GRN_modeler: An Intuitive Tool for Constructing and Evaluating Gene Regulatory Networks and its Applications to Oscillators and a Light Biosensor

Gábor Holló[1*], Jung Hun Park[1], Emanuele Boni[1], Yolanda Schaerli[1*]

[1]Department of Fundamental Microbiology, University of Lausanne, Biophore Building, 1015, Lausanne, Switzerland

Abstract

Modeling and simulating gene regulatory networks (GRNs) is crucial for understanding biological processes, predicting system behavior, guiding the design of synthetic biological systems, and interpreting experimental data. In synthetic biology, GRNs play a pivotal role in enabling the design and control of complex systems for a wide range of applications. However, GRN simulations can be time-consuming and often require specialized expertise. To make this process more accessible, we developed a user-friendly application with a graphical user interface (GUI), allowing users to create simple phenomenological models without requiring prior programming experience. We demonstrate the versatility of our tool through several examples, including the design of novel oscillator families capable of robust oscillation with an even number of nodes. These complement the well-known repressilator family, which only oscillates with odd-numbered nodes. Furthermore, we showcase how GRN_modeler allowed us to develop a light-detecting biosensor in *Escherichia coli* that can track light intensity over several days, leaving a record in the form of ring patterns in bacterial colonies. In summary, our work empowers biologists to model their systems of interest even without programming expertise.

Introduction

Gene Regulatory Networks (GRNs) are intricate systems that regulate gene expression and ultimately determine cellular function and behavior [1, 2]. They consist of molecular regulators, including transcription factors, that interact with specific DNA binding sites to control levels of mRNA and proteins. The multidisciplinary field of synthetic biology applies engineering concepts to create novel biological components, devices, and systems, or to re-engineer existing natural systems. Designing and testing synthetic GRNs is crucial in this process [3–5]. By carefully characterizing individual genetic components and integrating computational modeling into the design and engineering process, the field seeks to move beyond repeated cycles of trial and error. Therefore, as synthetic systems grow increasingly complex, computational modeling becomes vital for accurately predicting their behaviors. However, the expertise required in mathematics and programming presents a significant challenge for scientists with a training focused on experimental work.

*Correspondence E-mail: gabor.hollo@unil.ch, yolanda.schaerli@unil.ch

Modeling GRNs requires reaction kinetics simulations, which involve solving and analyzing ordinary differential equations (ODEs). There are many software options to solve and analyze ODEs. For example, the popular program XPP/XPPAUT [6] can be used for dynamic system analysis. To facilitate the creation of ODEs based on the stoichiometry and rate laws of reactions, specialized reaction kinetics softwares such as KinTek Explorer [7–9] are also available.

In the early 2000s, the Systems Biology Markup Language (SBML) [10, 11] was developed to standardize the representation of biological process models. SBML is a computer-readable format that includes information on species, reactions, parameters, rules, and constraints. Its purpose is to enable researchers to work with the same model across different software environments. As a result, several widely-used programs for modeling biochemical reaction networks support SBML, including TABASCO [12], COPASI [13], Bioscrape [14], libRoadRunner [15, 16], SBMLsimulator [17], and the MATLAB SimBiology toolbox.

Another set of tools facilitate the design of synthetic systems via a model-based design strategy. For example, txtools focuses on RNA structure analysis [18]. iBioSim (integrated Biological Systems Simulator) [19, 20] provides an intuitive graphical interface that simplifies the construction and visualization of genetic circuits using fundamental components such as promoters, ribosome binding sites, and terminators. TX-TLsim (Transcription-Translation Simulator) [21] is a computational tool designed to model and simulate the transcription and translation processes in cell-free systems, such as *Escherichia coli* extracts. It offers detailed simulations of these processes, enabling researchers to predict the behavior of *in vitro* genetic constructs with high accuracy. On the other hand, Metabolic Tinker is used for designing metabolic pathways [22], and WebCM supports individual-based simulations of multicellular microbial populations [23]. Finally, Flapjack [24] concentrates on the whole design, build, test, learn cycle (DBTL).

However, when describing the qualitative behavior of complex systems with many nodes (a regulatory unit composed of genetic elements such as a promoter, a gene and and terminator) or designing circuits for specific functionalities, these tools can still be limiting. In such cases, it is more efficient to design circuits using individual nodes as fundamental building blocks, avoiding the need to repeatedly add the same reactions and species for each node in the model-building process. This higher level of abstraction provides a simplified overview of the system, enhancing the feasibility of constructing complex networks. The genetic design automation tool LOICA (Logical Operators for Integrated Cell Algorithms) uses a high-level design abstraction by representing networks as combinations of components, rather than as combination of individual parts [25]. However, it only offers a command-line interface, limiting its accessibility to users who already possess modeling expertise.

We have developed a user-friendly software with a graphical user interface (GUI) specifically designed for biologists who may not have advanced computational skills. Our application is an intuitive and valuable tool for synthetic biology, aimed to support the entire process of designing, building, and testing synthetic GRNs. It enables users to model GRNs, predict and compare their behavior with ease. The tool simplifies the creation of novel circuits, facilitates rapid hypothesis testing, and guides the design process by highlighting where strong or weak interactions are needed to construct robust systems that are less sensitive to minor parameter changes. Additionally, it can be used to make predictions for potential experiments and genetic modifications, helping researchers anticipate results. For circuits that do not perform as expected, the application assists in troubleshooting and diagnosing issues, thereby enhancing problem-solving and optimization efforts. In summary, by simplifying the modeling process, our tool empowers researchers to design and engineer GRNs more effectively, thus accelerating progress in synthetic biology.

We demonstrate the main capabilities of the application and illustrate their use with examples. We start by modelling well-established circuits from the literature, focusing on oscillators and feed-forward loops. Oscillators have been a subject of great interest for synthetic biologists since the early days of the field [26]. Our analysis includes several oscillators: the

repressilator, the first synthetic oscillator described in 2000 [27], and an extension of it that facilitates tuning of amplitude and period [28]. We also present a model of the CRISPRlator – a synthetic oscillator with the repressilator topology, but leveraging CRISPR interference (CRISPRi) instead of protein transcription factors to inhibit transcription [29]. We also include feedforward loops (FFLs), fundamental network motifs composed of three interacting genes and commonly found in most living organisms [30–32]. These motifs can function as pulse generators, accelerate response times or act as band-pass concentration sensors [30–32] and have frequently been a focus of synthetic biology research [33–36]. After showcasing the capabilities of our application with known systems, we illustrate how it can be used to design novel circuits. As an example, we create new, robust oscillators that can oscillate with an even number of nodes. Finally, we present an example of the interplay with experimental work. We build a light-recording biosensors in *E. coli* and demonstrate how GRN_modeler predicts phase and antiphase output signals in response to periodic external light exposure, such as day-night light cycles. We then validate these predictions through experimental testing, showcasing the tool's effectiveness in guiding biological circuit behavior.

Results

Basic properties of the application

Our application is based on MATLAB SimBiology, a powerful toolbox within the MATLAB environment designed for modeling, simulating, and analyzing biological systems. Its integration with MATLAB provides significant advantages in terms of computational capabilities, data visualization tools, and an extensive library of functions for data analysis. Furthermore, SimBiology supports the SBML format, which allows users to build models efficiently within our software and then perform further analysis using any other SBML-compatible software. However, one limitation of SimBiology is that its stochastic solver is restricted to mass action kinetics, where the reaction rate is proportional to the product of the reactant concentrations. This constraint prevents the modeling of complex reactions, such as enzymatic reactions with Michaelis-Menten kinetics, or inhibitions, and activations, which are typically described using Hill functions. To address this, we have integrated our application with COPASI [13] through BasiCO (Basic COmpartmental model Interface) [37], a streamlined Python interface for COPASI. This integration enables access to a broader range of solvers, including stochastic ones, within our MATLAB application, leveraging COPASI's extensive solver capabilities. Most of the functionalities in our application become available after installing SimBiology. To access the COPASI solvers, it is also necessary to install COPASI [13], Python, and BasiCO [37].

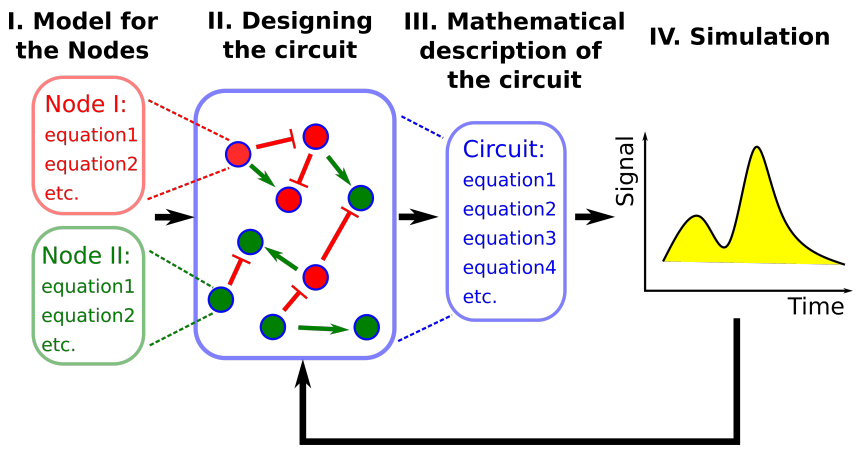

Figure 1: Flow chart of the application modeling steps. First, a model for each node must be chosen (a selection of existing models is made available in the app, or a new model can be implemented). Then, using the graphical user interface, users can assemble a circuit. Next, the application automatically generates the corresponding differential equations for the entire system, allowing simulations to be run and to observe the dynamic behavior of the GRN. If the circuit does not produce the expected results, users can redesign it. Once experimental data is available, the parameters can be fine-tuned to achieve a closer quantitative match with the simulations.

The key concepts and the level of abstraction that demonstrate how the GRN_modeler operates are summarized in Figure 1.

I. First, we define a model for each node, what we call "node model" to distinguish from the mathematical description or the model of the whole circuit, what we will call "circuit model" or simply "model". The user can select an implemented node model, or create a new one. New node models can be included in the `models` folder and these node models can be selected when starting the application. Our simulation functions at the node level, where nodes are here defined as the fundamental components of the biological network. Each node represents crucial elements such as reactions and concentrations of species (e.g., RNAs and proteins) produced through transcription and translation from the DNA coding sequence. In the node model, nodes can have varying input and output functions, allowing users to define interactions between them. Kinetic parameters can be either "individual" or "common". For individual parameters, a separate parameter is assigned to each node, named accordingly. For "common" parameters, all nodes use the same value.

II. Second, once a model for a node is established, the system can be constructed by adding nodes, defining interactions between them, and incorporating external regulators or protein degradation by proteases.

III. Third, the application automatically generates the corresponding differential and algebraic equations for the entire system, based on the model for the nodes and their interactions. Once the mathematical model for the circuit is constructed, we can further refine it by applying various "rules" to fine-tune the system's behavior. In SimBiology, "rules" define mathematical relationships between model components, such as differential equations, algebraic expressions, or rate laws, to govern the dynamics of biological systems. For example, inducer concentrations can be temporally modified by these user-defined rules. We will demonstrate this property in the section "Experimental example: a light sensor" by modulating the parameters sinusoidally in response to external light exposure.

IV. Finally, we can run simulations to observe the system's dynamic behavior. If the circuit does not show the expected behavior, we can return to step II to redesign the circuit and repeat the process.

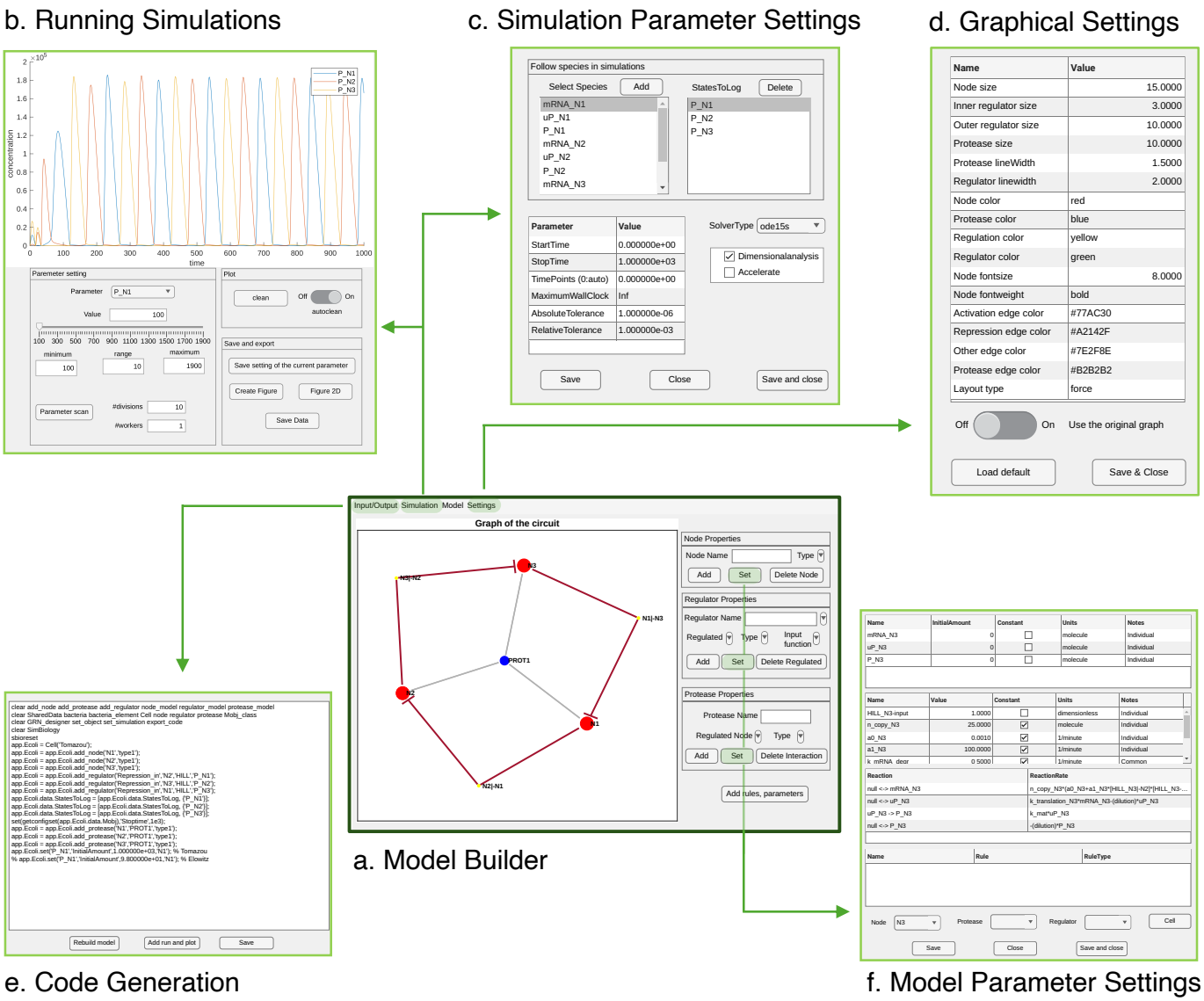

Figure 2: Key components of the graphical user interface: a) Model Builder for designing models, b) Running Simulations for executing and managing simulations, c) Simulation Parameter Settings for configuring simulation parameters, d) Graphical Settings for customizing the visual presentation, e) Code Generation for converting models into executable code, and f) Model Parameter Settings for adjusting model-specific parameters.

We designed a (GUI) (Figure 2) to enable users to create the model for the circuit, set properties, and run simulations without requiring any programming skill. It provides access to all the functions, while allowing users to select objects and set their parameters in a intuitive way. The constructed circuit can be visualized as a directed graph with multiple layout options, such as organizing nodes in a circular or layered structure. In the "Graphical Settings" tab (Figure 2d), users can

customize the appearance by adjusting colors, node sizes, and other visual aspects. Users can also compare deterministic and stochastic simulation results by overlaying their trajectories and adjusting specific parameters. Additionally, the tool allows users to scan a parameter across a defined range and observe its impact on the trajectories at a specific time point. For users that prefer not to use the GUI, all functionalities of GRN_modeler can also be achieved through the command line and the basic functions are presented in the Command Line Functionality section in the SI (Section 2.1). While the GUI provides many useful properties and functionalities, MATLAB offers additional powerful tools for model analysis. For this reason, once the user has built the model for the circuit in the GUI, the code generation feature allows to produce a MATLAB script that replicates all actions performed in the GUI. This approach enables users to work more conveniently without needing to know the specific species or parameter names or the syntax of the different functions. Finally, it is also possible to open the SimBiology Model Builder application or COPASI directly from the GUI for further analysis or parameter fitting.

Building GRNs and the implemented node models

To demonstrate the capabilities of our tool in simulating GRNs, we have implemented several node models. In the following sections, we illustrate how to build and reproduce the behavior of well-known oscillators and feed-forward loops. These examples can serve as templates for creating new node models or adapting existing ones from the literature. All examples presented in the manuscript can be found in the `analysing_examples` folder of the code.

The Elowitz node model

First, we implemented a node model for the repressilator. The repressilator, one of the first synthetic genetic circuits, was published by Elowitz and Leibler in 2000 [27]. This circuit consists of a network of three transcription factors (TFs) that sequentially repress each other's expression in a cyclic manner, generating oscillatory behavior (Figure 3Ia). The repressilator showed robust oscillations in *E. coli*, demonstrating the feasibility of engineering periodic gene expression in living cells. The node model describing the repressilator (referred to in this manuscript as the "Elowitz model") only includes mRNAs and transcription factors, making it an ideal choice to describe GRNs based on transcription factors. More details on the Elowitz model are provided in the Supplementary Information in Table S2, as well as a detailed description of the reaction kinetics simulations generated by SimBiology. In Figure 3I, we present the results of both deterministic and stochastic simulations. For the stochastic simulations in this example, we utilized the "adaptivesa" solver from COPASI. Notably, the fundamental model describing the nodes (e.g., RNA, protein concentrations and the repression) is not specific to the repressilator and can be applied to simulate systems with different topologies. This generalization allows the nodes to be used as building blocks for designing diverse circuits with varying functions and purposes.

The re-designed repressilator node model to provide independent control of amplitude and period

A more refined node model can be applied to the same genetic circuit, increasing the level of detail of the simulation. Building upon the Elowitz model, Tomazou *et al.* [28] developed a more complete transcription factor node model, which includes the processes of protein folding and protein degradation by proteases. We show the details of the Tomazou model in Table S3. In this node model, transcription (R_1) produces an (mRNA) which is translated (R_2), yielding first an unfolded protein (uP) which finally becomes, through maturation (R_3), a folded protein (P). Additionally, the proteins are degraded by proteases following Michaelis-Menten kinetics. Also, the concentrations of the mRNA, folded and unfolded

protein decrease with a dilution rate (k_d). R_4 reaction describes the elimination of the folded protein due to dilution by cell growth and degradation by proteases. In particular, Tomazou *et al.* [28] used this node model to re-design the repressilator in order to provide independent control of amplitude and period.

In the repressilator, amplitude and frequency are interdependent — typically, as frequency increases, amplitude decreases. However, for practical applications, it would be desirable to adjust these two parameters independently. To achieve this, Tomazou *et al.* [28] and Fengyu *et al.* [38] suggested incorporating an additional output node (N4) into the repressilator topology. N4 is repressed by one of the three oscillatory nodes (labelled N3 in Figure 3II). The expression of node 2 (N2) can be modulated with an inducer (via Y and I_2). In the simple repressilator, an increased expression of N2 would not only lead to an increased frequency, but also to a lower amplitude. However, in the re-designed topology, a small amount of N3 is sufficient to repress the output node N4. Therefore, increasing the frequency with an inducer "I_2" does not significantly affect the amplitude of the output in N4. Similarly, by modulating the concentration of another inducer, "I_1", it is possible to modulate the expression strength of N4 and change the amplitude in N4 oscillations. If N4 and the other nodes are degraded by the same protease, which can result in a protease "queuing effect", the change in amplitude affects the oscillator's time period. Conversely, if N4 is not coupled to the other nodes, for example if they are targeted to different proteases (C and L) (Figure 3II and Table S3), the amplitude can be modulated independently from the frequency. In Figure 3II, we replicate these simulations, demonstrating the independent modulation of amplitude and frequency. In summary, in this section we illustrated how to use different node models, add proteases, and build more complex circuits. Using the example of the implemented node models, one can easily integrate new node models. Even with the increased complexity of the reaction system, building the circuit model remained simple and can be completed within minutes using our app.

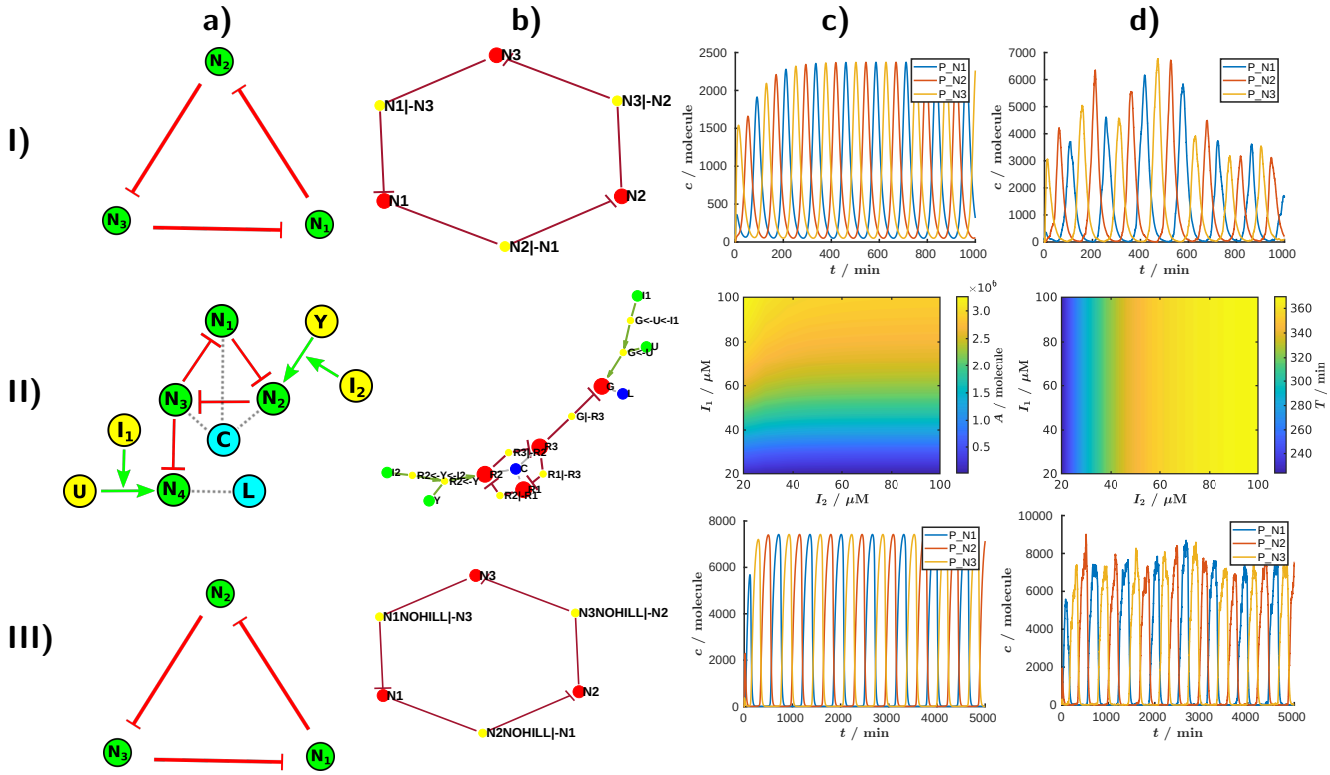

Figure 3: Applications of three different node models implemented in GRN_modeler. I) The repressilator a) Topology of the repressilator. b) Topology of the circuit in the interface generated by the `make graph()` method: the red dots N_1, N_2, and N_3 represent the nodes in the circuit. The red arrows denote the directed edges of the graph, illustrating the repression interactions between nodes. Additionally, the yellow dots $N_1 \dashv N_3$, $N_2 \dashv N_1$, and $N_3 \dashv N_2$ represent the regulatory interactions. For example, $N_1 \dashv N_3$ denotes that node N_1 is repressed by N_3. c) Deterministic and d) stochastic simulations of the repressilator using the Elowitz model, as described in Table S2 in the SI. The detailed information about the model is available in the SI, specifically in the file "repressilator.html". II) Re-designed repressilator for an independent modulation of amplitude and frequency. a) Topology of the re-designed repressilator. "C" and "L" are proteases, "I_1" and "I_2" external inducers, "U" and "Y" transcription factors activating N_4 and N_2 respectively. The dotted gray lines indicate which proteins are degraded by the proteases. b) Topology of the circuit in the interface. c) The effect of the inducers on the amplitude of N_4. d) The effect of the inducers on the time period of N_4. The detailed information about the model is available in the SI, specifically in the file "Tomazou.html". III) The CRISPRlator circuit. a) Topology of the CRISPRlator. b) Topology of the circuit in the interface. For example, N_1NOHILL$\dashv N_3$ denotes that node N_1 is repressed by N_3 through an input named NOHILL, which corresponds to the CRISPRi interaction in the node model. This repression is incorporated into the system by introducing new reactions (R_6 and R_7 in Table S4), where the dCas:sgRNA$_{N3}$ complex binds to DNA$_{N1}$, inhibiting transcription at node N_1. c) Deterministic and d) stochastic simulations of the CRISPRlator generated by the model detailed in Table S4 in the SI. The detailed information about the model is available in the SI, specifically in the file "CRISPR.html". The deterministic simulations were fulfilled with the "ode15s" solver of MATLAB, while the stochastic simulations were performed using the "adaptivesa" solver of COPASI in every case.

The CRISPRlator node model

Synthetic GRNs based on protein transcription factors are robust and effective, but they have some limitations. Engineering genetic circuits with CRISPR-based gene expression control represents a promising alternative [39]. In CRISPR interference (CRISPRi) a catalytically inactive Cas protein (typically dCas9) is targeted to specific DNA sequences with the help of a single guide RNA (sgRNA) and represses transcription by blocking the RNA polymerase. Due to its RNA-guided nature, as opposed to protein-based, CRISPRi allows for rapid and straightforward design of a large number of highly orthogonal sgRNAs. CRISPR-based circuits also impose low burden on host cells (as sgRNAs are only transcribed, but not translated) [29, 39–41]. Previously, in our group we have built a synthetic oscillator based on the repressilator topology, but utilizing CRISPRi-mediated repression, termed the CRISPRlator [29]. Each node expresses a sgRNA in an operon with a fluorescent protein (mCherry, mCerulean or mCitrine). The composite mRNA is cleaved by Csy4 [42], and the resulting sgRNA binds to dCas9 to target the upstream region of the next node's promoter, while the rest of the mRNA is translated into a fluorescent reporter. The oscillation occurs analogously to the repressilator, however the interaction between the nodes are different, thus requiring a new node model to describe the dynamics.

For this, we built upon the node model published by Santos-Moreno *et al.* [43]. The modifications are detailed in SI Section 2.4. Briefly, to align with existing transcription factor node models, we adapted parameters from the Elowitz model [27] for reactions R_1, R_2, R_3 (Table S4). To match the experimentally observed time period for the CRISPRlator (approximately 10-11 hours) [29], we fine-tuned the dilution rate k_d. The revised node model is summarized in Table S4, and robust oscillations for the CRISPRlator are shown in Figure 3III. This node model provides a general framework for the CRISPRi-based gene regulation, suitable for initial exploration of the behavior of these circuits. However, further refinement of parameters is required for more accurate descriptions of specific experimental systems.

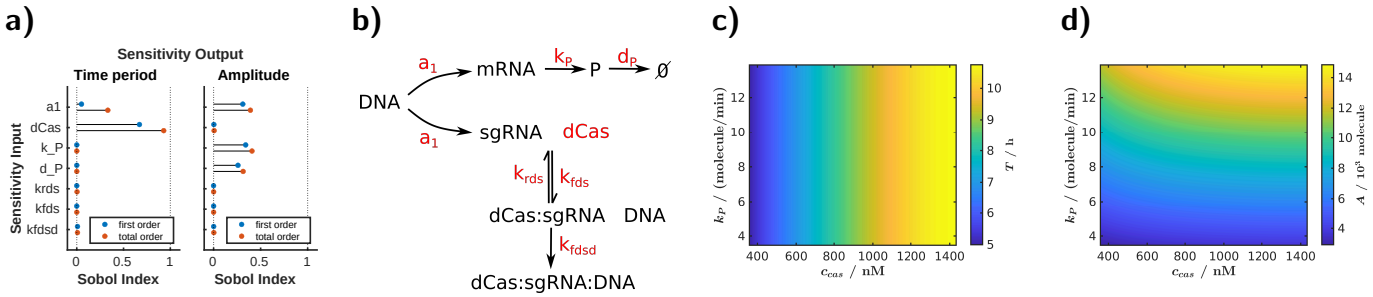

Figure 4: The CRISPRlator allows for independent frequency and amplitude modulation. a) Global sensitivity analysis was performed using MATLAB's `sbiosobol` function, where each parameter was varied by a factor of 0.5 and 2 times its original value, and the number of samples was set to 1000. b) Schematic representation of the system highlighting the examined parameters. c) Amplitude A and d) time period T of the oscillations as functions of dCas9 concentration c_{cas} and protein translation rate k_P. The detailed information about the model is available in the SI, specifically in the file "CRISPR.html".

In order to identify which parameters have the largest impact on the CRISPRlator's amplitude and time period in this model, we performed a global sensitivity analysis with the SimBiology toolbox of MATLAB. We tested the following parameters: "a_1" represents the promoter strength, "dCas" refers to dCas9 concentration, "k_P" and "d_P" are the rates for protein production and degradation, while "krds" and "kfds" correspond to the formation and degradation rates of the dCas complex, and "kfdsd" represents the formation rate of the dCas:sgRNA complex. The first-order Sobol index

quantifies the influence of each parameter on its own, while the total-order Sobol index captures the combined effect of a parameter in interaction with other parameters. We found that the oscillation period is primarily influenced by promoter strength and dCas9 concentration (Figure 4a). In contrast, the amplitude is determined largely by promoter strength and the rates of reporter protein translation and degradation. This oscillator operates at a slower pace than the repressilator, allowing protein concentrations to nearly saturate and providing sufficient time for production and degradation rates to equilibrate. Consequently, the concentration of dCas9 has minimal impact on the amplitude, as it does not directly affect these rates. Conversely, the protein kinetic parameters have little effect on the oscillation period, since the CRISPRlator mechanism does not depend on transcription factors, unlike the repressilator. This decoupling enables independent control of the oscillation frequency and amplitude by modulating the concentration of dCas9 and the translation and degradation rate of the reporter protein, respectively. Such control produces horizontal and vertical isoclines in two-dimensional parameter space (Figure 4b).

The feed-forward loop network motif to showcase parameter scanning

GRN_modeler allows easy adjustments of parameters to observe their effects on system trajectories. However, sometimes the final state of the system after a specified time is more relevant than its dynamics. For such cases, the parameter scan feature is particularly useful. This feature allows for varying a parameter over a broad range and analyzing the system's response at a specific time point. If needed, this process can be accelerated through parallelization, since these are independent simulations. This capability is integrated into the GUI.

We demonstrate the parameter scan feature using feed-forward loops (FFLs) [31, 32], a class of circuits that have attracted significant attention in SynBio [33, 44, 45]. FFLs are 3-node regulatory networks where the first node controls the third node through two different paths, directly and indirectly through a second node. In coherent FFLs, the direct and indirect regulation paths have the same sign (both activations, or both repressions), while the two signs are different in incoherent FFLs. Figure S6 shows the topologies for coherent FFLs type 1 and incoherent FFLs type 2, along with the system's response to varying R_1 inducer concentrations (using the parameters from the Elowitz model, Table S2). The model successfully predicts the system's behaviour observed by others in experimental systems . For coherent FFL 1, increasing the inducer concentration progressively activates the output node N_3 until it reaches saturation, as N_3 is activated both directly and indirectly (through N_2) by N_1. In contrast, for incoherent FFL 2 the output is activated the most at intermediate inducer concentrations, showing a peak in the response function. This behavior, known as band-pass filter or stripe-forming, has previously been implemented experimentally in SynBio [33, 44, 45].

Novel oscillatory circuits

In the previous sections, we implemented models described in the literature and demonstrated the capabilities of our application through these examples. However, our tool is also valuable for exploring the behavior of novel circuit topologies. It allows for qualitative testing of new circuit designs, assessing their robustness under varying parameters, and helps determining the feasibility of an experimental implementation. In this section, our goal is to illustrate how GNR_modeler facilitates the iterative process of designing new circuits, building models with the appropriate equations, and theoretically testing them.

The reptolator

The original repressilator has 3 nodes [27]. A version with 5 nodes has also been constructed [46]. The 4-node repressilator does not oscillate because if every second node is activated (either N1 and N3, or N2 and N4) and the other two nodes are deactivated, the system reaches a stable state [46]. More generally, the repressilator family can oscillate with an odd number of nodes and is bistable with an even number of nodes. We aimed to design a robust oscillator that can oscillate with an even number of nodes. By introducing mutual repressions (toggle switches) [47] between activated nodes in a 4-node repressilator (between N1-N3 and N2-N4), this state becomes unstable, causing the system to start oscillating again, as shown in Figure 5 a,b. Since this circuit combines elements of the repressilator and toggle switches, we will refer to it as the "reptolator". Using this configuration, we can observe oscillations in systems with any integer multiple of 4 nodes (4, 8, 12, $\ldots N$ [1]). Conversely, systems with 6, 10, $\ldots$ nodes remain bistable. Generally, if half the number of nodes is odd, this reptolator family will not oscillate. In such cases, the nodes on opposite sides of the circuit, connected by a toggle switch, can stabilize in opposite states, resulting in a bistable system, as demonstrated for the six-node circuit shown in Figure 5 a,b.

In the original repressilator, if the nodes are ordered sequentially (N1 represses N2, N2 represses N3, and so on), and the first node is activated, the sequence of activated nodes is 1, 3, 5, $\ldots$, N, 2, 4, $\ldots$, N-1[2]. In other words, every second node is activated because two consecutive repressions effectively result in an activation, causing the first node to activate the third one. In this new reptolator family the pattern of activated nodes is more complex. Let us illustrate this with the eight-node circuit (Figure 5 a,b): If N1 is activated, the next activated node is not the third one, because it is repressed by both the second and seventh nodes. Instead, the next activated node is the sixth one because the first node represses both the second and fifth nodes. In other words, the first node releases repression of the sixth node through two pathways (N1 represses N5, which represses N6, and N1 represses N2, which represses N6) and no other node inhibits the sixth node. Following this logic, the next activated node is the third node, and the system oscillates in the following sequence: N1, N6, N3, N8, N5, N2, N7, N4. Other circuits in this family follow the same activation rules[3].

[1] $N = 4n, n \in \mathbb{Z}^+$ and N is the number of nodes

[2] $n_j = n_{i+2}$, where $i, j \in \mathbb{Z}_N^+$ and n_i, n_j are the actual and next node while N is the number of the nodes, while $\mathbb{Z}_N^+$ describe the set {1, 2, $\ldots$, N} under modulo N arithmetic

[3] $n_j = n_{i+N/2+1}$, where $i, j \in \mathbb{Z}_N^+$ and n_i, n_j are the next and previous node while N is the number of the nodes.

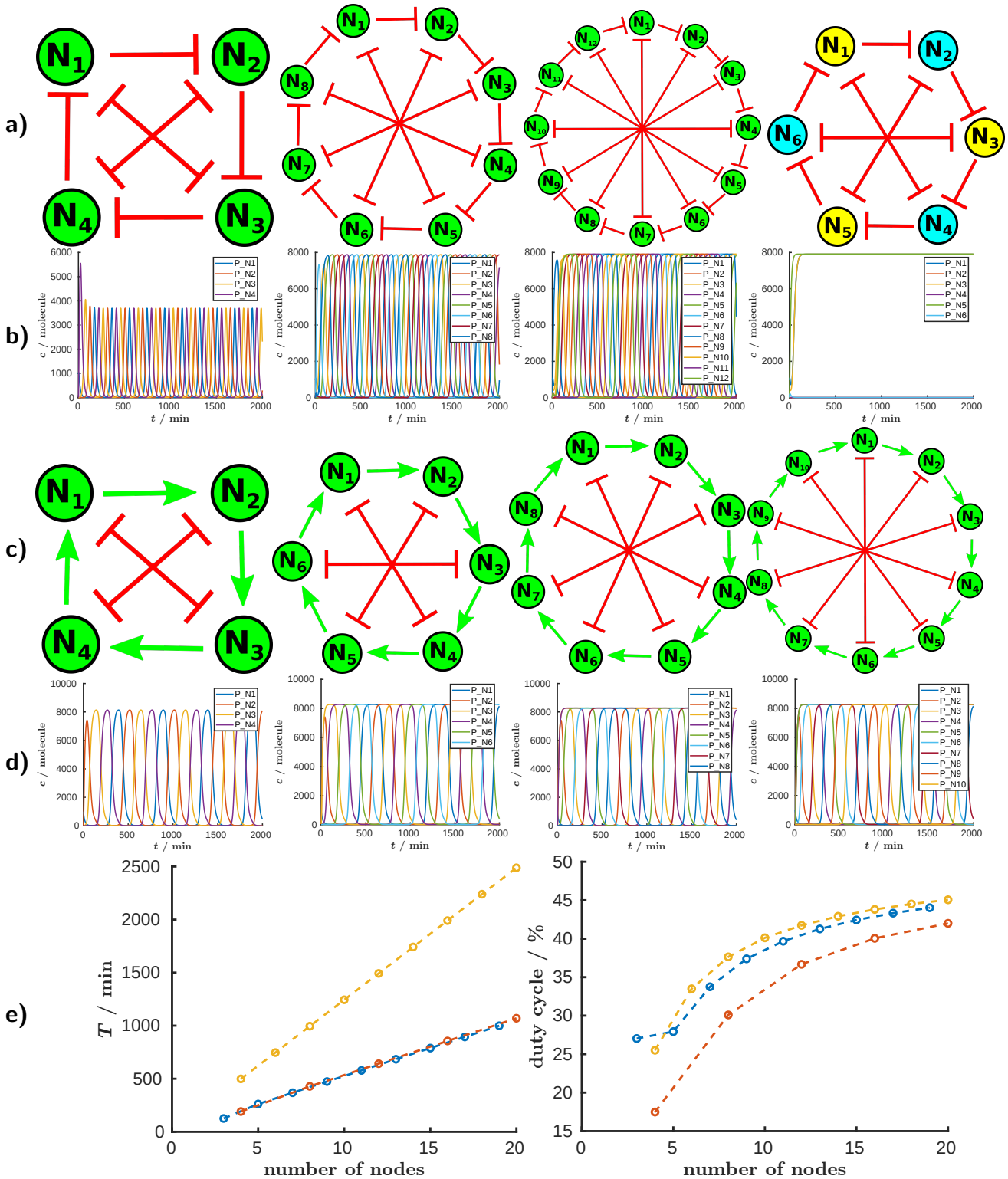
a)
b)
c)
d)
e)
c / molecule
t / min
P_N1
P_N2
P_N3
P_N4
P_N5
P_N6
P_N7
P_N8
P_N9
P_N10
P_N11
P_N12
T / min
duty cycle / %
number of nodes

Figure 5: New oscillatory circuits with even number of nodes. a-b) The reptolator. a) Topologies of gene regulatory networks with 4, 8, 12, and 6 nodes, respectively, which were created by incorporating toggle switches into the original repressilator circuit. b) Protein concentration for each node of the circuits in a) over time. While the circuits with 4, 8, and 12 nodes exhibit oscillatory behavior, the 6-node circuit remains bistable. The detailed information about the models are available in the SI, specifically in the files "N4.html", "N8.html", "N12.html" and "N6.html", respectively. c-d) The actolator. We designed the actolator circuit by replacing the repressions in the original repressilator with activations and introducing toggle switches between opposite nodes. c) Topologies of gene regulatory networks with 4, 6, 8 and 10 nodes, respectively. d) Protein concentration for each node of the circuits in a) over time. e) Comparison of circuit properties, specifically the oscillation period (T) and duty cycle (left and right, respectively), as a function of the number of nodes. The compared circuit families are: the repressilator (blue), the reptolator (red), and the actolator (yellow). The detailed information about the models are available in the SI, specifically in the files, "actolator4.html", "actolator6.html", "actolator8.html" and "actolator10.html", respectively. The deterministic simulations were performed using the Elowitz model [27].

The actolator

The repressilator can oscillate with an odd number of nodes, while the reptolator can oscillate with an integer multiple of 4 nodes. However, these circuits cannot oscillate with 6, 10, 14, ... nodes[4]. To create an oscillator capable of operating with 6 or 10 nodes, we designed the "actolator." This circuit is similar to the reptolator, except that the repression loop is replaced with an activation loop (Figure 5 c,d). Compared to repression, where a node is active in the absence of repression, in activation, a node is not expressed until activation occurs. Unlike the reptolator, the 6-node actolator is not bistable, and this circuit oscillates for every even number of nodes. If we consider the state in which N1 levels are high and N6 levels are low, it means N2 is being activated. At the same time, N1 is not activated by any node, so it slowly gets depleted. Once N1 levels are low, N2 is no longer activated, and this instability makes the system move forward. Another difference compared to the repressilator or reptolator families is the order of activated nodes. Since the circuit is based on an activation loop, the activation of nodes follows the sequence of N1, N2, N3, ... [5]. It is worth noting that there is an alternative ladder-like representation of the actolator (Figure S8 c,d), which highlights the role of the toggle switches in the circuit more than the activation loop. In the 4-node actolator, if node N1 is active, it will in turn activate node N2. Note that N1 and N2 are on two separate toggle switches, and if N2 is active it will repress node N4, preventing it from activating the first node, so the first toggle switch will turn off. In the next iteration, the first toggle switch would turn on again, this time with high level of N3. In other words, the two toggle switches periodically destabilize each other with a delay (due to interactions), causing them to switch one after another.

In Figure 5e we compare the oscillation period and duty cycle (the proportion of time during which the system is active) across three oscillator families: the repressilator, the reptolator, and the actolator. For the duty cycle calculation, a node is considered active when its protein concentration exceeds the midpoint between its minimum and maximum values during one oscillation period. The data show that as the number of nodes increases, the duty cycle approaches ≈50%. As a qualitative example, we can consider the repressilator, where each node is repressed by only one other node. If the average duty cycle of any given node were to fall below 50%, the node it regulates would subsequently exhibit a higher duty cycle. Due to the symmetrical nature of the circuit, this ultimately results in an average duty cycle of ≈50%

[4] $N = 4n + 2$, where $n \in \mathbb{Z}^+$ and N is the number of the nodes.

[5] $n_j = n_{i+1}$, where $i, j \in \mathbb{Z}_N^+$ and n_i, n_j are the actual and next node while N is the number of the nodes.

for all nodes. However, when the number of nodes is lower, the protein concentration either does not reach saturation or takes longer to do so, resulting in a decreased duty cycle. This increase in duty cycle with increasing nodes, combined with a longer oscillation period, suggests that larger circuits could impose a significant metabolic burden on cells. If we aim to design larger oscillators in the future, it would be beneficial to develop GRNs where the duty cycle decreases as the number of nodes increases. Another noteworthy model prediction is that, for the same number of nodes, the actolator oscillates at a slower rate than the reptolator and repressilator. This difference arises because, in the longest (or rate-determining) pathway between two consecutively activated nodes, the repressilator and reptolator each require the deactivation of one intermediate node, whereas the actolator requires the consecutive deactivation of two nodes[6]. As a result, the predicted oscillation periods for the repressilator and reptolator are similar, while the actolator oscillates with roughly twice the period. In Figure S7, we present the special case of the 2-node actolator.

Modified repressilator circuits: the "acrelator" family

As discussed previously, the repressilator with an even number of nodes is typically bistable. Instead of adding toggle switches, another way to destabilize such a system and induce oscillations is by replacing an odd number of repressive interactions with activations. We named this circuit, which integrates both *ac*tivations and *re*pressions, the "acrelator". The acrelator circuits, similarly to the repressilator, are ideal in that they contain the minimum number of regulatory connections, matching the number of nodes in the system. Fewer connections would result in some nodes remaining unregulated. An additional advantage of these circuits is the presence of a lower duty cycle in certain nodes. Further details of these circuits can be found in the SI Section 2.6.2.

Experimental example: a light sensor

In the previous section, we demonstrated how to use GRN_modeler to theoretically explore the behavior of new circuit designs. However, preliminary experimental data are often available when investigating a GRN. In this case, a model can help to gain further insight to support data interpretation and to suggest additional experiments. In this section, we provide an example demonstrating the interplay between GRN_modeler and experimental approaches in the development of a light-sensitive biosensor.

We sought to engineer *E. coli* as a bacterial light sensor. We envisioned *E. coli* growing on a solid gel surface and tracking light intensity and fluctuations over several days. Light exposure regulates the expression levels of the biosensor circuitry, resulting in dynamic encoding of the input in the form of concentric rings within the colony. This approach translates dynamical changes of the input into a spatial pattern, which is very much reminiscent of concentric seasonal rings within tree trunks. Likewise, the biosensor would allow us to decode the level of input at each given time point without continuous output recording.

Engineering the light-driven biosensor

To construct the light sensor, we chose to employ the VVD-AraC (BLADE system) [48], a protein composed of the C-terminal DNA binding domain of the transcription factor AraC fused to a light-triggered dimerization domain (Vivid, VVD) [49]. In the presence of light, VVD-AraC dimerizes and binds to the pBAD promoter to activate the expression

[6]The regulation route for the repressilator: $n_i \dashv n_{i+1} \dashv n_{i+2}$, for the reptolator: $n_i \dashv n_{i+1} \dashv n_{i+1+N/2}$ and $n_i \dashv n_{i+N/2} \dashv n_{i+1+N/2}$, while for the actolator: $n_i \rightarrow n_{i+1}$ and $n_i \dashv n_{i+N/2} \rightarrow n_{i+1+N/2} \dashv n_{i+1}$, where n_i denotes the actual activated node, N is the number of the nodes and $i, j \in \mathbb{Z}_N^+$.

of the downstream gene. We started by characterizing VVD-AraC in liquid culture, using cells harboring the plasmids coding for a fluorescent reporter (sfGFP) controlled by pBAD promoter, and VVD-AraC. As expected, after exposure to varying intensities of blue light, we observed an increase in fluorescence with increasing light intensity (Figure 6a).

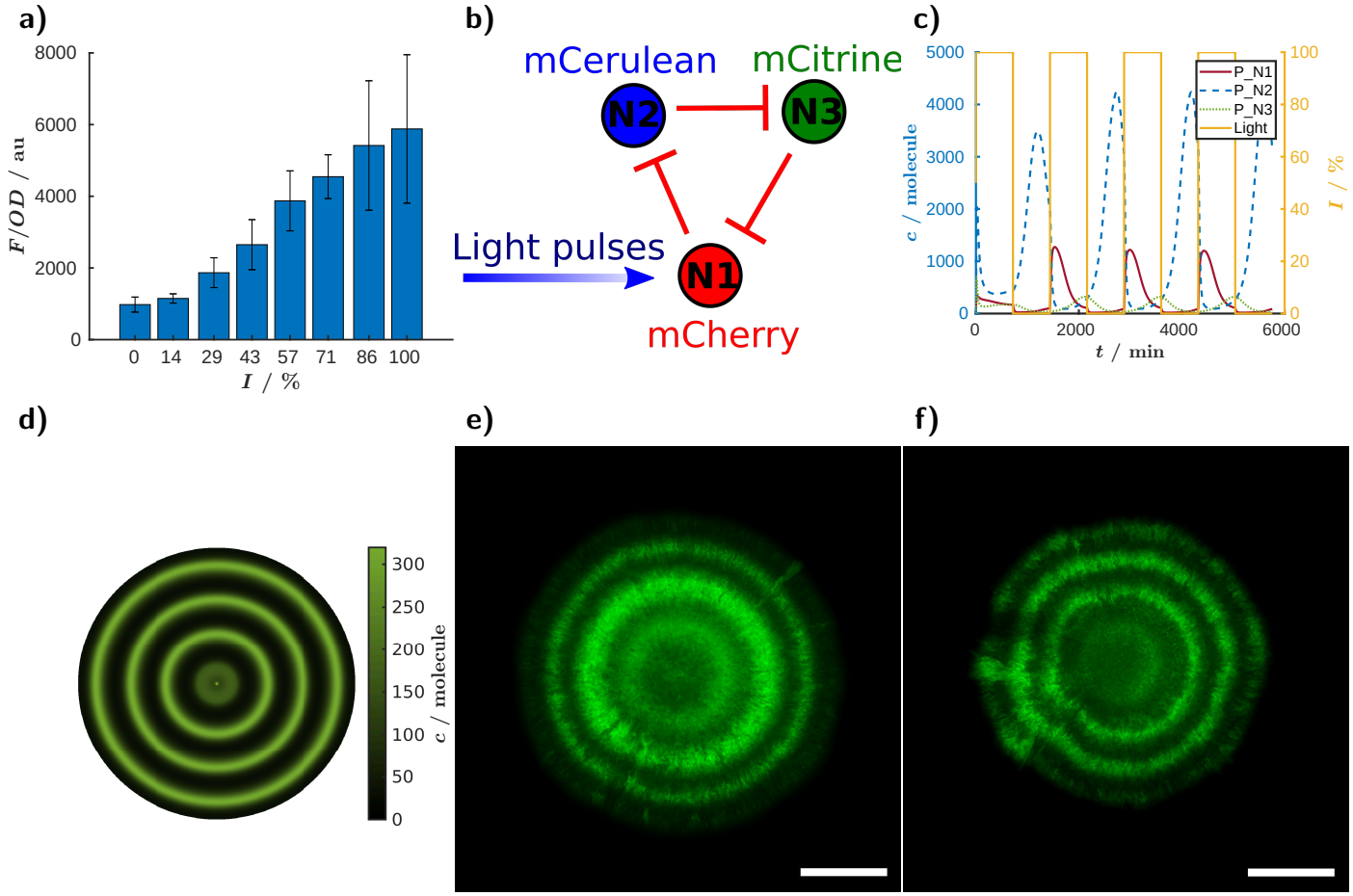

Figure 6: The light biosensor. a) Characterisation of the BLADE system [48]. The cells express VVD-AraC, and pBAD is upstream of sfGFP. Cells were grown for 24 h at different light intensities, then culture OD and GFP fluorescence were measured by a plate reader. Bars represent mean GFP fluorescence intensity (F) ± standard deviation, normalised by the optical density (OD) at 600 nm, as a function of light intensity, for three biological replicates. b) Schematic diagram of the three-node light-sensitive circuit based on the CRISPRlator. c) Calculated reporter concentrations and light intensity as a function of time, generated with GRN_modeler. d) Pseudo two-dimensional image showing the calculated protein concentrations for node N3, generated from GRN_modeler. The detailed information about the model is available in the SI, specifically in the file "light_sensor.html". e) Fluorescence image of a colony grown at room temperature and with light pulses (460 nm) for 4 days. A square wave light intensity was applied from 0% to 15% of device's maximal intensity, with a 41.66% duty cycle. mCitrine fluorescence is shown in green. f) Fluorescence image of a colony grown for 4 days on the bench with direct access to sunlight. mCitrine fluorescence is shown in green. Scale bars = 1mm.

We implemented a 1-node circuit under the control of an external inducer (i.e., light) in the GRN_modeler interface, and set the input to be alternatively "ON" and "OFF". Based on our simulations, a single-node light-sensitive circuit should

be capable of tracking changes in light intensity over the day-night cycle and recording them as ring patterns. Briefly, cells with a fluorescent reporter can form ring patterns as the colony grows on solid medium. Active cells at the colony edge are inducible by light, while inner cells enter stationary phase, "freezing" the level of fluorescence at the corresponding space-time, which creates the rings. To test this prediction, we employed a 1-node circuit using light-inducible mCitrine to dynamically sense fluctuations in light. Figure S1 shows that at high light intensities, this simple 1-node sensor resulted in alternating bright and dark fluorescent rings, effectively registering changes in light intensity. However, the sensitivity was low, with a maximally 1.25-fold difference between the brighter and darker rings.

Thereby, we wondered whether a more complex circuit with negative feedback would improve the biosensor dynamical range output, so we opted for the CRISPRlator, since our group was already working with this circuit. We sought to construct a CRISPRlator that is light-inducible (Figure 6b). We modeled this circuit by starting from the CRISPRlator model. Although the CRISPRlator oscillates in liquid medium [29], it does not oscillate under the conditions tested when grown on a solid surface. Therefore, in order to resemble our biological system as closely as possible, we tuned the parameters so that the CRISPRlator model operates outside its oscillatory regime. Compared to the model presented in Table S4, we used the parameter $k_{rdsd} = 0.7762\,min^{-1}$ as suggested by Santos-Moreno *et al.* [43] for non-oscillatory circuits. Additionally, the dilution rate was reduced to $k_d = 0.005\,min^{-1}$, as cell division occurs more slowly on a solid surface [50]. The model generated from GRN_modeler confirmed that by applying light pulses, the expression of the fluorescent reporters should follow the input level (Figure 6c). In Figure 6d, we illustrate an interesting feature of our application for such cases: the ability to generate pseudo two-dimensional images from time series data. This pattern is not the result of a spatial simulation but is instead produced by assuming a constant growth rate and a small active zone at the edge of the colony, with the image generated by "rotating" the trajectories. By using this feature, the user can not only know the expression level of their circuit in liquid culture, but also predict the spatial patterns that the system generates when grown on solid culture.

Experimentally, we made the CRISPRlator light-inducible by cloning a pBAD promoter upstream of the mCherry node and co-transforming the circuit together with a plasmid (pJP_Bla01) coding for VVD-AraC (Figure 6b). Similarly to the original CRISPRlator, the light-inducible version did not oscillate on solid medium (Figure S2). We grew cells containing the light-inducible CRISPRlator on agar plates with light that changed dynamically in a square wave shape (period 24 h, 41.6% duty-cycle) - resembling day-night cycles. Light pulses led to the expression of mCitrine (shown in green) at the edge of a growing colony, resulting in concentric fluorescent rings (Figure 6e). This proved that our light sensor was capable to register the changes in the light intensity. Moreover, it was able to follow the light intensity changes in a wide range of light pulses, from period as short as 12 h to period as long as 40 hours (Figure S3). Encouraged by these results, we demonstrated the light-sensing capabilities of our system in a real-world application: We placed the petri dish on the laboratory bench, exposing it directly to natural light. As shown in Figure 6f, our circuit successfully detected changes in ambient light levels.

We wondered what determined the substantial difference in the performance of the 1-node and the 3-node circuits. As our reaction kinetics model predicted visible rings in both circuits, it seems that the observed difference arises from an effect not accounted for in these simulations. We hypothesize that cells that are initially dark and still close to the edge of the colony can still partially react to the next light pulse and increase their fluorescence signal. This leads to a low dynamic range between the brighter and darker rings in the 1-node circuit. However, in the 3-node circuit, there is time-delay between the light-inducible node and the reporter node, due to the time it takes for the two repressions to happen (Figure 6b). Thanks to this time delay, the dark regions become inactive before they have time to increase their fluorescence signal. This results in a higher dynamic range between bright and dark regions, which translates to

sharper rings. This phenomenon is linked to the growth of colonies, so it is not reflected in our ODE-based simulations, which consistently produce sharp rings. More detailed spatial simulations would be necessary to observe this effect computationally.

A two-input biosensor produces anti-phase rings

Besides the plasmid-encoded VVD-AraC, our working strain also contains a genomic copy of the (wild-type) AraC transcription factor. In the presence of L-arabinose, AraC dimerizes and activates the pBAD promoter. This chemical induction is up to 8-fold stronger (Figure 7a) than the light induction. Consequently, in the presence of L-arabinose and light, both wild-type AraC (genomically expressed) and VVD-AraC (expressed from the plasmid) activate the same promoter. Contrary to our initial expectation that the two inducers would act synergistically, for L-arabinose concentrations $> 10^{-4}\%$, addition of light results in lower expression than induction with L-arabinose only (Figure 7a-b). Our explanation is that VVD-AraC has significantly lower activation efficiency compared to AraC. In the presence of both inducers, VDD-AraC competes with AraC for the same promoter, thus decreasing the binding of AraC and reducing the overall expression (Figure 7c-d). We describe this phenomenon with a two variable Hill function:

$$\mathrm{HILL}([\mathrm{Ara}], I_{light}) = \frac{([\mathrm{Ara}]/K_{ara})^{n_{ara}} + k_{light}\,(I_{light}/K_{light})^{n_{light}}}{(1 + ([\mathrm{Ara}]/K_{ara})^{n_{ara}}) \cdot (1 + (I_{light}/K_{light})^{n_{light}})}, \tag{1}$$

where both light and arabinose can activate the system, [Ara] and I_{light} are the arabinose concentration and the light intensity, k_{light} represents the strength of the light-inducible system compared to L-arabinose, while n_{ara}, K_{ara}, n_{light} and K_{light} are the Hill exponents and half saturation constants for the L-arabinose and for the light-inducible system, respectively. To achieve a quantitative agreement between the model and the experimental data, we fitted the parameters (Table S5) using MATLAB's `lsqnonlin` function. In Figure S12, we demonstrate how this two-input node can be handled within GRN_modeler.

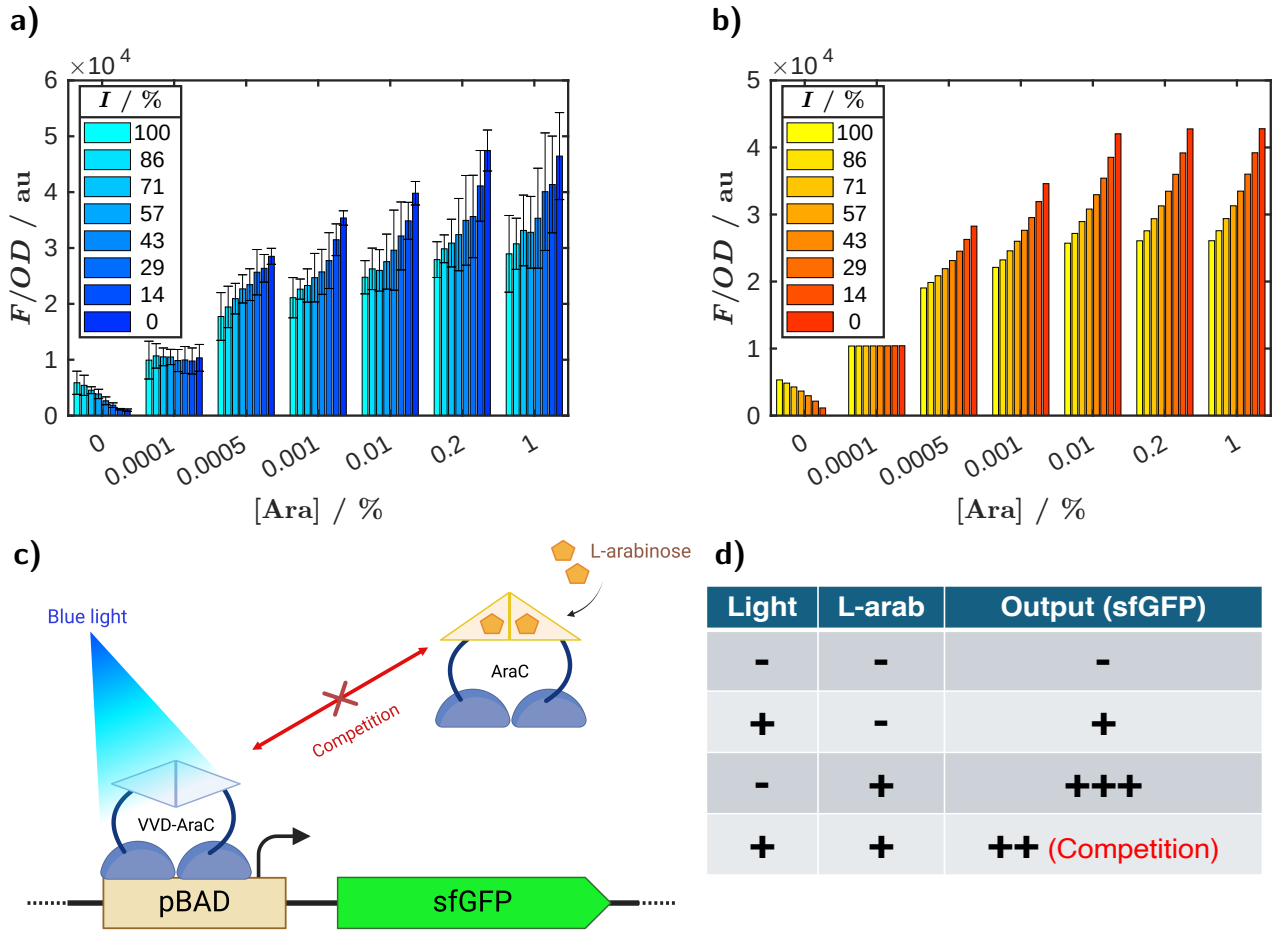

Figure 7: Characterization of the system's response to light and L-arabinose. a) The cells express VVD-AraC, and pBAD is upstream of sfGFP. Cells were grown for 24 h at different light intensities and L-arabinose concentrations, then culture OD and GFP fluorescence were measured by a plate reader. Bars represent mean GFP fluorescence intensity (F) $\pm$ standard deviation (three biological replicates), normalised by the optical density (OD) at 600 nm, as a function of L-arabinose concentration and light intensity. b) Model fitted to experimental results shown in (a). c) Schematic illustration of the light-inducible system: a depiction of the competition between VVD-AraC and wild-type AraC, activated by blue light and L-arabinose, respectively. d) Summarizing table showing the observed output with different combination of inputs. Competition is seen when both light and L-arabinose are present.

Using the model with the fitted parameters, we simulated the patterns generated by the biosensor in response to light and L-arabinose. The simulations predicted that the sensor would form rings when exposed to light pulses and L-arabinose concentrations exceeding $> 10^{-4}\%$. Remarkably, these rings would be in anti-phase compared to the rings produced in the absence of L-arabinose (that is, dark rings would replace bright rings, and vice versa; Figure 8c). We then performed the experiment with square-wave light pulses in the presence of 0.2% L-arabinose. Indeed, L-arabinose inverted the phase of ring patterns upon light pulses (Figure 8d-f, Figure S4). In other words, in the absence of L-arabinose, the light exposure

leads to the expression of mCitrine. When L-arabinose is added, the pBAD promoter is strongly activated, so in the absence of light the entire colony is green. However, due to competition between AraC and VVD-AraC, light exposure decreases the expression of the first and third nodes (mCitrine), resulting in anti-phase rings in the colony (Figure 8e).

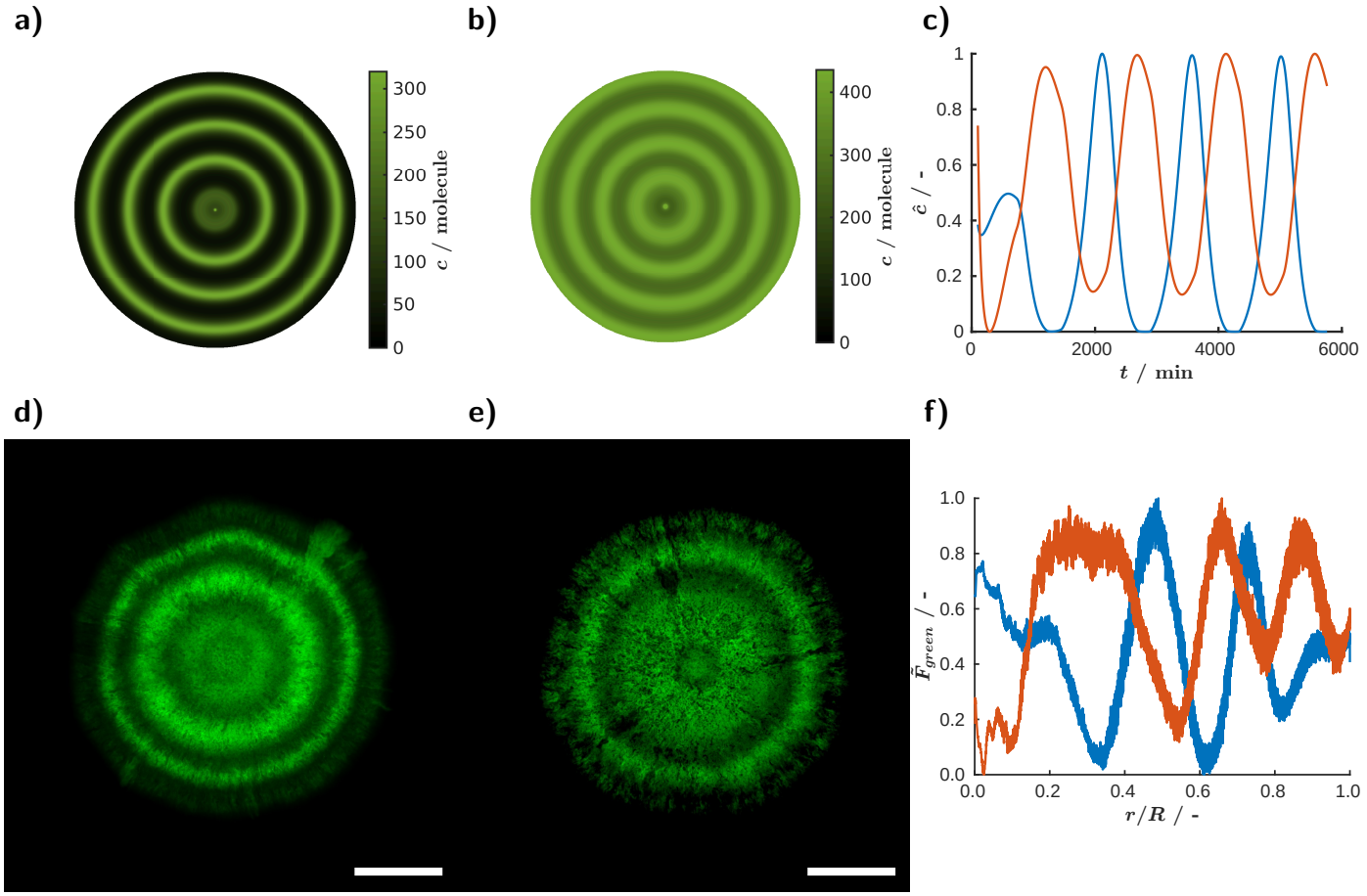

Figure 8: Phase and anti-phase response of the light-sensing CRISPRlator in absence and presence of L-arabinose. a-b) Pseudo two-dimensional images based on simulations of mCitrine expression under light pulses, in absence (a) or in presence of 0.2% L-arabinose (b). c) Normalized mCitrine protein concentrations over time from simulations in absence (blue) or in presence of 0.2% L-arabinose (red). The detailed information about the model is available in the SI, specifically in the file "light_sensor.html". d-e) Fluorescence microscopy images showing mCitrine (green) expression in colonies harboring the light-sensing CRISPRlator. The colonies were grown under light pulses of 24 h (41.6% duty-cycle) in the absence (d) or presence of 0.2% L-arabinose (e). Scale bar = 1 mm. f) Normalized and baseline corrected mCitrine fluorescence intensity as a function of the colony radius derived from the microscopy images of colonies grown with (red) and without L-arabinose (blue).

Increasing the light sensitivity with L-arabinose

The use of high and low L-arabinose concentrations in the light sensor goes beyond creating interesting patterns – it enables the development of more precise and reliable light-recording biosensors. In Figure 9a, we illustrate how relative sensitivity to light varies with different L-arabinose concentrations under varying light exposures. Relative sensitivity

$(S_{F,I})$ is defined as:

$$S_{F,I} = \left| \frac{\partial F}{\partial I} \frac{I}{F} \right|. \tag{2}$$

This dimensionless quantity describes how the recorded fluorescence intensity (F) changes with variations in applied light intensity (I). The absolute value simplifies the comparisons across different L-arabinose concentration regions. Without this adjustment, the sensitivity at higher L-arabinose concentrations would be negative, as increased light intensity results in decreased fluorescence (producing anti-phase rings, as previously discussed). The relative sensitivity was calculated based on the function fitted in Figure 7b. A higher sensitivity value indicates more reliable measurements. The maximum light intensity produced by the light device LITOS (2.3 W$\cdot m^{-2}$) is still lower than the total sunlight spectrum (97.8 W$\cdot m^{-2}$) measured during the daytime in our lab, meaning that it is realistic to extrapolate the light intensity in our model to $> 100\%$. Interestingly, at low light exposure ($< 150\%$), relative sensitivity is greatest at low L-arabinose concentrations ($< 10^{-4}\%$). Conversely, at high light intensity ($> 150\%$), sensitivity is highest at higher L-arabinose concentrations ($> 10^{-4}\%$). For ambient light exposure, where light intensity fluctuates over a wide range, we thus predict that using two sensors with both high and low L-arabinose concentrations would ensure higher accuracy across the full intensity spectrum. Notably, the relative sensitivity approaches zero around $10^{-4}\%$ L-arabinose, where the strength of the light and L-arabinose systems are balanced, resulting in a light insensitive system.

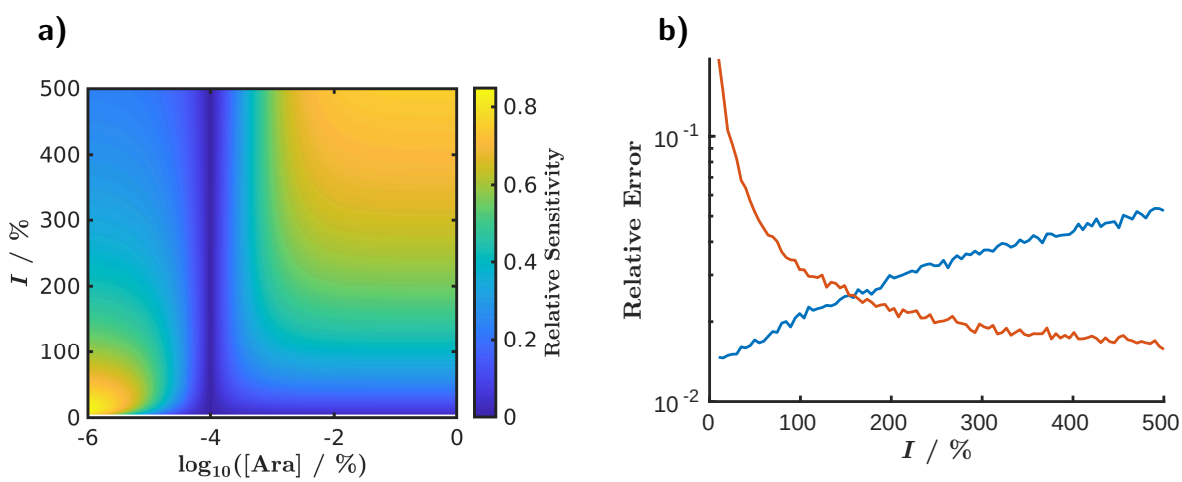

Figure 9: The impact of L-arabinose concentration on the light sensor. a) Relative sensitivity of the measured fluorescence intensity as a function of L-arabinose concentration and light intensity. b) Relative error from simulated experiments as a function of light intensity for low (blue curve, $10^{-6}\%$) and high (red curve, $10^{0}\%$) L-arabinose concentrations. The mathematical definitions of relative sensitivity and relative error, along with details of the simulated experiments, are provided in the text.

In Figure 9b, we show the results of a simulated experiment investigating how arabinose concentration affects the precision of light measurements. Fluorescence intensity (F) was calculated for two L-arabinose concentrations: low ($10^{-6}\%$) and high (1%), as a function of light intensity (ranging from 0% to 500%). To simulate experimental data (F_{exp}), Gaussian noise ($\mathcal{N}(0,1)$) with an amplitude $A = 0.1$ was added, resulting in $F_{exp} = (A \cdot \mathcal{N}(0,1) + 1)F$. This process was repeated 1,000 times, and the light intensity was estimated from the simulated fluorescence data using MATLAB's

`nlinfit` function. Finally, we determined the relative error (err) from the 95% confidence interval ($w_{95\%}$) of the fit:

$$err = \frac{w_{95\%}}{I_{fit}}, \tag{3}$$

where I_{fit} is the fitted light intensity. Consistent with our earlier conclusions, low L-arabinose concentrations (shown in blue) yield low relative error at low light exposures, while higher L-arabinose concentrations (shown in red) are advantageous at higher light exposures (Figure 9b). Thus, by using L-arabinose as an input, we can extend the operational range of our light sensor, enabling more reliable measurements across a broader light spectrum.

Discussion

In this work, we have developed a user-friendly application with a graphical interface for simulating GRNs. Initially, we demonstrated the tool's basic features using well-established circuits. We showed how to construct the classic repressilator circuit using both the Elowitz [27] and Tomazou models [28], and how to conduct both deterministic and stochastic simulations. Recognizing the increasing importance of CRISPR-based circuits in synthetic biology, we incorporated and enhanced a recently published model [43] to describe the CRISPRlator [29]. This advancement facilitates the design of subsequent CRISPRi-based genetic circuits by users. Additionally, we showcased the parameter scan functionality with examples of FFLs, illustrating the tool's capability to explore and analyze different parameter settings.

After presenting well-known circuits, we aimed to demonstrate how our tool can be utilized to design and analyze new GRNs. While the original repressilator family can oscillate with an odd number of nodes, we have developed two new oscillator families: the "reptolator" and the "actolator", which can give rise to robust oscillations with an even number of nodes. These designs incorporate consecutive repression and activation rings combined with toggle switches. Furthermore, we demonstrated that by replacing some repressions with activations in the repressilator, we can create the "acrelator", which can also oscillate with an even number of nodes. Furthermore, the varying duty cycles of the nodes in this configuration could be advantageous for decreasing the burden on the cells or for specific applications such as synthesizing block copolymers, where the properties are influenced by the length of the polymer blocks [51, 52]. The complexity of these newly designed circuits represents an exciting and challenging opportunity for their construction and experimental validation in the future. These examples highlight the value of our application as a powerful tool for streamlining the modeling process and significantly accelerating the design of new GRNs.

Finally, we presented an experimental example featuring a CRISPRi-based circuit sensitive to both light and L-arabinose, demonstrating the capability of our software to predict complex dynamics. Specifically, we used the application to model competitive effects between inputs at varying concentrations, leading to phase and anti-phase responses to periodic external light signals. As a potential application, colonies with this circuit could be used to track day-night light intensity changes over time. Moreover, according to the model and our preliminary experimental results, we propose that adjusting L-arabinose concentrations could enhance the accuracy of light measurements.

As research advances, specialized tools continue to emerge for specific fields. For instance, the BioSwitch toolbox is designed for studying the dynamic behavior of gene regulatory networks [53], iBioSim [19, 20] and Flapjack [24] allow for the construction of GRNs using fundamental components. Although these are powerful applications, they may not be as efficient for modeling the phenomenological behavior of larger systems, while other otherwise great tools lack a GUI [25], which can simplify the model building process. Our interface addresses this gap by utilizing nodes as basic building blocks. This results in several advantages: it can significantly facilitate the modeling process, enhance the analysis of complex systems and provide a user-friendly GUI. Our intuitive tool enables users to model GRNs and predict and compare their

dynamical behavior with ease, accelerating the achievement of desired outcomes. Additionally, it simplifies the creation of novel circuits and facilitates the rapid testing of hypotheses, making it easier to explore and validate innovative genetic designs.

Using our application becomes straightforward once a model has been implemented. However, a basic familiarity with SimBiology syntax is necessary to effectively develop or implement new models. In the future, we aim to expand our application to support the implementation of node models. For instance, our current model for the CRISPR system does not yet include a description for CRISPR activation. Among others, future work will involve creating a model that incorporates CRISPR activation and fine-tuning the parameters based on experimental data. Another limitation, as observed in our one node spatial pattern experiments (Figure S1), is that our application only accounts for reaction kinetics. To accurately capture other spatio-temporal phenomena, such as diffusion, additional modeling steps are required.

While our primary focus has been on gene regulatory networks for synthetic biology, other biologists might also use our tool to explore the properties of their favorite (naturally occurring) gene regulatory networks. Here, we have been using GRN_modeler for describing intercellular processes and gene regulatory networks. However, the framework we have developed could be adapted for other applications. In this broader context, "nodes" might represent different cells or even organs or organisms, and "regulators" could denote various interactions between these entities. With appropriate modifications and new models, our tool has the potential to model a wide range of systems.

In conclusion, we aim, with GRN_modeler, to empower (synthetic) biologists, even those with limited expertise in mathematics or programming, to effectively model their systems of interest. By bridging computational and experimental biology, this tool has the potential to accelerate scientific progress and innovation in the field.

Methods

Strains and plasmids

For cloning we used *E. coli* strain DH5α. For the light sensor, we used the *E. coli* strain MK01 [54]. MK01 was a kind gift from Sander Tans (Addgene #195090). The plasmids were constructed, using the methods described previously [55]. All plasmids are summarized in Table S1. Plasmids, and their maps and sequences are available through Addgene. The light-inducible system is based on the pBLADE_ONLY_C plasmid [48]. pBLADE_ONLY_C was a kind gift from Barbara Di Ventura (Addgene plasmid #168050). We modified it to contain a gentamicin resistance gene instead of the chloramphenicol resistance, generating plasmid pJP_Bla01. We created a reporter plasmid starting from plasmid pCOLA-AraC-pBAD-GFP [56], which contains a ColA origin of replication, a copy of *araC* and sfGFP under control of the pBAD promoter. By deleting *araC* from the original backbone, we generated pJP_Ctrl04.

For the 1-node biosensor we constructed plasmid pJP_1node. It contains a pBAD promoter upstream of mCitrine. It was constructed by substituting the sfGFP reporter from pJP_Ctrl04 with mCitrine, making it comparable to the 3-node circuit in the ring pattern experiments.

For the 3-node light- and L-arabinose-inducible biosensor we created plasmid pJP_Osc05. It is derived from pJ2072 (1-OS2), an L-arabinose inducible CRISPRlator [43], which contains a pBAD promoter upstream of the mCherry node. Additionally, we moved the kanamycin resistance gene between the mCitrine and mCerulean nodes. This improved the construct stability, as recombination between mCitrine and mCerulean genes would result in the loss of the resistance gene. We also removed the *araC* gene.

Characterization of light and L-arabinose induction in liquid culture

For characterizing the light and L-arabinose sensing, we co-transformed pJP_Bla01 and pJP_Ctrl04 into MK01. A single colony was inoculated into a tube containing 3 mL of LB liquid medium supplemented with gentamicin (25 μg/mL) and kanamycin (50 μg/mL) and incubated overnight at 30°C with shaking. 0.5 mL of the culture was pelleted, washed with saline water (0.9%) and resuspended in 0.5 mL of EZ medium (Teknova), supplemented with 0.4% glycerol. The resuspended culture was diluted 50x in EZ medium containing the necessary antibiotics and L-arabinose (final concentration of 0, 0.0001, 0.0005, 0.001, 0.01, 0.2 and 1%). For each condition 200 μL were added to a 96-well plate (Greiner, REF #655090). The 96-well plate was placed on top of a LITOS light device [57] and incubated at room temperature under agitation at 450 rpm (Titramax 1000, heidolph). For every L-arabinose concentration, cells were induced with constant blue light (460 nm), at 0, 14, 29,43, 57, 71, 83 and 100% intensity. After 24 hours, OD 600 and sfGFP fluorescence were measured (Ex: 479/12.5, Em: 520/12.5) with a microplate reader (Synergy H1, BioTek). This experiment was performed three times, corresponding to three biological replicates (for which we calculated the mean and standard deviation). For each well, sfGFP measurement was normalized by its respective OD 600. No background subtraction was performed.

Spatial patterning experiment

For the 1-node light-sensor, we transformed pJP_Bla01, pJP_1node and pJ1996_v2 into MK01. For the 3-node light-sensor we transformed pJP_Osc05, pJP_Bla01 and pJ1996_v2 (Addgene plasmid #140664)[29] into MK01. Tubes containing 5 ml LB liquid medium with gentamicin (25 μg/mL), kanamycin (50 μg/mL) and spectinomycin (50 μg/mL) were inoculated with single colonies and incubated overnight at 30°C with shaking. The cultures were serially diluted in saline water (0.9%) in order to seed approximately 20 colonies per plate (90 mm diameter Petri dish containing LB agar, antibiotics as indicated above and 0 or 0.2% L-arabinose). The plates were covered in aluminum foil and placed in a 30 °C incubator. After 20 h, the aluminum foil was removed and the plates were placed upside down on top of the LITOS device [57] for light induction at room temperature. For the phase and anti-phase experiment using the 3-node circuit, light intensity varied from 0% to 15%, in a square wave shape (T = 24h) and 41.66% duty cycle, while for the experiment with different period of light pulses, the duty cycle was 50%. For the 1-node circuit, cells were subject to light pulses with square wave shape (T = 24h) with intensity varying from 0% to L% (L = 0%, 25%, 50%, 75%, 100%) and 50% duty cycle. After 96 h, we took pictures of the colonies using an AxioImager M1 fluorescence microscope (Zeiss) with an Achromat 2.5X/0.12 Fluar objective. YFP (Ex: 500/20, Em: 535/30) filter was used to measure the fluorescence of mCitrine. The exposure time was set to 100 ms.

For better visualization of the anti-phase rings, the contrast of the images was enhanced. Quantification was done on original data and not on the images with enhanced contrast.

Daylight (15 hours daylength) sensing was done following the same method described above for the ring patterns without L-arabinose, but instead of placing the plates on the LITOS device, they were placed near the window in our laboratory, for 4 days, at 22 °C. The indoor white light was turned off after 18h00 and turned on from 9h00.

Image analysis

In our image analysis, we first identified the center of the colony. An intensity threshold was determined using Otsu's method [58], and the colony center was located in the resulting binary image using MATLAB's *regionprops* function. Next, we transformed the image to polar coordinates and calculated the fluorescent intensity as a function of radius. Baseline correction was then applied by fitting a third-order polynomial to the intensity data and subtracting it to enhance the

visibility of individual peaks. We subsequently applied a moving average smoothing with a window of 100 data points. Finally, we rescaled the intensity to the [0,1] range for better comparison across images.

Simulations

We conducted deterministic simulations using MATLAB's `ode15s` and `sundials` solvers with the SimBiology toolbox. For stochastic simulations, we employed the `adaptivesa` solver from COPASI. We built the reaction networks and the differential equations using our application, the GRN_modeler. All examples can be found in the `analyzing_examples` folder of the application. Details on initial conditions, reactions, and rate laws were generated using the SimBiology toolbox and are provided in the SI. The specific filenames for these files are listed in the figure captions that display the respective simulation results.

Competing interests

The authors declare no competing interests.

Author contributions

G.H. designed the application and performed the mathematical modeling. J.P. and Y.S. designed the experimental research. J.P. performed the experiments. J.P. and G.H. analysed the data and prepared the corresponding figures. E.B. demonstrated how to use our application in the instructional video. J.P., G.H, E.B. and Y.S. wrote the manuscript. All authors have given approval to the final version of the manuscript.

Acknowledgements

This work was funded by a Swiss National Science Foundation (310030_200532 awarded to Y.S), a fellowship of the Agassiz foundation (awarded to J.P), a UNIL FBM PhD fellowship in Life Sciences (awarded to J.P) and the University of Lausanne.

References

1. Ranganathan, S., Nakai, K. & Schonbach, C. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics* 1st (Elsevier Science Publishers B. V., NLD, 2018).
2. Walhout, M., Vidal, M. & Dekker, J. *Handbook of Systems Biology: Concepts and Insights* 1st (Academic Press, Inc., USA, 2018).
3. Hanczyc, M. M. Engineering Life: A Review of Synthetic Biology. *Artificial Life* **26,** 260–273 (2020).
4. Benner, S. A. & Sismour, A. M. Synthetic biology. *Nature Reviews Genetics* **6,** 533–543 (2005).
5. Garner, K. L. Principles of synthetic biology. *Essays in Biochemistry* **65,** 791–811 (2021).
6. Ermentrout, B. *Simulating, Analyzing, and Animating Dynamical Systems* (Society for Industrial and Applied Mathematics, 2002).

7. Johnson, K. A., Simpson, Z. B. & Blom, T. Global Kinetic Explorer: A new computer program for dynamic simulation and fitting of kinetic data. *Analytical Biochemistry* **387,** 20–29 (2009).

8. Johnson, K. A., Simpson, Z. B. & Blom, T. FitSpace Explorer: An algorithm to evaluate multidimensional parameter space in fitting kinetic data. *Analytical Biochemistry* **387,** 30–41 (2009).

9. Johnson, K. A. in *Computer Methods Part B* (eds Johnson, M. L. & Brand, L.) 601–626 (Academic Press, 2009).

10. Hucka, M. *et al.* The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19,** 524–531 (2003).

11. Kitano, H. Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. English. *Systems Biology* **1,** 41–53(12) (1 2004).

12. Kosuri, S., Kelly, J. R. & Endy, D. TABASCO: A single molecule, base-pair resolved gene expression simulator. *BMC Bioinformatics* **8,** 480 (2007).

13. Hoops, S. *et al.* COPASI—a COmplex PAthway SImulator. *Bioinformatics* **22,** 3067–3074 (2006).

14. Swaminathan, A., Poole, W., Pandey, A., Hsiao, V. & Murray, R. M. Fast and flexible simulation and parameter estimation for synthetic biology using bioscrape. *bioRxiv* (2022).

15. Somogyi, E. T., Karlsson, M. T., Swat, M., Galdzicki, M. & Sauro, H. M. libRoadRunner: A High Performance SBML Compliant Simulator. *bioRxiv* (2013).

16. Welsh, C. *et al.* libRoadRunner 2.0: a high performance SBML simulation and analysis library. *Bioinformatics* **39,** btac770 (2022).

17. Dörr, A., Keller, R., Zell, A. & Dräger, A. SBMLSimulator: A Java Tool for Model Simulation and Parameter Estimation in Systems Biology. *Computation* **2,** 246–257 (2014).

18. Garcia-Campos, M. A. & Schwartz, S. txtools: an R package facilitating analysis of RNA modifications, structures, and interactions. *Nucleic Acids Research* **52,** e42–e42 (2024).

19. Myers, C. J. *et al.* iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics* **25,** 2848–2849 (2009).

20. Watanabe, L. *et al.* iBioSim 3: A Tool for Model-Based Genetic Circuit Design. *ACS Synthetic Biology* **8,** 1560–1563 (2019).

21. Singhal, V., Tuza, Z. A., Sun, Z. Z. & Murray, R. M. A MATLAB toolbox for modeling genetic circuits in cell-free systems†. *Synthetic Biology* **6,** ysab007 (2021).

22. McClymont, K. & Soyer, O. S. Metabolic tinker: an online tool for guiding the design of synthetic metabolic pathways. *Nucleic Acids Research* **41,** e113–e113 (2013).

23. Philippou, J., Yáñez Feliú, G. & Rudge, T. J. WebCM: A Web-Based Platform for Multiuser Individual-Based Modeling of Multicellular Microbial Populations and Communities. *ACS Synthetic Biology* **13,** 1952–1955 (2024).

24. Yáñez Feliú, G. *et al.* Flapjack: Data Management and Analysis for Genetic Circuit Characterization. *ACS Synthetic Biology* **10.** PMID: 33382586, 183–191 (2021).

25. Vidal, G., Vitalis, C. & Rudge, T. J. LOICA: Integrating Models with Data for Genetic Network Design Automation. *ACS Synthetic Biology* **11.** PMID: 35507566, 1984–1990 (2022).

26. Rossi, N. A. & Dunlop, M. J. Making Waves with Synthetic Oscillators. *Cell Systems* **6,** 406–407 (2018).

27. Elowitz, M. B. & Leibler, S. A synthetic oscillatory network of transcriptional regulators. *Nature* **403,** 335–338 (2000).

28. Tomazou, M., Barahona, M., Polizzi, K. M. & Stan, G.-B. Computational Re-design of Synthetic Genetic Oscillators for Independent Amplitude and Frequency Modulation. *Cell Systems* **6,** 508–520.e5 (2018).

29. Santos-Moreno, J., Tasiudi, E., Stelling, J. & Schaerli, Y. Multistable and dynamic CRISPRi-based synthetic circuits. *Nature Communications* **11,** 2746 (2020).

30. Weldemichael, T., Asemoloye, M. D. & Marchisio, M. A. Feedforward Loops: Evolutionary Conserved Network Motifs Redesigned for Synthetic Biology Applications. *Applied Sciences* **12** (2022).

31. Mangan, S. & Alon, U. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences* **100,** 11980–11985 (2003).

32. Alon, U. Network motifs: theory and experimental approaches. *Nature Reviews Genetics* **8,** 450–461 (2007).

33. Schaerli, Y. *et al.* A unified design space of synthetic stripe-forming networks. *Nature Communications* **5,** 4905 (2014).

34. Basu, S., Gerchman, Y., Collins, C. H., Arnold, F. H. & Weiss, R. A synthetic multicellular system for programmed pattern formation. *Nature* **434,** 1130–1134 (2005).

35. Santos-Moreno, J., Tasiudi, E., Kusumawardhani, H., Stelling, J. & Schaerli, Y. Robustness and innovation in synthetic genotype networks. *Nature Communications* **14,** 2454 (2023).

36. Tickman, B. I. *et al.* Multi-layer CRISPRa/i circuits for dynamic genetic programs in cell-free and bacterial systems. *Cell Systems* **13,** 215–229.e8 (2022).

37. Bergmann, F. T. BASICO: A simplified Python interface to COPASI. *Journal of Open Source Software* **8,** 5553 (2023).

38. Zhang, F. *et al.* Independent control of amplitude and period in a synthetic oscillator circuit with modified repressilator. *Communications Biology* **5,** 23 (2022).

39. Santos-Moreno, J. & Schaerli, Y. CRISPR-based gene expression control for synthetic gene circuits. *Biochemical Society Transactions* **48,** 1979–1993 (2020).

40. Bikard, D. *et al.* Programmable repression and activation of bacterial gene expression using an engineered CRISPR-Cas system. *Nucleic acids research* **41,** 7429–7437 (2013).

41. Chappell, J., Watters, K. E., Takahashi, M. K. & Lucks, J. B. A renaissance in RNA synthetic biology: new mechanisms, applications and tools for the future. *Current opinion in chemical biology* **28,** 47–56 (2015).

42. Tsai, S. Q. *et al.* Dimeric CRISPR RNA-guided FokI nucleases for highly specific genome editing. *Nature biotechnology* **32,** 569–576 (2014).

43. Santos-Moreno, J., Tasiudi, E., Kusumawardhani, H., Stelling, J. & Schaerli, Y. Robustness and innovation in synthetic genotype networks. *Nature Communications* **14,** 2454 (2023).

44. Basu, S., Gerchman, Y., Collins, C. H., Arnold, F. H. & Weiss, R. A synthetic multicellular system for programmed pattern formation. *Nature* **434,** 1130–1134 (2005).

45. Greber, D. & Fussenegger, M. An engineered mammalian band-pass network. *Nucleic Acids Research* **38,** e174–e174 (2010).

46. Niederholtmeyer, H. *et al.* Rapid cell-free forward engineering of novel genetic ring oscillators. *elife* **4,** e09771 (2015).

47. Gardner, T. S., Cantor, C. R. & Collins, J. J. Construction of a genetic toggle switch in Escherichia coli. *Nature* **403,** 339–342 (2000).

48. Romano, E. *et al.* Engineering AraC to make it responsive to light instead of arabinose. *Nature chemical biology* **17,** 817–827 (2021).

49. Schwerdtfeger, C. & Linden, H. VIVID is a flavoprotein and serves as a fungal blue light photoreceptor for photoadaptation. *The EMBO journal* (2003).

50. Kim, K., Kim, S. & Jeon, J. S. Visual estimation of bacterial growth level in microfluidic culture systems. *Sensors* **18,** 447 (2018).

51. Lee, H.-G. *et al.* The effect of rigid-block length in elastomer-containing photoactive block copolymers on the photovoltaic and mechanical properties of polymer solar cells. *J. Mater. Chem. A* **12,** 19039–19051 (30 2024).

52. Dai, B. *et al.* Influence of block lengths and symmetries of block copolymers on phase behavior of polymer A/polymer B/block copolymer ternary blends. *Polymer* **45,** 1019–1026 (2004).

53. Yordanov, P., Stelling, J. & Otero-Muras, I. BioSwitch: a tool for the detection of bistability and multi-steady state behaviour in signalling and gene regulatory networks. *Bioinformatics* **36,** 1640–1641 (2019).

54. Kogenaru, M. & Tans, S. J. An improved Escherichia coli strain to host gene regulatory networks involving both the AraC and LacI inducible transcription factors. *Journal of biological engineering* **8,** 1–5 (2014).

55. Park, J. H., Holló, G. & Schaerli, Y. From resonance to chaos by modulating spatiotemporal patterns through a synthetic optogenetic oscillator. *Nature Communications* **15,** 7284 (2024).

56. Duarte, J. M., Barbier, I. & Schaerli, Y. Bacterial microcolonies in gel beads for high-throughput screening of libraries in synthetic biology. *ACS synthetic biology* **6,** 1988–1995 (2017).

57. Höhener, T. C. *et al.* LITOS: a versatile LED illumination tool for optogenetic stimulation. *Scientific reports* **12,** 13139 (2022).

58. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics* **9,** 62–66 (1979).