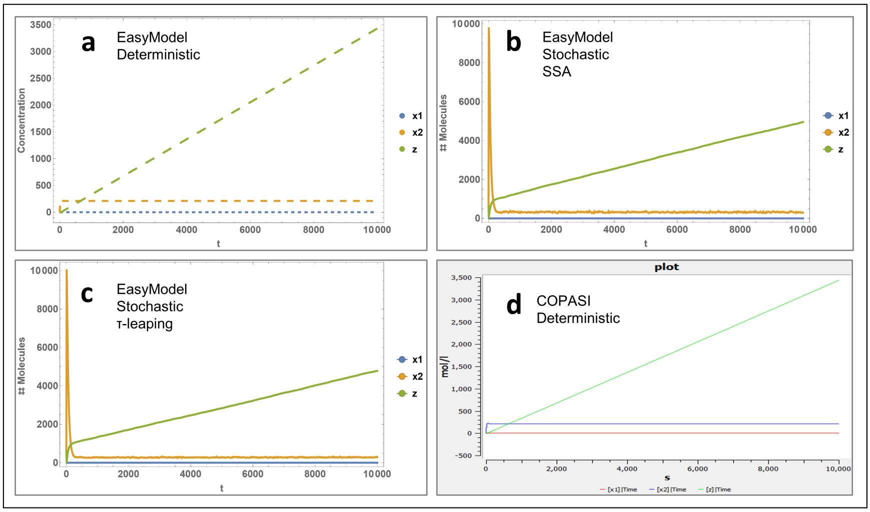


Figure 4.22: Studying the dynamic behavior of BioModel 148 Bone Remodeling in a comparative benchmark between EasyModel and COPASI. The interaction between these variables helps simulate the dynamic balance of bone remodeling, capturing both bone resorption (osteoclast activity) and bone formation (osteoblast activity). **a** EasyModel deterministic dynamic simulation. **b** EasyModel stochastic SSA simulation. Median values for eight stochastic replicates. **c** EasyModel stochastic τ-leaping simulation. Median values for eight stochastic replicates. **d** COPASI deterministic dynamic simulation the LSODA method.

sive τ-leaping being during the simulation is that computation times can be greatly reduced. If a τ-leap cannot be performed, the algorithm executes a small number of SSA time steps (usually 100), followed by resuming the auxiliary numerical tasks that permit checking if τ-leaps can be made. If τ-leaping checking operations always fail to allow τ-leaping, a pure SSA simulation will be performed instead, with the added overhead of the τ-leaping auxiliary operations. To know more about these auxiliary checking operations, please refer to Section 3.4.2.1.

When performing a τ-leap, several reactions are fired multiple times and the time point jumps forward the equivalent of many single SSA time steps. As a consequence, the time course plot for simulations that extensively use τ-leaping would show big gaps in the variables values between the leap time points. To avoid this visual effect, EasyModel applies linear interpolation between these leap time points.

BioModel 101 is an example of a system where the τ-leaping method is less efficient than the SSA method (see table 4.1). In this system, the τ-leaping method can only perform one small τ-leap, at the beginning of the simulation, making the full τ-leaping simulation slower than the SSA simulation. This is because in this case the τ-leaping method ends up using the SSA method for the whole simulation but one time step, with the overhead of the τ-leaping checking operations. When comparing 4.21B and 4.21C, we can see that the two curves are similar and have a similar ruggedness. This indicates that both have been calculated using the same SSA method.

BioModel 101 (Time range=[0,3000])					
Simulation type	**Exe. time (s)**	**Int. steps**	**No. Leaps**	**Leap time**	**Leap boost (%)**
Deterministic	0.01	30000	n/a	n/a	n/a
Stochastic (SSA)	5605.995	408391	n/a	n/a	n/a
Stochastic (τ-leap)	5669.395	406928	1	0.002227	-1.12
BioModel 148 (Time range=[0,10000])					
Simulation type	**Exe. time (s)**	**Int. steps**	**No. Leaps**	**Leap time**	**Leap boost (%)**
Deterministic	0.008	100000	n/a	n/a	n/a
Stochastic (SSA)	444.806	170272	n/a	n/a	n/a
Stochastic (τ-leap)	136.704	124200	1492	570.2720	+225.38

Table 4.1: EasyModel's efficiency of the alternative stochastic algorithms using BioModel 101 and BioModel 148 as benchmarks. Average values for eight runs are shown for the stochastic simulations.

In contrast, BioModel 148 is a good example of a model that can be more efficiently simulated using the τ-leaping method than using the SSA algorithm, as the required conditions for the algorithm to leap are met several times during the simulation, resulting in a significant reduction of the execution time. Simulation of this model shows that the τ-leaping code part of our implementation is working properly. By comparing Figure 4.22B and Figure 4.22C, we can see that the two curves are similar, indicating that both algorithms have similar accuracy. Still, the curves for the SSA algorithm are more ragged than those of the τ-leaping method, indicating that the later uses linear interpolation between the time leap points.

For information regarding on how to calculate the τ*-leaping boost*, please refer to the Section 3.4.2.2.

5 Use cases

The path to solving humanity's greatest challenges lies in our ability to innovate and apply technology effectively.

(Elon Musk)

In this section, we will delve into some of the key features of EasyModel and showcase examples of how you can use the tool to solve a variety of real-world problems. By following our comprehensive step-by-step instructions and tips, you'll be able to harness the potential of EasyModel's features to gain insights that can benefit your work in the biological field.

This section will extend on the information described in Section 4.1 about the usage of EasyModel.

5.1 Brusselator: Deterministic simulations

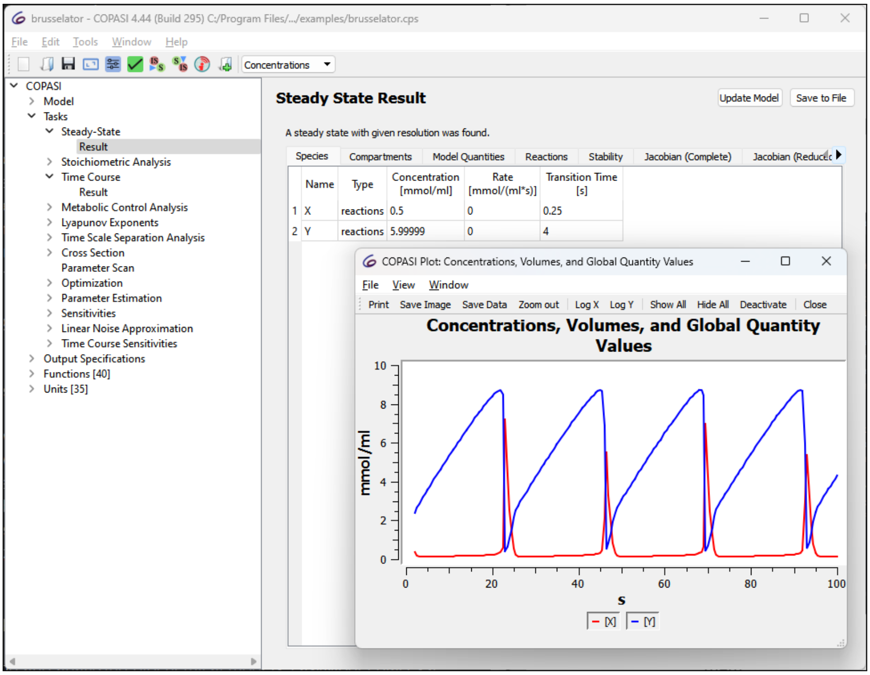
We will utilize the **Brusselator** model to showcase the deterministic features of EasyModel.

The *Brusselator* model is a classic theoretical model in systems biology used to describe oscillatory chemical reactions. It represents a simplified reaction network that exhibits periodic behavior, often used to study biochemical oscillations and pattern formation in biological systems.

The specifications of this *Brusselator* model are extracted from the example file *brusselator.cps* from the COPASI software (see Figure 5.1).

To perform simulations on this model, we will follow the Section 4.1 usage instructions. To begin, let's assume we have just landed on the home page of EasyModel. The first step would be to select the *Brusselator* model from the database. To do so, we first click on the *Public Repository* button to access the public model list (see Figure 5.2). Next we scroll down the list, select *Brusselator* and click on *Load Model* (see Figure 5.3).

From the *Model Builder* view, we can modify the model. In this case, no modifications will be made. Typical changes that do not alter the model's



Steady State Result

A steady state with given resolution was found.

	Name	Type	Concentration [mmol/ml]	Rate [mmol/(ml*s)]	Transition Time [s]
1	X	reactions	0.5	0	0.25
2	Y	reactions	5.99999	0	4

Figure 5.1: **Brusselator:** Simulation results for dynamic and steady state in the COPASI software.

structure include adjusting the initial concentrations of species, updating control variable values, and modifying numerical parameters within the rate laws associated with the model. The *Model Builder* interface is shown in Figure 5.4.

Next, we proceed to validate the model. If no structural changes have been made, the model will validate successfully, as all models in the database are verified to be valid. In the validation window, the stoichiometric matrix and the regulatory matrix are displayed (see Figure 5.5). These two matrices provide an alternative representation of the model's reactions. The stoichiometric matrix indicates the substrates and products of the reactions, while the regulatory matrix identifies the modifiers involved in the reactions. In this case, the regulatory matrix is filled with zeros, as no modifiers are present. Modifiers are represented by a value of one, in contrast to zero. If variable X_i modulates process P_j, then position {i, j} of the regulatory matrix will be one, instead of zero. To continue, click on the *Configure Simulation* button.

In the *Simulation Launcher* we select *Dynamic (Deterministic)* and *Steady*

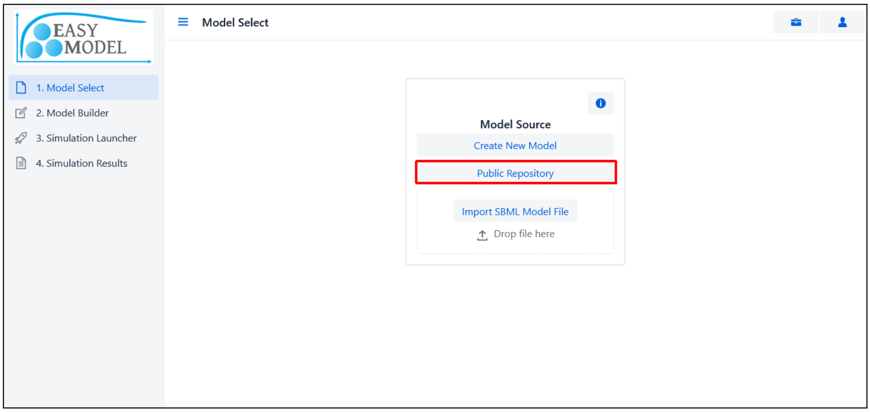
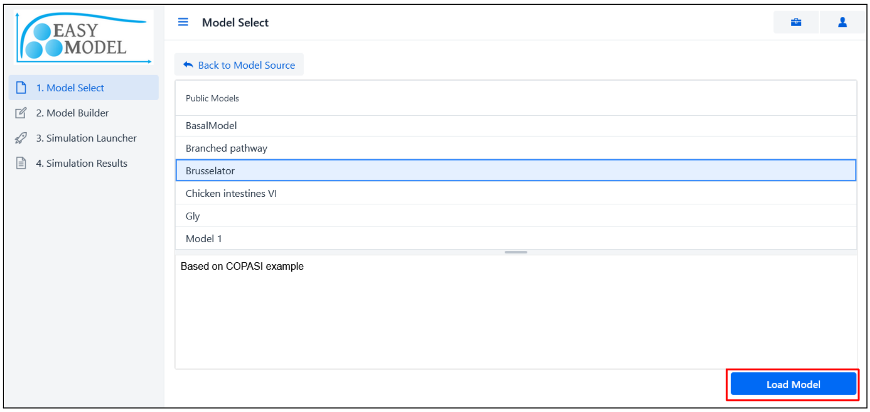


Figure 5.2: From the landing page, we click on *Public Repository*.



Public Models
BasalModel
Branched pathway
Brusselator
Chicken intestines VI
Gly
Model 1

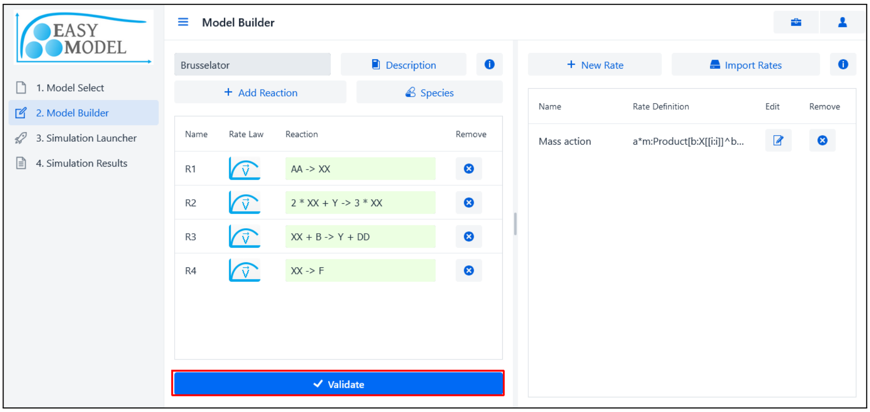
Based on COPASI example

Figure 5.3: We scroll down the list, select the *Brusselator* model and click on *Select Model*.

State. For dynamic simulation settings we set the final time to 100 (see Figure 5.6).

We open the *Analysis* tab and select *Gains* and *Sensitivities* to perform a local analysis on the control variables and rate laws parameters (see Figure 5.7).

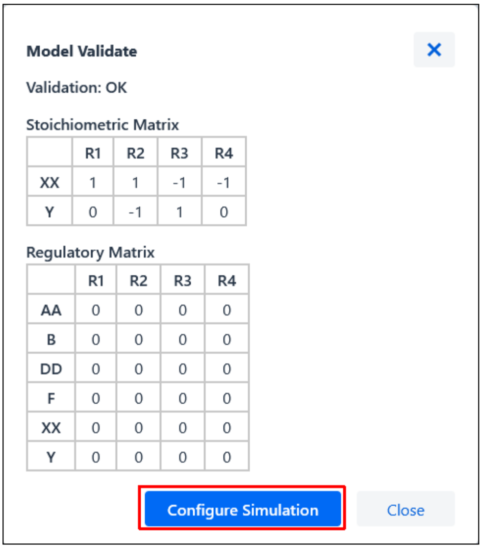
We open the *Plot Views* tab and add a new view to create a view that only contains the *Y* variable to analize more clearly the changes on this time-dependent variable (see Figure 5.8).



Name	Rate Law	Reaction	Remove
R1		AA -> XX	
R2		2 * XX + Y -> 3 * XX	
R3		XX + B -> Y + DD	
R4		XX -> F	

Name	Rate Definition	Edit	Remove
Mass action	a*m:Product[b:X[[i:i]]^b...		

Figure 5.4: **Model Builder:** Structure of the *Brusselator* model. As no changes are made, we click on the *Validate* button to proceed.



Model Validate

Validation: OK

Stoichiometric Matrix

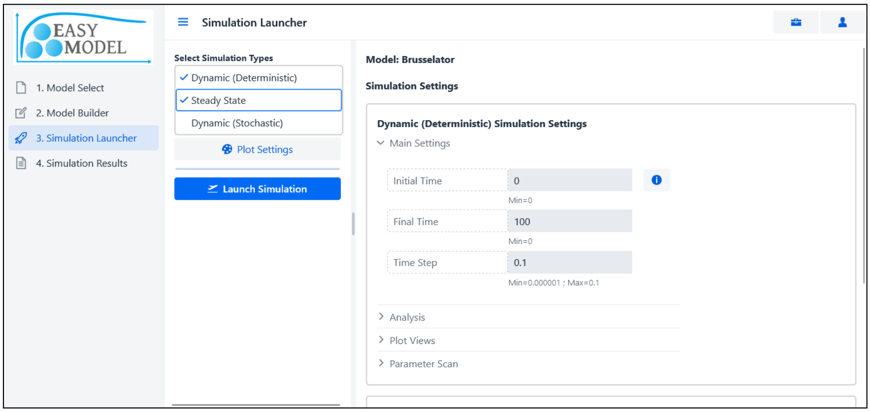
	R1	R2	R3	R4
XX	1	1	-1	-1
Y	0	-1	1	0

Regulatory Matrix

	R1	R2	R3	R4
AA	0	0	0	0
B	0	0	0	0
DD	0	0	0	0
F	0	0	0	0
XX	0	0	0	0
Y	0	0	0	0

Figure 5.5: **Validate model window:** Displays the stoichiometric and regulatory matrices.

Next open the *Analysis* tab in the *Steady State Simulation Settings* box. Check the *Stability* analysis checkbox (see Figure 5.9).


Simulation Launcher

1. Model Select
2. Model Builder
3. Simulation Launcher
4. Simulation Results

Select Simulation Types
✓ Dynamic (Deterministic)
✓ Steady State
Dynamic (Stochastic)

Plot Settings

Launch Simulation

Model: Brusselator

Simulation Settings

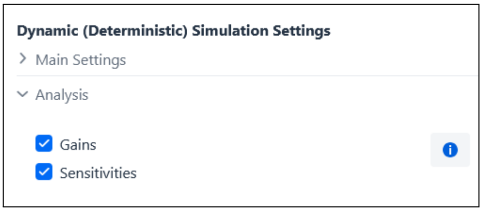
Dynamic (Deterministic) Simulation Settings

Main Settings

Initial Time	0
	Min=0
Final Time	100
	Min=0
Time Step	0.1
	Min=0.000001 ; Max=0.1

Analysis
Plot Views
Parameter Scan

Figure 5.6: **Simulation Launcher:** We select *Dynamic (Deterministic* and *Steady State*.


Dynamic (Deterministic) Simulation Settings

Main Settings

Analysis

✓ Gains
✓ Sensitivities

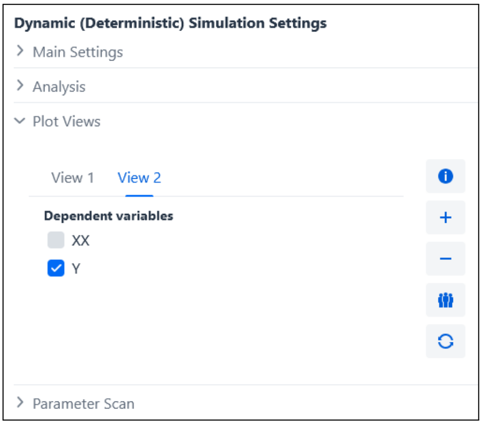
Figure 5.7: **Dynamic analysis:** We select *Gains* and *Sensitivities*.

We launch the simulation using the *Launch Simulation* button (see Figure 5.6) and wait for the simulation process to finish. We will now analyze the results.

The simulation results begins displaying the dynamic simulation in two plots: The first shows all the variables, and the second isolates the variable Y to make its behavior in the system clearer for analysis (see Figure 5.10). This separation will also be applied to subsequent analyses.

The simulation results also include the gains and sensitivities analysis. This analysis performs a local examination of how the system would behave if constant values and parameters in the rate laws were altered. While this analysis provides an approximate view—since it doesn't account for global system changes like in the *Parameter Scan* analysis—it is faster to execute.

We will use the *Absolute Dynamic Gains* plot to assess changes in the variable Y (see Figure 5.11). This plot evaluates changes in absolute values, in contrast


Dynamic (Deterministic) Simulation Settings

- Main Settings
- Analysis
- Plot Views

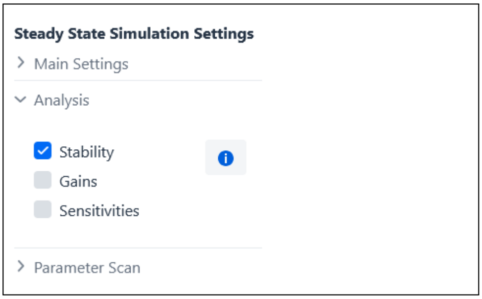
View 1 | View 2

Dependent variables

- [ ] XX
- [x] Y

- Parameter Scan

Figure 5.8: **Plot Views:** We add a new view that only contain the *Y* dependent variable.


Steady State Simulation Settings

- Main Settings
- Analysis
  - [x] Stability
  - [ ] Gains
  - [ ] Sensitivities
- Parameter Scan

Figure 5.9: **Steady State Simulation Settings:** Check the *Stability* analysis to add this analysis to the simulation job.

to relative values which are shown in a separate plot. *Gains* measure how modifications to the constants affect the system.

In the analysis, we observe that the constant *AA* significantly impacts the behavior of the dependent variable *Y* at specific time points: [23, 45, 68, 93]. From the plot, we can deduce that the influence of this constant increases approximately every ~22 time units and persists over time. The constant *B* also affects *Y* at these same intervals, but to a lesser degree.

Gains are denoted as: *G_<dependent_variable_name>_<constant_name>*. The same notation is used for sensitivities—using *S* as the initial character instead of *G*—which analyze how changes in parameter values affect the system.

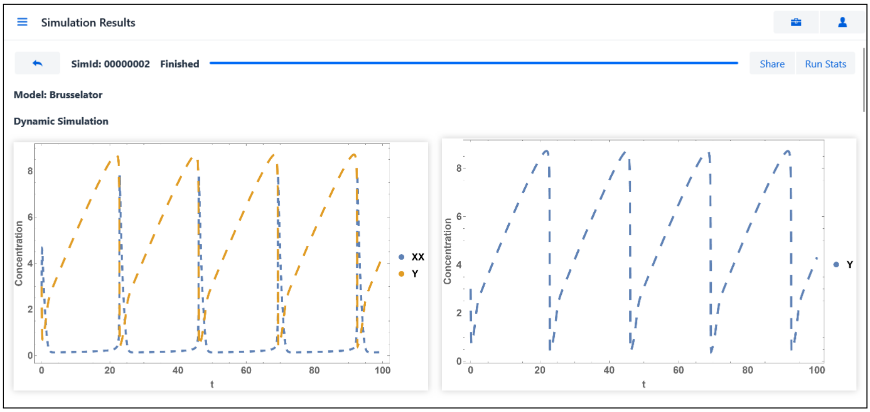


Figure 5.10: **Simulation Results:** The left plot displays the dynamic simulation for the full system, while the right plot displays only the evolution of the variable *Y*. Results are in accordance with COPASI's (see Figure 5.1).

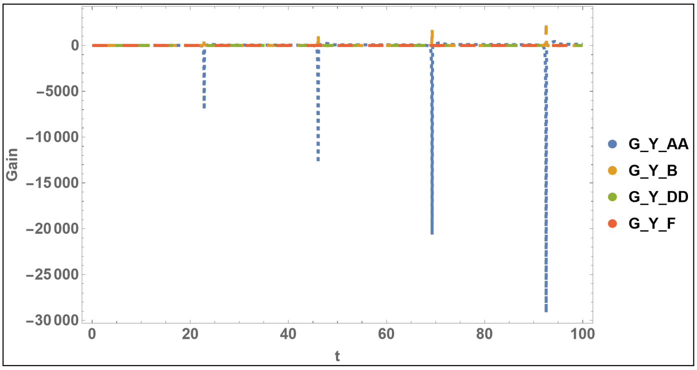
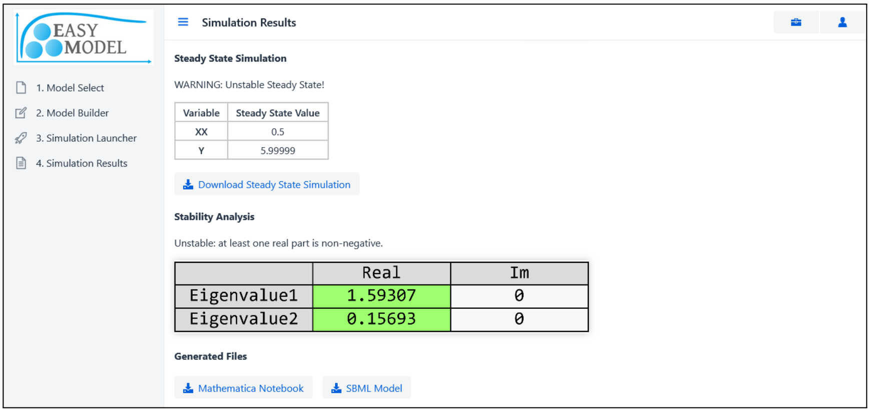


Figure 5.11: **Absolute Gains Plot:** This plot shows how value changes in the constants *AA* and *B* have an impact on the system's dependent variable *Y*.

To conclude this simulation example, we examine the *Steady State Simulation* results (see Figure 5.12). A steady state was identified, and the concentrations of the dependent variables at this state are shown in table format. However, a warning message indicates an *"unstable steady state,"* meaning that while a steady state has been found, it is not stable. By observing the dynamic plot of the system, we can see that the system continuously fluctuates between values 0 and ~9. A stable state would be represented as a perfectly stabilized horizontal line. In this case, because the steady state is unstable, we perform a dynamic simulation and observe a stable oscillation. The stable oscillation repeats itself periodically. The stability of the steady state, which is found via steady state simulations, is determined by calculating the Eigenvalues of the Jacobian matrix. For the steady state to be locally stable, all the real parts of the Eigenvalues must be negative (73).



Steady State Simulation

WARNING: Unstable Steady State!

Variable	Steady State Value
XX	0.5
Y	5.99999

Stability Analysis

Unstable: at least one real part is non-negative.

	Real	Im
Eigenvalue1	1.59307	0
Eigenvalue2	0.15693	0

Generated Files

Figure 5.12: **Steady State Simulation** results: Unstable steady state is found. Stability analysis delves in the analysis by displaying the Eigenvalues. Results are in accordance with COPASI's (see Figure 5.1).

In this use case, we have illustrated how to simulate and analyze the *Brusselator* model using EasyModel. This example provided an overview of the various stages involved in modeling, simulating, and interpreting the results for a well-known biochemical system. We explored different types of simulations, including dynamic and steady-state analyses, and discussed how to handle and interpret the outputs generated. The process included examining the behavior of the system under various conditions, assessing the stability of the steady states, and understanding the impact of different control variables on the system's dynamics. This example serves to demonstrate the practical application of EasyModel in studying complex biological systems.

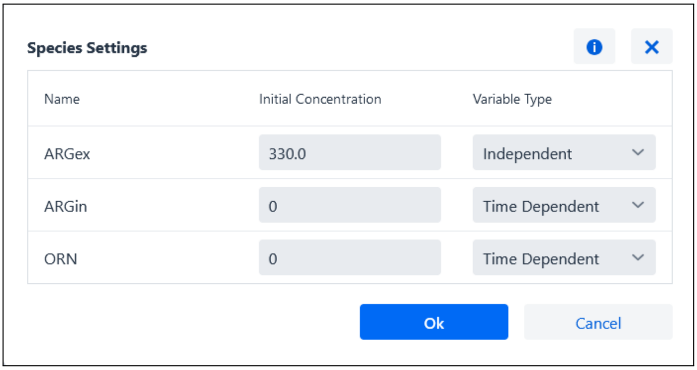
5.2 BioModel 191: Stochastic simulations

To showcase how to perform stochastic simulations, we will use the **BioModel 191** imported from the BioModels database.

BioModel 191, developed by Montañez et al. (2008), focuses on arginine catabolism in cellular metabolism. This model provides a comprehensive representation of the biochemical processes involved in the breakdown of arginine, an amino acid with crucial roles in cellular functions and metabolism.

To perform stochastic simulations, we will follow the guidelines of the previous Section 5.1). We begin by loading the model from the *Public Repository*. In the *Model Builder*, we open the *Species* dialog (see Figure 5.13) and set the initial concentrations of the time-dependent species *ARGin* and *ORN* to 0, ensuring that the plot curves start at 0 on the molecule count axis (Y-axis).

We leave the remaining model settings unchanged and proceed to configure the simulation by clicking the *Validate* button.


Species Settings

Name	Initial Concentration	Variable Type
ARGex	330.0	Independent
ARGin	0	Time Dependent
ORN	0	Time Dependent

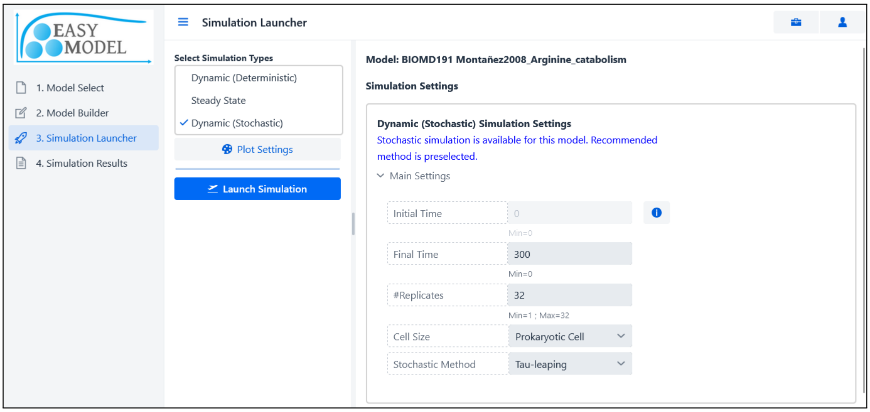
Figure 5.13: **Species Settings:** We set the initial concentrations of the time-dependent species *ARGin* and *ORN* to 0, ensuring that the plot curves start at 0 on the molecule count axis (Y-axis).

In the simulation configuration view (see Figure 5.14), we select the *Dynamic (Stochastic)* simulation to be performed. The specific settings for the stochastic simulation are as follows:

- *Final Time*: Set to 300 so it reaches equilibrium state.
- *Number of Replicates*: Set to 32, to obtain more precise plotting.
- *Cell Size*: Set to *Prokaryotic Cell*.

- *Stochastic Method*: Set to *Tau-leaping* as this model can benefit from it.

Then we launch the simulation by clicking the corresponding button.



Model: BIOMD191 Montañez2008_Arginine_catabolism

Simulation Settings

Dynamic (Stochastic) Simulation Settings

Stochastic simulation is available for this model. Recommended method is preselected.

Initial Time: 0 (Min=0)

Final Time: 300 (Min=0)

#Replicates: 32 (Min=1 ; Max=32)

Cell Size: Prokaryotic Cell

Stochastic Method: Tau-leaping

Figure 5.14: **Simulation Launcher:** We select the *Dynamic (Stochastic)* simulation to be performed and set the specific settings for the stochastic simulation.

In the *Simulation Results* view, we analyze the stochastic simulation results. When stochastic simulations are performed, a statistical table is produced to measure the performance of the stochastic algorithm (see Figure 5.15). This table measures, for each stochastic replicate—or run—the following metrics:

- *Replicate*: Indicates the measured stochastic replicate.

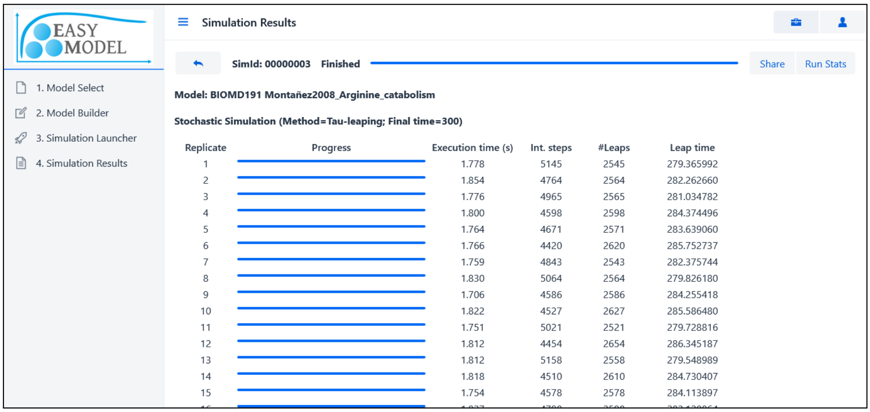
- *Progress bar*: Shows the progress of the replicate simulation in real-time.

- *Execution time*: The time, in seconds, that it took to finish the replicate calculation.

- *Internal steps*: The number of time steps or time points created on the time axis (X-axis) of the plot. Due to the nature of stochastic simulations, steps are not uniformly distributed as in deterministic simulations, but are randomly generated depending on the model, yielding a more accurate representation of behavior.

- *Number of leaps*: A metric for the τ-leap method, indicating how many leaps were executed. The more leaps, the more effectively the algorithm performs on the selected model.

- *Leap time*: A metric for the τ-leap method, indicating how much time has been leaped during the simulation. During the leaped time, the algorithm avoids executing the time-consuming SSA steps, saving computational resources. Larger values suggest higher efficiency of the algorithm on the selected model.



SimId: 00000003 Finished

Model: BIOMD191 Montañez2008_Arginine_catabolism

Stochastic Simulation (Method=Tau-leaping; Final time=300)

Replicate	Progress	Execution time (s)	Int. steps	#Leaps	Leap time
1		1.778	5145	2545	279.365992
2		1.854	4764	2564	282.262660
3		1.776	4965	2565	281.034782
4		1.800	4598	2598	284.374496
5		1.764	4671	2571	283.639060
6		1.766	4420	2620	285.752737
7		1.759	4843	2543	282.375744
8		1.830	5064	2564	279.826180
9		1.706	4586	2586	284.255418
10		1.822	4527	2627	285.586480
11		1.751	5021	2521	279.728816
12		1.812	4454	2654	286.345187
13		1.812	5158	2558	279.548989
14		1.818	4510	2610	284.730407
15		1.754	4578	2578	284.113897

Figure 5.15: **Simulation Results:** The initial part of the stochastic results include a performance table of the stochastic simulation.

A significant metric that we can observed on this stochastic simulation is the *Leap time*, as it covers around the 93% of the time to be simulated. This means most of the simulation can be produced by leaping time instead of producing single SSA steps. Thus the τ-leap method is effective on this model.

The median plot produced smooth curves as 32 replicates were performed (see Figure 5.16). The greater the number of replicates, the smoother the resulting curves typically become. This is because a larger number of replicates helps to average out random fluctuations, providing a more accurate representation of the underlying trends and reducing the impact of outliers.

From the plot, we observe that the system reaches equilibrium or a steady state. Performing a steady-state stability analysis confirms this assumption: A stable steady state is achieved, as all the real parts of the Eigenvalues are negative.

Next, we study the linear noise analysis of the stochastic simulation shown in Figure 5.17. In Figure 5.17A, we observe the 32 different trajectories calculated for the *ARGin* variable. This plot provides a broad idea of the variability of the variable under the stochastic regime. In Figure 5.17B, we observe the median across all trajectories along with the quantile deviation from the median, giving

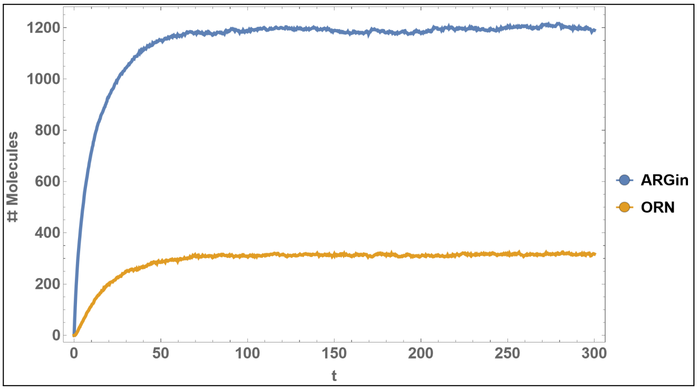


Figure 5.16: **Stochastic Median Plot:** A general stochastic plot that consolidates all the calculated trajectories by applying a median function across the replicates.

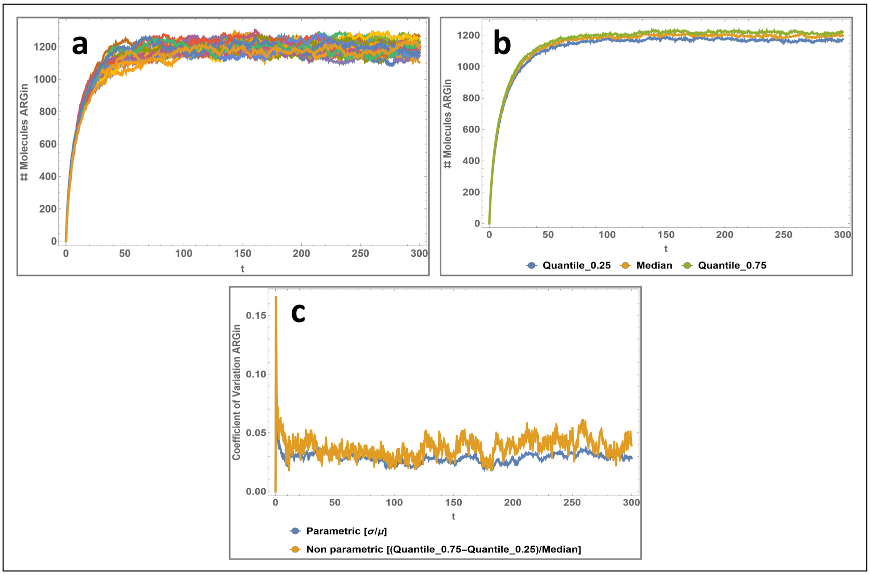


Figure 5.17: **Linear Noise Analysis.** **a** Raw simulation trajectories calculated for the *ARGin* variable. **b** Quantile plot for the *ARGin* variable. **c** Coefficient of Variation for the *ARGin* variable.

a more precise interpretation of the results. Finally, in Figure 5.17C, we observe the coefficient of variation, which indicates how much the deviation differs from the median. Lower values reflect less variability in the variable across all the trajectories.

Overall, stochastic simulations offer a more precise representation of the system. Although these simulations often result in jagged curves, increasing the number of replicates helps smooth the output and produce more reliable results. Additionally, the *Linear Noise Analysis* assists researchers in assessing the validity of the simulation by providing insights into the system's inherent variability.

5.3 BioModel 101: Parameter Scan

To illustrate how *Parameter Scan* can be used to interrogate models we performed two experiments using **BioModel 101**.

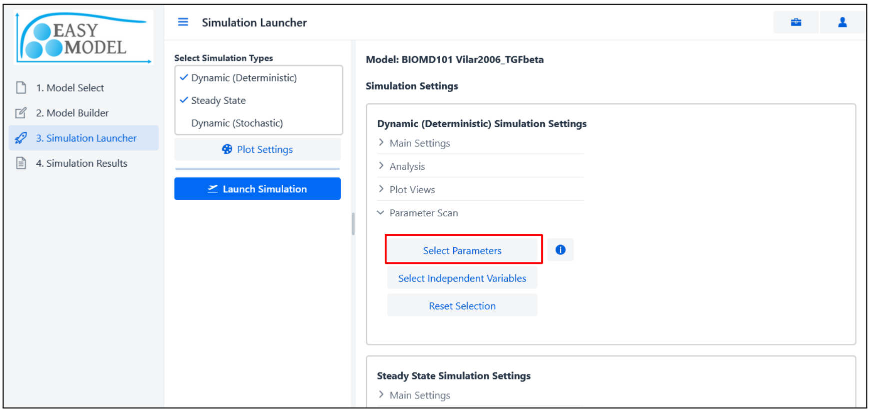
BioModel 101, developed by Vilar et al. (2006), focuses on the TGF-beta signaling pathway. This model provides a detailed representation of the signaling cascade triggered by Transforming Growth Factor-beta (TGF-beta). It aims to understand how TGF-beta influences cellular processes such as growth, differentiation, and apoptosis.

This use case will illustrate the *Parameter Scan* functionality under the deterministic regime. Currently, *Parameter Scan* is not available for the stochastic regime, as it would be significantly more computationally expensive.

To perform the *Parameter Scan*, we will follow the guidelines of the previous Section 5.1). We first select the model from the public repository, validate it and proceed to simulation configuration. Once in the simulation configuration view, we select the simulation types *Dynamic (Deterministic)* and *Steady State* (see Figure 5.18). For dynamic simulations we set the final time at 3000. For the Steady State simulations we used the default settings.

After adjusting the general settings, we configure the *Parameter Scan* by clicking the *Select Parameters* button under the *Parameter Scan* tab in both simulations. A dialog allows to setup the *Parameter Scan*, where we configure one of the rate law parameters (see Figure 5.19).

It is well known that extracellular ligands induce the binding between receptor types RI and RII in the membrane, followed by receptor internalization (71). Here we ask about the effect of ligands that increase the affinity between the two types of receptors on the dynamic behavior of the system. To measure this effect we focus on parameter *ka* from the binding reaction between RI and RII (Reaction 1). Low values for *ka* are equivalent to ligands that induce low affinity binding between receptors. As *ka* increases so does the affinity between RI and



Select Simulation Types
- ✓ Dynamic (Deterministic)
- ✓ Steady State
- Dynamic (Stochastic)

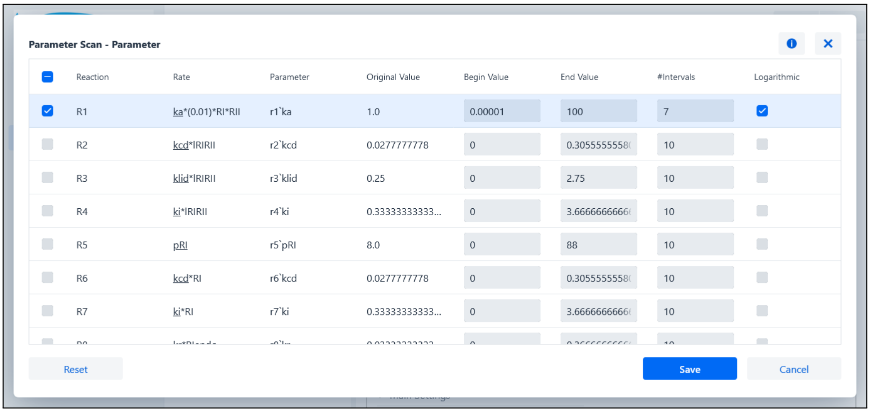
Model: BIOMD101 Vilar2006_TGFbeta

Simulation Settings

Dynamic (Deterministic) Simulation Settings
- Main Settings
- Analysis
- Plot Views
- Parameter Scan

Steady State Simulation Settings
- Main Settings

Figure 5.18: **Simulation Launcher:** We select the simulation types *Dynamic (Deterministic)* and *Steady State*.



Parameter Scan - Parameter

	Reaction	Rate	Parameter	Original Value	Begin Value	End Value	#Intervals	Logarithmic
✓	R1	ka*(0.01)*RI*RII	r1`ka	1.0	0.00001	100	7	✓
	R2	kcd*IRIRII	r2`kcd	0.0277777778	0	0.30555555558	10	
	R3	klid*IRIRII	r3`klid	0.25	0	2.75	10	
	R4	ki*IRIRII	r4`ki	0.33333333333...	0	3.66666666666	10	
	R5	pRI	r5`pRI	8.0	0	88	10	
	R6	kcd*RI	r6`kcd	0.0277777778	0	0.30555555558	10	
	R7	ki*RI	r7`ki	0.33333333333...	0	3.66666666666	10	

Figure 5.19: **Parameter Scan - Parameter:** We configure the parameter scan to be performed on the *ka* parameter from reaction 1.

RII. The *Parameter Scan* configuration can be found in Table 5.1. We will use the values to fill the *Parameter Scan* configuration dialog (see Figure 5.19).

We launch the simulation and discuss the simulation results.

In Figure 5.20 (Dynamic simulations) we can see that when we change *ka*, the time to reach steady state remains fairly constant for all species but the internalized receptor forms, IRIRII and IRIRIIendo. Reaching steady state for these variables is faster as *ka* increases.

Parameter	Begin Val	End Val	No. Intervals	Scale
ka (from reaction 1 rate)	10^{-5}	10^{2}	7	Logarithmic

Table 5.1: *Parameter Scan* settings for dynamic and steady state simulations. BioModel 101.

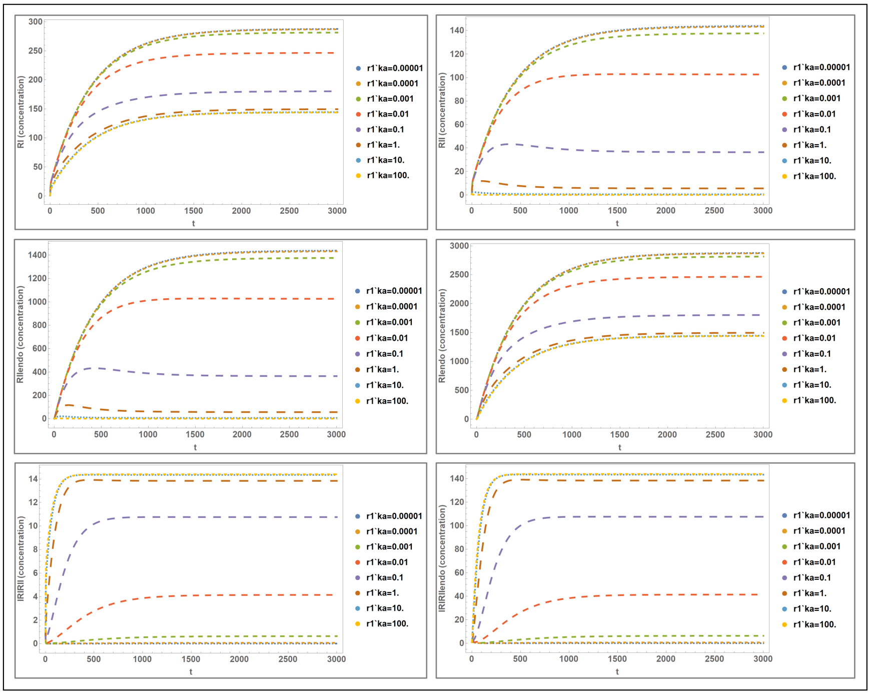


Figure 5.20: Studying the effect of changing parameter *ka* from reaction 1 on the dynamic behavior of BioModel 101 TGF-β. We scan *ka* between 0.00001 and 100 using 7 intervals evenly spaced on logarithmic scale. We plot each dependent variable of the model.

In Figure 5.21 (Steady State simulations) we see that increasing *ka* decreases the steady state concentrations for the free receptors and increases the steady state concentrations for the bound forms of the receptors. We also see that the system only responds to stimulus that make *ka* be roughly between 0.001 and 1. Outside of this range of values the system has very similar steady states.

Parameter Scan is a powerful functionality that allows the user to globally explore how parameters and control variables affect the system. This global

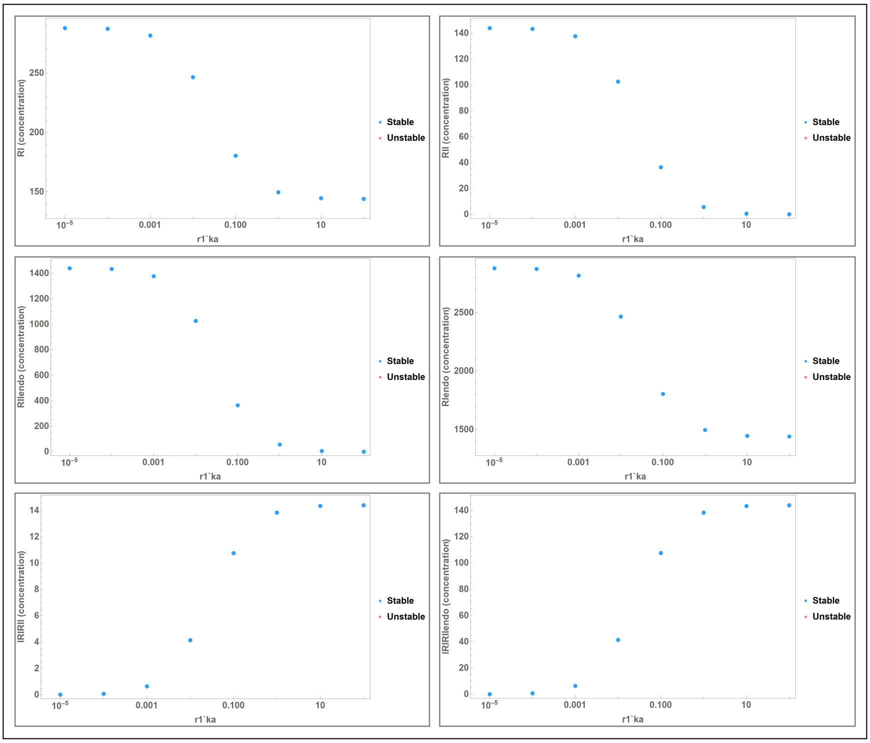


Figure 5.21: Studying the effect of changing parameter *ka* from reaction 1 on the steady state of BioModel 101 TGF-β. We scan *ka* between 0.00001 and 100 using 7 intervals evenly spaced on logarithmic scale. We plot each dependent variable of the model.

analysis performs multiple complete simulations, systematically varying the values of the parameters under study. It provides a more comprehensive analysis than sensitivity analysis, which offer local, approximate estimations of how parameters influence the system.

In this use case, we observed that the system's variables are sensitive to changes in the parameter *ka* in reaction 1, with variations spanning over two orders of magnitude.

6 Global discussion

The essence of science is not in the results but in the process of inquiry that leads us to understanding the universe.

(Richard Feynman)

In this chapter, we discuss the results obtained from this thesis work.

6.1 User interface

One of the primary objectives of EasyModel was to provide a user-friendly interface, aimed at reducing the learning curve typically associated with tools of this kind for both novel and expert users. In consequence, we minimized the number of screen changes required to use the application as much as possible. The application is designed to be used in four consecutive logical steps:

1. Select the model (or create a new one)
2. Modify or build the model
3. Configure the simulation
4. Get the simulation results

The interface guides the user through these conceptual steps in the modeling and simulation process. The initial screen deals with model selection or creation. The second screen deals with model implementation and validation. The third screen deals with model simulation and analysis settings and the fourth and final screen presents the results. While it would be possible to put all functionality in one screen, we believe this approach would make EasyModel less user-friendly, because it would contain too much information and options to select from simultaneously. A short tutorial is also provided during access to the application to the new users, in order to facilitate its use. This tutorial remains available to consult from the *Tools* option found at the top bar. In addition, there are *Information* buttons all across the application to keep the user informed on what to do next.

We believe we have successfully developed a user-friendly interface by leveraging the Vaadin UI web framework. The intuitive design not only simplifies the user experience, significantly reducing the learning curve for new users, but also enhances productivity for expert users, enabling them to work more efficiently and complete tasks with greater agility. The interface design focuses on accessibility and ease of use, making advanced modeling tools more approachable for a wider audience.

6.2 Intended use and comparison with other tools

EasyModel's main objective is to provide a user-friendly experience for modeling, simulation and analysis of models in systems biology. While many tools are available for doing these tasks, they are often difficult to use by the novel user because either they present a complex User Interface (UI), require knowledge of a programming language or both. EasyModel circumvents these issues by providing a user-friendly experience that uses a powerful mathematical calculus platform without requiring programming knowledge. In addition, if the user is more advanced, s/he can download the simulation script and personalize it for more advanced analysis using their own Mathematica license.

While there is a wide range of simulation applications for systems biology, JWS Online (6, 32) is the most similar to EasyModel. JWS Online also uses Wolfram Mathematica as the underlying calculus engine. While EasyModel targets novel users, JWS Online targets more advanced users, allowing them to perform different types of analysis and simulations, and providing model schema visual representation, reaction plots, etc. The application has databases for users, models, simulations and manuscripts. It also has SBML compatibility as EasyModel. Conveniently, it has online documentation.

JWS Online's interface seems to be less intuitive than EasyModel's for novel users. We infer this from limited empirical comparative testing with four undergraduate classes of biomedical and biotechnology students. Our interaction with these students suggests that EasyModel has a faster learning curve than JWS Online.

EasyModel also provides features that we haven't found in JWS Online: Stochastic simulations, automated implementation of kinetic laws derived from structured mathematical formalisms, the ability to download the full model, simulation, and analysis in the form of a Mathematica notebook (JWS Online allows to experimentally download the model as a Mathematica notebook), etc.

Another tool on which EasyModel draws inspiration is COPASI (26, 27). Features that both applications have in common include: Deterministic and stochastic simulations, parameter scanning, SBML compatibility, etc. A differentiating trait between EasyModel and COPASI is that the later is a standalone software (available for Linux, Windows and macOS) with a fixed set of functionalities, while the former gives the possibility that users can program locally more sophisticated analysis using the Mathematica language.

There are other simulation tools like the APMonitor Optimization Suite (33, 34, 35) that take a different approach to modeling. This type of software uses their own modeling language to define the models in plain text as a script. This gives a lot of flexibility to the programmers as they can introduce new features more easily to their own language. Also they can include a lot of features to the program, centering most of the effort on the coding area. This kind of tool is very profitable for expert users as it can offer a wide range of solutions. Yet, because of this, the program becomes difficult to use and understand, as it has a much slower learning curve than a tool like EasyModel. APMonitor has online documentation and the team has uploaded several webinars trough the years in order to educate its users. APMonitor is freely available as a website, MATLAB toolbox and a Python package. The developers also offer the GEKKO Python package for dynamic optimization. We think again that both kind of tools are very profitable for different types of user.

Overall, we believe that EasyModel has successfully differentiated itself from other tools by focusing on delivering a user-friendly experience through an intuitive GUI that aids both novel and expert users. The application caters to advanced users by offering the option to download the generated Mathematica code for further, more sophisticated applications. Thanks to its SBML format compatibility, EasyModel enables seamless model exchange and integration with other software programs, enhancing collaboration with other research groups.

6.3 Results

The results of this thesis revolve around the creation of the EasyModel tool.

EasyModel provides a user-friendly web interface accessible from anywhere, designed to offer a quick-to-learn experience. With its intuitive design, tutorial guides, and information buttons throughout the application, EasyModel is ideal for newcomers to systems biology modeling, offering a shallow learning curve. Advanced users can also benefit from quickly mastering the tool, thereby boosting their productivity. Additionally, EasyModel provides advanced analysis options and allows users to download the generated Mathematica code. By adopting a web application design and a simulation job queue system, users can leverage the computational power of Mathematica without needing their own license or high-performance machine.

EasyModel allows users to exchange and communicate models with other software programs through its compatibility with the SBML format. This feature facilitates collaboration among research groups and ensures that models created or modified in EasyModel can be shared, edited, and analyzed in other systems biology tools that support SBML.

Moreover, the platform supports a wide array of model analysis techniques, including parameter scans, dynamic simulations, and steady-state analysis. These features provide users with the capability to deeply interrogate the behavior of their models under various conditions. Advanced users can explore complex relationships between variables, observe how parameters influence model behavior over time, and assess system stability through Eigenvalue analysis.

Another key feature of EasyModel is the stochastic simulation capability. By supporting both the SSA (Stochastic Simulation Algorithm) and τ-leaping methods, users can analyze the inherent randomness of biological processes. This is especially relevant for models with small molecule populations where stochastic effects play a significant role. While the SSA provides exact trajectories, the τ-leaping method offers a computationally faster alternative for models that meet the appropriate conditions, making large-scale simulations more feasible.

In terms of performance, EasyModel's primary objective was not focused on execution speed but rather on usability. Software performance depends on various factors such as code optimization, the choice of programming language, and the hardware on which the software runs. The developers have made efforts to enhance performance by writing efficient Java and Mathematica code for the application's backend. The Vaadin framework ensures a responsive user interface aligned with modern design trends. When compared to other tools, we can consider the current performance of EasyModel as acceptable, striking a balance between usability and speed.

When implementing the τ-leaping variant of the stochastic dynamic simulation algorithm, we assessed its performance gains using the classic SSA algorithm as a benchmark. The τ-leaping algorithm provides benefits only when applied to certain models that meet its specific conditions. If a model qualifies, the algorithm can leap over multiple individual SSA simulation steps, significantly reducing execution time. The skipped time intervals are filled with linear interpolation, ensuring the plot's continuity. Our analysis of execution time improvements from the τ-leaping method showed significant results, reducing execution time by more than half. This confirms that incorporating this algorithm was a valuable enhancement to the system.

In conclusion, EasyModel offers a comprehensive toolset for both novice and advanced users in systems biology. Its focus on usability, coupled with robust analytical capabilities and compatibility with existing formats like SBML, ensures that it is well-suited for a wide range of applications in biological modeling, simulation, and analysis. Additionally, EasyModel provides users with the ability to download the Mathematica notebook corresponding to their model, enabling further customization and exploration beyond the platform's built-in features. This makes it particularly attractive for advanced users who wish to delve deeper into specific mathematical or computational aspects of their models.

By lowering the learning curve and maintaining high performance in simulations, EasyModel effectively bridges the gap between ease of use and advanced functionality, making it a valuable addition to the tools available in systems biology research.

6.4 Use cases

EasyModel provides a comprehensive range of functionalities within the field of systems and synthetic biology.

Some of the key use cases it supports include:

- **Model editing:** Users can create or modify models from the public repository without affecting the original database entries. Registered users can save their models to their private repository.
- **Deterministic simulations:** Users can perform time course simulations and steady-state analysis under the deterministic regime, including stability, gains and sensitivities analyses.
- **Stochastic simulations:** Users can perform stochastic simulations, featuring advanced algorithms like the τ-leaping method to analyze the inherent randomness in biological systems. Users can also visualize the stochastic noise analysis.
- **Parameter scanning:** Users can utilize parameter scan functionalities under deterministic regimes, enabling global gains and sensitivities analyses to assess how various parameters influence system behavior.
- **SBML import/export:** Users can import and export models in SBML format, allowing them to save models for future use or to work with other tools.

- **Mathematica code export:** Advanced users can perform a simulation and export the generated Mathematica code for further customization and more complex computational analyses outside of the platform.

- **Model publication:** Registered users can save new models into the database for public sharing. Models are validated for publication after two weeks of no modifications. Publication is then done automatically by the system.

- **Model publication:** Registered users can save new models into the database for public sharing. Models are validated for publication after two weeks of no modifications. Publication is then done automatically by the system.

- **Advanced kinetic rate laws:** Users can use structured mathematical formalisms as kinetic rate laws, a feature uncommon in most similar platforms, catering to more advanced modeling needs.

- **Sharing simulation results:** Users can share simulation results through a simple hyperlink, facilitating collaboration and review.

- **Administration panel:** For users with administrator privileges, the platform provides a dedicated custom-made panel to manage database entries.

In conclusion, EasyModel supports a wide range of use cases in systems and synthetic biology, enabling users to create, edit, and simulate models with both deterministic and stochastic approaches. The platform's capabilities, including model publication and result sharing, foster collaboration and extend its utility across diverse research needs, solidifying EasyModel as a valuable tool for both educational and professional applications.

7 General conclusions and future directions

When something is important enough, you do it even if the odds are not in your favor.

(Elon Musk)

To conclude this thesis, we highlight the key features of EasyModel and explore potential future developments.

7.1 Conclusions

This thesis focuses on the development of the EasyModel tool for modeling, simulation, and analysis in systems and synthetic biology. EasyModel makes a significant contribution to the community by offering a user-friendly experience that caters to both novice and experienced users. The intuitive interface not only reduces the learning curve but also enhances productivity by minimizing errors and shortening the time required for users to complete tasks.

Since its inception, EasyModel has undergone continuous improvement, with the latest version 2.4 (see Section 3.10) reflecting substantial advancements. Initially designed to provide a user-friendly interface for harnessing the powerful Wolfram Mathematica calculus engine, EasyModel has now evolved into a robust platform. The current version incorporates advanced functionalities beyond its original goals, including support for stochastic simulations and parameter scanning.

By leveraging technologies such as Java EE, the Vaadin framework, and Mathematica for backend computations, EasyModel delivers a seamless and efficient user experience within a web-based application that is easily accessible from any location. Notably, users are not required to register on the platform, further enhancing its accessibility.

Looking ahead, EasyModel has the potential to expand its capabilities and incorporate additional computational methods and tools, ensuring that it continues to serve the evolving needs of the systems biology community.

The following list summarizes the conclusions drawn from the development and implementation of EasyModel:

- We conclude that EasyModel is presented in a streamlined user interface that encourages deeper exploration of the tool and reduces complexity for users.

- We conclude that EasyModel is accessible from anywhere and without user registration, eliminating the requirement for a Mathematica license.

- We conclude that EasyModel includes a simple model editor with the ability to import predefined kinetic rate laws, making model creation more accessible.

- We conclude that EasyModel appropriately handles dynamic and steady state simulations.

- We conclude that Easy model appropriately handles deterministic and stochastic ODE simulations.

- We conclude that EasyModel can perform several analyses: steady state stability analysis, gain/sensitivity analysis under deterministic regime and noise analysis under stochastic regime.

- We conclude that EasyModel appropriately handles parameter scanning for dynamic and steady state simulations, under deterministic regime.

- We conclude that EasyModel is able to export models and simulations in the form of Mathematica notebooks for further offline customization and analysis.

- We conclude that EasyModel packages a large collection of models, including partially those from the BioModels database as well as user-contributed models and other testing models.

- We conclude that EasyModel is compatible with the SBML file format, ensuring model exchange with other software used in systems biology.

- We conclude that EasyModel facilitates its use by using a simulation job queue system, which enables users to submit new simulation jobs at any time.

- We conclude that EasyModel allows its users to share their simulation results through hyperlinks, enabling easy collaboration and data exchange.

- We conclude that EasyModel allows its users to publish new created models within the platform, fostering collaboration within the EasyModel community.
- We conclude that EasyModel supports structured mathematical formalisms as rate laws, catering to more complex modeling needs.
- We conclude that EasyModel allows its administrator users to access the administrator panel, which enables easy management of the database entries.
- We conclude that EasyModel appropriately includes a tutorial and information buttons throughout the interface, assisting users in navigating the tool and performing tasks efficiently.

The following list summarizes the conclusions drawn from comparing EasyModel to other tools:

- **JWS Online:** EasyModel offers a more user-friendly approach to its users. EasyModel supports stochastic simulations, user-defined mathematical formalisms as kinetic functions and allows to modify the models from the public repository before simulation. These features aren't available in JWS Online.
- **COPASI:** EasyModel is more easy to learn and use. EasyModel is more accessible as it is directly accessed through a website. EasyModel offers a public model database, sharing of simulation results via hyperlinks and user-defined mathematical formalisms as rate laws, features that are not available in COPASI. EasyModel allows to export the Mathematica code, which allows for advanced local analyses.
- **APMonitor:** EasyModel's emphasis on a user-friendly interface allows for easier model construction and simulation without needing specialized knowledge as learning a particular syntax code. EasyModel is compatible with the SBML file format. APMonitor does not natively support the SBML format, but a converter tool, *SBFC: Systems Biology Format Converter*, can be used to translate APMonitor's model format into SBML format.

In conclusion, EasyModel demonstrates its value by addressing the needs of both novice and expert users in the field of systems biology. Its intuitive interface, combined with advanced functionality, makes it a unique and powerful tool for the community. Future development of EasyModel will be discussed in the next Section 7.2, *Future directions*.

7.2 Future directions

As this work revolves around the EasyModel tool, future efforts will likely focus on expanding and enhancing its current functionalities. From a technical standpoint, the potential to improve EasyModel is vast. By leveraging the robust capabilities of the Java programming language, the modern and sleek user interface provided by the Vaadin web UI framework, and the computational power of Wolfram Mathematica, we can further evolve this tool to meet advanced goals in systems biology.

In terms of scope, there are numerous functionalities in systems biology that could be integrated into EasyModel. As discussed in Chapter 2, *Related Work*, other tools offer features currently unavailable in EasyModel. We could expand EasyModel's functionality to match these tools, while maintaining and enhancing its user-friendly approach. This philosophy has guided EasyModel's development since its inception. The key difference lies in EasyModel's commitment to implementing similar features in a more accessible and intuitive manner, with improvements whenever possible.

We consider that at this point with the current version 2.4 we have implemented the staple features as well as others more advanced. Through the different versions we have refined the user experience to make it more friendly while gathering feedback from the users to understand better their needs and provide more value in the next version release. We have always tried to keep a proactive attitude towards this project and to learn new knowledge along the way.

For future releases of EasyModel, we envision the addition of several advanced features, which are summarized in the following list:

- **Automatic BioModels Updates:** Implement unattended and periodic updates of the BioModels (65).
- **SBML Test Suite Adaptation:** Adapt EasyModel to conform to the *SBML Test Suite* for enhanced validation of the simulation algorithms and SBML compatibility. The SBML Test Suite is a conformance testing system for SBML that consists of a collection of test models and a framework for running software through the suite.
- **Model Branching:** Enable the creation of branches from preexisting models, allowing users to complicate or simplify them as needed.
- **Model Merging:** Provide functionality to merge individual models into a single cohesive model.
- **Bifurcation Analysis:** Incorporate bifurcation analysis (74).

- **Chemical Reaction Network Theory Analysis:** Implement analysis features based on chemical reaction network theory (75).

- **Spatially Non-Homogeneous Models:** Enhance the capability to handle spatially non-homogeneous models, utilizing tools such as partial differential equations (PDEs) (76).

These changes to EasyModel would enhance its versatility and power, improving usability for both novice and expert users in the field of systems biology. Importantly, these enhancements will adhere to the user-friendly philosophy characteristic of EasyModel, ensuring that both new and experienced users find the tool beneficial and accessible.

Bibliography

[1] Wolfram Research Inc. Mathematica, 2024. URL https://www.wolfram.com/mathematica/.

[2] Stephen Wolfram. *The Mathematica Book*. Wolfram Media/Cambridge University Press, Cambridge, 1996.

[3] Rui Alves, Fernando Antunes, and Armindo Salvador. Tools for kinetic modeling of biochemical networks. *Nature Biotechnology*, 24:667–672, 6 2006. ISSN 1087-0156. doi: 10.1038/nbt0606-667. URL http://www.nature.com/articles/nbt0606-667.

[4] Herbert M Sauro and Frank T Bergmann. *Software Tools for Systems Biology*, pages 289–314. Elsevier Inc., 2010. ISBN 9780123725509. doi: 10.1016/B978-0-12-372550-9.00012-2.

[5] Maplesoft. Maplesoft - software for mathematics, online learning, engineering, 2024. URL https://www.maplesoft.com/.

[6] Martin Peters, Johann J Eicher, David D van Niekerk, Dagmar Waltemath, and Jacky L Snoep. The jws online simulation database. *Bioinformatics*, 33:btw831, 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btw831. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btw831.

[7] Tomáš Helikar, Bryan Kowal, Sean McClenathan, Mitchell Bruckner, Thaine Rowley, Alex Madrahimov, and et al. The cell collective: toward an open and collaborative approach to systems biology. *BMC systems biology*, 6:96, 2012. ISSN 1752-0509. doi: 10.1186/1752-0509-6-96. URL http://www.ncbi.nlm.nih.gov/pubmed/22871178http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3443426.

[8] David Benque, Sam Bourton, Caitlin Cockerton, Byron Cook, Jasmin Fisher, Samin Ishtiaq, and et al. Bio model analyzer: Visual tool for modeling and analysis of biological networks. *LNCS*, 7358:686–692, 2012. doi: 10.1007/978-3-642-31424-7_50. URL http://link.springer.com/10.1007/978-3-642-31424-7%5C_50.

[9] Nestor V Torres and Guido Santos. The (mathematical) modeling process in biosciences. *Frontiers in Genetics*, 6, 2015. ISSN 1664-8021. doi: 10.3389/fgene.2015.00354. URL https://www.frontiersin.org/articles/10.3389/fgene.2015.00354.

[10] Trey Ideker and Nevan J Krogan. Differential network biology. *Molecular systems biology*, 8:565, 1 2012. ISSN 1744-4292 (Electronic). doi: 10.1038/msb.2011.99.

[11] Hiroaki Kitano. Systems biology: a brief overview. *Science (New York, N.Y.)*, 295:1662–1664, 3 2002. ISSN 1095-9203 (Electronic). doi: 10.1126/science.1069492.

[12] Eric E Schadt and Johan L M Björkegren. New: network-enabled wisdom in biology, medicine, and health care. *Science translational medicine*, 4:115rv1, 1 2012. ISSN 1946-6242 (Electronic). doi: 10.1126/scitranslmed.3002132.

[13] Bernhard Palsson. *Systems Biology: Properties of Reconstructed Networks*. 2006. ISBN 978-0-521-85903-4.

[14] Rui-Sheng Wang, Assieh Saadatpour, and Réka Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical biology*, 9:55001, 10 2012. ISSN 1478-3975 (Electronic). doi: 10.1088/1478-3975/9/5/055001.

[15] H M Berman, T N Bhat, P E Bourne, Z Feng, G Gilliland, H Weissig, and J Westbrook. The protein data bank and the challenge of structural genomics. *Nature structural biology*, 7 Suppl:957–959, 11 2000. ISSN 1072-8368 (Print). doi: 10.1038/80734.

[16] Andriy Kryshtafovych, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. Critical assessment of methods of protein structure prediction (casp)-round xiii. *Proteins*, 87:1011–1020, 12 2019. ISSN 1097-0134 (Electronic). doi: 10.1002/prot.25823.

[17] Joël Janin, Kim Henrick, John Moult, Lynn Ten Eyck, Michael J E Sternberg, Sandor Vajda, Ilya Vakser, and Shoshana J Wodak. Capri: a critical assessment of predicted interactions. *Proteins*, 52:2–9, 7 2003. ISSN 1097-0134 (Electronic). doi: 10.1002/prot.10381.

[18] Florian Kiefer, Konstantin Arnold, Michael Künzli, Lorenza Bordoli, and Torsten Schwede. The swiss-model repository and associated resources. *Nucleic acids research*, 37:D387–92, 1 2009. ISSN 1362-4962 (Electronic). doi: 10.1093/nar/gkn750.

[19] Lorenza Bordoli, Florian Kiefer, Konstantin Arnold, Pascal Benkert, James Battey, and Torsten Schwede. Protein structure homology modeling using swiss-model workspace. *Nature protocols*, 4:1–13, 2009. ISSN 1750-2799 (Electronic). doi: 10.1038/nprot.2008.197.

[20] Konstantin Arnold, Lorenza Bordoli, Jürgen Kopp, and Torsten Schwede. The swiss-model workspace: a web-based environment for protein structure homology modelling. *Bioinformatics (Oxford, England)*, 22:195–201, 1 2006. ISSN 1367-4803 (Print). doi: 10.1093/bioinformatics/bti770.

[21] Manuel C Peitsch. Protein modeling by e-mail. *Bio/Technology*, 13:658–660, 1995. ISSN 1546-1696. doi: 10.1038/nbt0795-658. URL https://doi.org/10.1038/nbt0795-658.

[22] Juergen Haas, Steven Roth, Konstantin Arnold, Florian Kiefer, Tobias Schmidt, Lorenza Bordoli, and Torsten Schwede. The protein model portal–a comprehensive resource for protein structure and model information. *Database : the journal of biological databases and curation*, 2013: bat031, 2013. ISSN 1758-0463 (Electronic). doi: 10.1093/database/bat031.

[23] B B Goldman and W T Wipke. Qsd quadratic shape descriptors. 2. molecular docking using quadratic shape descriptors (qsdock). *Proteins*, 38:79–94, 1 2000. ISSN 0887-3585 (Print).

[24] Gökçe Tuncer and Vilda Purutçuoğlu. Comparative assessment of simulation tools for biochemical networks. *American Review of Mathematics and Statistics*, 3:69–82, 2015. ISSN 23742348. doi: 10.15640/arms.v3n2a9.

[25] Gökçe Tuncer. Comparison of the simulation tools for the deterministic modeling of biochemical networks. 2015.

[26] Ursula Kummer, Pedro Mendes, and Sven Sahle. Copasi, 2024. URL http://copasi.org/.

[27] Frank T. Bergmann, Stefan Hoops, Brian Klahn, Ursula Kummer, Pedro Mendes, Jürgen Pahle, and Sven Sahle. Copasi and its applications in biotechnology, 11 2017. ISSN 18734863.

[28] Lucian A. Brodsky and Peter J. G. Barker. Gepasi 3 - biochemical kinetics simulator, 2024. URL http://www.gepasi.org/.

[29] Leslie M. Loew, Ion I. Moraru, Boris Slepchenko, and James C. Schaff. Vcell- modeling and analysis software - virtual cell modeling and analysis software, 3 2024. URL http://vcell.org/.

[30] António Ferreira. Plas - enzymology group, 2022. URL http://enzymology.fc.ul.pt/software/plas/.

[31] MathWorks. Matlab, 2024. URL https://www.mathworks.com/products/matlab.html.

[32] Brett G. Olivier and Jacky L. Snoep. Web-based kinetic modelling using jws online. *Bioinformatics (Oxford, England)*, 20:2143–2144, 9 2004. ISSN 1367-4803. doi: 10.1093/BIOINFORMATICS/BTH200. URL https://pubmed.ncbi.nlm.nih.gov/15072998/.

[33] John D. Hedengren, Reza Asgharzadeh Shishavan, Kody M. Powell, and Thomas F. Edgar. Nonlinear modeling, estimation and predictive control in apmonitor. *Computers and Chemical Engineering*, 70:133–148, 11 2014. ISSN 00981354. doi: 10.1016/j.compchemeng.2014.04.013.

[34] Logan D.R. Beal, Daniel C. Hill, R. Abraham Martin, and John D. Hedengren. Gekko optimization suite. *Processes 2018, Vol. 6, Page 106*, 6:106, 7 2018. ISSN 2227-9717. doi: 10.3390/PR6080106. URL https://www.mdpi.com/2227-9717/6/8/106/htmhttps://www.mdpi.com/2227-9717/6/8/106.

[35] Wentao Ma, Xinghua Liu, Jiandong Duan, Siyuan Peng, Eric Von Lieres, Lagrande Lowell Gunnell, Kyle Manwaring, Xiaonan Lu, Jacob Reynolds, John Vienna, and John Hedengren. Machine learning with gradient-based optimization of nuclear waste vitrification with uncertainties and constraints. *Processes 2022, Vol. 10, Page 2365*, 10:2365, 11 2022. ISSN 2227-9717. doi: 10.3390/PR10112365. URL https://www.mdpi.com/2227-9717/10/11/2365/htmhttps://www.mdpi.com/2227-9717/10/11/2365.

[36] Pawan Dhar, Tan Chee Meng, Sandeep Somani, Li Ye, Anand Sairam, Mandar Chitre, Zhu Hao, and Kishore Sakharkar. Cellware - a multi-algorithmic software for computational systems biology. *Bioinformatics*, 20:1319–1321, 5 2004. ISSN 13674811. doi: 10.1093/BIOINFORMATICS/BTH067.

[37] The Eclipse Foundation. Jakarta® ee | cloud native enterprise java | java ee | the eclipse foundation, 2024. URL https://jakarta.ee/.

[38] Vaadin Ltd. Vaadin | the full-stack java web app platform, 2024. URL https://vaadin.com/.

[39] Oracle Corporation. Mysql :: Mysql community edition, 2024. URL https://www.mysql.com/products/community/.

[40] JetBrains. Intellij idea - the leading java and kotlin ide, 2024. URL https://www.jetbrains.com/idea/.

[41] Amazon. Openjdk download - corretto - aws, 2024. URL https://aws.amazon.com/corretto/.

[42] Apache Software Foundation. Maven - welcome to apache maven, 2024. URL https://maven.apache.org/.

[43] Apache Software Foundation. Apache tomcat® - welcome!, 2024. URL https://tomcat.apache.org/.

[44] leokhoa. Laragon - portable, isolated, fast and powerful universal development environment for php, node.js, python., 2024. URL https://laragon.org/.

[45] The Eclipse Foundation. Eclipse ide | the eclipse foundation, 2024. URL https://eclipseide.org/.

[46] M Hucka, A Finney, H M Sauro, H Bolouri, J C Doyle, H Kitano, and the rest of the SBML Forum. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19:524–531, 3 2003. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg015. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btg015.

[47] Sarah M Keating, Dagmar Waltemath, Matthias König, Fengkai Zhang, Andreas Dräger, Claudine Chaouiya, and et al. Sbml level 3: an extensible format for the exchange and reuse of biological models. *Molecular Systems Biology*, 16, 2020. ISSN 1744-4292. doi: 10.15252/msb.20199110. URL https://pubmed.ncbi.nlm.nih.gov/32845085/.

[48] Andreas Dräger, Nicolas Rodriguez, Marine Dumousseau, Alexander Dörr, Clemens Wrzodek, Nicolas Le Novère, and et al. Jsbml: a flexible java library for working with sbml. *Bioinformatics*, 27: 2167–2168, 2011. ISSN 1460-2059. doi: 10.1093/bioinformatics/btr361. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr361.

[49] Albert Sorribas, Benito Hernández-Bermejo, Ester Vilaprinyo, and Rui Alves. Cooperativity and saturation in biochemical networks: A saturable

formalism using taylor series approximations. *Biotechnology and Bioengineering*, 97:1259–1277, 8 2007. ISSN 00063592. doi: 10.1002/bit.21316. URL http://doi.wiley.com/10.1002/bit.21316.

[50] Uri M. Ascher and Linda R. Petzold. Computer methods for ordinary differential equations and differential-algebraic equations. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, pages 3–305, 1998. doi: 10.1137/1.9781611971392/ASSET/18908807-1118-0880-3111-908807311189/1.9781611971392.COVER.JPG.

[51] Shijie Liu. Sustainability and stability. *Bioprocess Engineering*, pages 871–947, 2017. doi: 10.1016/B978-0-444-63783-3.00015-0.

[52] R. Alves and M. A. Savageau. Systemic properties of ensembles of metabolic networks: application of graphical and statistical methods to simple unbranched pathways. *Bioinformatics*, 16:534–547, 6 2000. ISSN 1367-4803. doi: 10.1093/bioinformatics/16.6.534. URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/16.6.534.

[53] Timo R. Maarleveld, Brett G. Olivier, and Frank J. Bruggeman. Stochpy: A comprehensive, user-friendly tool for simulating stochastic biological processes. *PLoS ONE*, 8, 11 2013. ISSN 19326203. doi: 10.1371/journal.pone.0079345.

[54] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976. ISSN 10902716. doi: 10.1016/0021-9991(76)90041-3.

[55] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. In *Journal of Physical Chemistry*, volume 81, pages 2340–2361. American Chemical Society, 1977. doi: 10.1021/j100540a008. URL https://pubs.acs.org/doi/abs/10.1021/j100540a008.

[56] Johan Paulsson. Summing up the noise in gene networks, 2004. ISSN 00280836.

[57] Pascal Ballet, Jérémy Rivière, Alain Pothet, Michaël Theron, Karine Pichavant, Frank Abautret, and et al. *Modelling and simulating complex systems in biology: Introducing NetBioDyn - a pedagogical and intuitive agent-based software*, pages 128–158. IGI Global, 2016. ISBN 9781522517573. doi: 10.4018/978-1-5225-1756-6.ch006.

[58] D T Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115:1716–1733, 2001. ISSN 00219606. doi: 10.1063/1.1378322. URL http://aip.scitation.org/doi/10.1063/1.1378322.

[59] Daniel T Gillespie and Linda R Petzold. Improved lead-size selection for accelerated stochastic simulation. *Journal of Chemical Physics*, 119: 8229–8234, 2003. ISSN 00219606. doi: 10.1063/1.1613254. URL http://aip.scitation.org/doi/10.1063/1.1613254.

[60] Muruhan Rathinam, Linda R Petzold, Yang Cao, and Daniel T Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *Journal of Chemical Physics*, 119:12784–12794, 2003. ISSN 00219606. doi: 10.1063/1.1627296. URL http://aip.scitation.org/doi/10.1063/1.1627296.

[61] Tianhai Tian and Kevin Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *Journal of Chemical Physics*, 121:10356–10364, 2004. ISSN 00219606. doi: 10.1063/1.1810475. URL http://aip.scitation.org/doi/10.1063/1.1810475.

[62] Abhijit Chatterjee, Dionisios G Vlachos, and Markos A Katsoulakis. Binomial distribution based tau-leap accelerated stochastic simulation. *Journal of Chemical Physics*, 122:24112, 2005. ISSN 00219606. doi: 10.1063/1.1833357. URL http://aip.scitation.org/doi/10.1063/1.1833357.

[63] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Avoiding negative populations in explicit poisson tau-leaping. *Journal of Chemical Physics*, 123:54104, 2005. ISSN 00219606. doi: 10.1063/1.1992473. URL http://aip.scitation.org/doi/10.1063/1.1992473.

[64] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Efficient step size selection for the tau-leaping simulation method. *Journal of Chemical Physics*, 124:44109, 2006. ISSN 00219606. doi: 10.1063/1.2159468. URL http://aip.scitation.org/doi/10.1063/1.2159468.

[65] European Bioinformatics Institute. Biomodels, 2024. URL https://www.ebi.ac.uk/biomodels/.

[66] Nicolas Le Novère, Benjamin Bornstein, Alexander Broicher, Mélanie Courtot, Marco Donizelli, Harish Dharuri, Lu Li, Herbert Sauro, Maria Schilstra, Bruce Shapiro, Jacky L Snoep, and Michael Hucka. Biomodels

database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34:D689—-D691, 2006.

[67] Mihai Glont, Tung V.N. Nguyen, Martin Graesslin, Robert Hälke, Raza Ali, Jochen Schramm, Sarala M. Wimalaratne, Varun B. Kothamachu, Nicolas Rodriguez, MacIej J. Swat, Jurgen Eils, Roland Eils, Camille Laibe, Rahuman S. Malik-Sheriff, Vijayalakshmi Chelliah, Nicolas Le Novère, and Henning Hermjakob. Biomodels: expanding horizons to include more modelling approaches and formats. *Nucleic Acids Research*, 46:D1248–D1253, 1 2018. ISSN 0305-1048. doi: 10.1093/NAR/GKX1023. URL https://dx.doi.org/10.1093/nar/gkx1023.

[68] Rahuman S. Malik-Sheriff, Mihai Glont, Tung V.N. Nguyen, Krishna Tiwari, Matthew G. Roberts, Ashley Xavier, Manh T. Vu, Jinghao Men, Matthieu Maire, Sarubini Kananathan, Emma L. Fairbanks, Johannes P. Meyer, Chinmay Arankalle, Thawfeek M. Varusai, Vincent Knight-Schrijver, Lu Li, Corina Dueñas-Roca, Gaurhari Dass, Sarah M. Keating, Young M. Park, Nicola Buso, Nicolas Rodriguez, Michael Hucka, and Henning Hermjakob. Biomodels-15 years of sharing computational models in life science. *Nucleic Acids Research*, 48:D407–D415, 1 2020. ISSN 0305-1048. doi: 10.1093/NAR/GKZ1055. URL https://dx.doi.org/10.1093/nar/gkz1055.

[69] Jordi Bartolome, Rui Alves, Francesc Solsona, and Ivan Teixido. Easymodel: user-friendly tool for building and analysis of simple mathematical models in systems biology. *Bioinformatics*, 36:976–977, 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz659. URL https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/btz659/5554697.

[70] Jordi Bartolome, Rui Alves, and Francesc Solsona. Easymodel 1.1: User-friendly stochastic and deterministic simulations for systems biology models. In *BIOINFORMATICS 2020 - 11th International Conference on Bioinformatics Models, Methods and Algorithms, Proceedings; Part of 13th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2020*, volume 3, pages 145–149, 2020. ISBN 9789897583988. doi: 10.5220/0008966001450149.

[71] Jose M G Vilar, Ronald Jansen, and Chris Sander. Signal processing in the tgf-beta superfamily ligand-receptor network. *PLoS Computational Biology*, 2:36–45, 2006. ISSN 1553734X. doi: 10.1371/journal.pcbi.0020003. URL https://pubmed.ncbi.nlm.nih.gov/16446785/.

[72] Svetlana V Komarova, Robert J Smith, S Jeffrey Dixon, Stephen M Sims, and Lindi M Wahl. Mathematical model predicts a critical role for osteoclast autocrine regulation in the control of bone remodeling. *Bone*, 33: 206–215, 2003. ISSN 87563282. doi: 10.1016/S8756-3282(03)00157-1.

[73] Steven H. Strogatz. *NONLINEAR DYNAMICS AND CHAOS: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 1 2018. ISBN 9780429961113. doi: 10.1201/9780429492563/NONLINEAR-DYNAMICS-CHAOS-STEVEN-STROGATZ/ACCESSIBILITY-INFORMATION. URL https://www.taylorfrancis.com/books/mono/10.1201/9780429492563/nonlinear-dynamics-chaos-steven-strogatz.

[74] M. Hazewinkel, R.Jurkovich, and J. H. P. Paelinck. Bifurcation analysis. *Bifurcation Analysis*, 1985. doi: 10.1007/978-94-009-6239-2.

[75] Martin Feinberg. Foundations of chemical reaction network theory. *Applied Mathematical Sciences (Switzerland)*, 202:i–454, 2019. ISSN 2196968X. doi: 10.1007/978-3-030-03858-8/COVER.

[76] Elf Lab. Elf lab | elfware, 2024. URL https://elflab.icm.uu.se/craft/elfware/.

The important thing is not to stop questioning. Curiosity has its own reason for existing.

(Albert Einstein)