Inferring causality in biological oscillators

Jonathan Tyler,[1,2] Daniel Forger,[1,3] Jae Kyoung Kim[4,5,*]

[1]Department of Mathematics, University of Michigan
[2]Division of Pediatric Hematology/Oncology, Department of Pediatrics
University of Michigan
[3]Department of Computational Medicine and Bioinformatics, University of Michigan
Ann Arbor, MI, 48109
[4]Department of Mathematical Sciences
Korea Advanced Institute of Science and Technology, Daejeon, 34141, Republic of Korea
[5]Biomedical Mathematics Group
Institute for Basic Science, Daejeon, 34126, Republic of Korea
[*]To whom correspondence should be addressed; E-mail: jaekkim@kaist.ac.kr

A fundamental goal of biological study is to identify regulatory interactions among components. The recent surge in time-series data collection in biology provides a unique opportunity to infer regulatory networks computationally. However, when the components oscillate, model-free inference methods, while easily implemented, struggle to distinguish periodic synchrony and causality. Alternatively, model-based methods test whether time series are reproducible with a specific model but require inefficient simulations and have limited applicability. Here, we develop an inference method based on a general model of molecular, neuronal, and ecological oscillatory systems that merges the advantages of both model-based and model-free methods, namely accuracy, broad applicability, and usability. Our method successfully infers the positive and negative regulations of various oscillatory networks, including the repressila-

tor and a network of cofactors of pS2 promoter, outperforming popular inference methods. We also provide a computational package, ION (Inferring Oscillatory Networks), that users can easily apply to noisy, oscillatory time series to decipher the mechanisms by which diverse systems generate oscillations.

Introduction

A fundamental goal in biology is to uncover the causal interactions among system components. To identify the casual interactions, conventional methods require experimental manipulation of one or more components to investigate the effect on others in the system. However, this approach is time-consuming and costly, particularly when the number of components in a system increases. On the other hand, thanks to recent technological advances (e.g., GFP, luciferase, microarray, etc.), measuring time-series data has become relatively easy. Accordingly, inferring direct regulations along with type (positive/negative) solely given time-series data is an important tool to provide key insights into the mechanisms underlying the system in a timely and inexpensive manner (*1*).

The unprecedented growth in the amount of biological data has revealed that biological processes frequently exhibit oscillatory behavior in time-series data, e.g., about half of the protein-coding genome is transcribed rhythmically (*2, 3*). To infer networks from oscillatory data, a popular model-free method, Granger Causality (GC) based on predictability, i.e., X causes Y if X has unique information that can improve the prediction of Y, has been used (*4, 5*). However, as GC relies heavily on the assumption that the time-series data are stationary (*6*), it is challenging to apply GC to highly nonstationary oscillatory time-series data (*5, 7–9*). To overcome this limitation of GC, inference methods for dynamical systems, such as Convergent Cross Mapping (CCM), have been developed, based on a differing view of predictability, i.e., X causes Y if historical values of X can be recovered from Y alone (*10–20*). Despite the

success of CCM methods in many biological applications, they frequently infer interactions between independent components when they oscillate with similar periods due to difficulty in distinguishing synchrony and casual interaction (*21*), indicating that these methods are likely to infer false-positive interactions in oscillatory networks. Nonetheless, these model-free methods remain widely used due to their ease of implementation and broad applicability to a large class of networks.

Alternatively, model-based methods were proposed that infer causality by determining whether time-series data are reproducible with mechanistic models. Testing the reproducibility requires computationally-expensive model simulations and fittings (*22–35*), but, as long as the underlying model is accurate, model-based methods do not suffer from false positive predictions unlike model-free methods. However, the inference results strongly depend on the choice of model, which is frequently based on limited information. Thus, inference methods using more general ODE forms were developed (*36–45*). For example, previously, we developed a method that infers causation from X to Y by checking whether oscillatory time-series data for X and Y are reproducible with a common ODE model for biological oscillators: $\frac{\mathrm{d}Y}{\mathrm{d}t} = f(X) - g(Y)$, where f and g describe the synthesis and degradation rates of Y, respectively (*41*). Pigolotti et al. (*36*) considered the most general possible mechanistic model between two components:

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = f(X, Y). \tag{1}$$

However, unlike (*41*), this method uses only the minima and maxima rather than all of the time-series data (*36*). Thus, the method requires the restrictive assumption that all given components are in a single negative feedback loop (i.e., the method determines the order of given components in a feedback loop). Moreover, extensions of the method (*37, 38*) require that a single negative feedback loop structure drive the dynamics, limiting their applicability.

Here, we develop an inference method for biological oscillators described by Eqn. (1) that merges the advantages of model-based and model-free methods, namely usability, broad ap-

plicability, and accuracy, while mitigating the drawbacks of each. Specifically, we identify a fundamental relationship between the general model (Eqn. (1)) and its oscillatory solution. By using this relationship, we develop a simple functional transformation (i.e., regulation-detection function) of a pair of oscillatory time-series data that easily tests whether the time-series data are reproducible with the general model. This transformation enables accurate and precise inference of the (self-)regulation type (e.g., positive, negative, or a mixture) between two components X and Y described by Eqn. (1). This allows us to infer various network structures such as a cycle, multiple cycles, and a cycle with outputs from *in silico* oscillatory time-series data. Furthermore, our method also successfully infers regulation types from noisy experimental data measured at the molecular and organismal levels. In particular, from time-series data of the repressilator and cofactors at the pS2 promoter, our method infers networks that match current biological knowledge while popular model-free methods incorrectly infer nearly fully connected networks. Importantly, our method predicts hidden regulations for the pS2 promoter after estradiol treatment, which guides investigation. We also provide a user-friendly computational package (ION: Inferring Oscillatory Networks) that implements our method to infer network structures of biological oscillators, which requires minimal user effort.

Results

Inferring regulation types from oscillatory time series

In the reduced FitzHugh-Nagumo model (Fig. 1A) (*46*), which describes the interactions between the membrane potential of a neuron (V) and the accommodation and refractoriness of the membrane (W) (*46, 47*), W positively regulates V while V negatively regulates W. In addition, V displays a mixture of positive and negative self-regulation while W negatively regulates itself.

How are such inter- and self-regulations reflected in the oscillatory change of V and W

simulated with the model (Fig. 1B)? The change in V and W does not directly reflect their regulatory interactions. For instance, although W positively regulates V, when W increases, V does not always increase (e.g., in the region highlighted in yellow, Fig. 1B). This is because W positively regulates $\dot{V}$ rather than the value of V (Fig. 1A). However, the relationship between the change in W and $\dot{V}$ also does not reflect the positive regulation of W on V. For example, in the yellow region (Fig. 1B), $\dot{V}$ decreases despite increasing W, which happens because the self-regulation of V on $\dot{V}$ masks the effect of W on $\dot{V}$. Thus, to infer the effect of W on $\dot{V}$ independently of V, we investigate the relationship between W and $\dot{V}$ at the pair of time points t and the *reflection time*, t_V, where $V(t) = V(t_V)$ (Fig. 1B). As $V(t) = V(t_V)$, the difference in $\dot{V}(t) = f(V(t), W(t))$ and $\dot{V}(t_V) = f(V(t_V), W(t_V))$ is solely determined by the difference between $W(t)$ and $W(t_V)$. Thus, because W positively regulates V (Fig. 1A), if $W(t)$ is greater (less) than $W(t_V)$, $\dot{V}(t)$ should be greater (less) than $\dot{V}(t_V)$. Similarly, to infer the type of self-regulation of V, we must remove the variation of $\dot{V}$ due to W that masks the effect of V on $\dot{V}$. Thus, we investigate the relationship between V and $\dot{V}$ at the pair of time points t and the reflection time, t_W, where $W(t) = W(t_W)$ (Fig. 1B). To quantify such relationships between W and $\dot{V}$ and V and $\dot{V}$, we develop the *regulation-detection functions*:

$$\begin{aligned} R^{t_V}_{W\to V}(t) &:= (W(t) - W(t_V)) \cdot (\dot{V}(t) - \dot{V}(t_V)) \\ &:= W^{t_V}_d(t) \cdot \dot{V}^{t_V}_d(t), \end{aligned} \tag{2}$$

and

$$\begin{aligned} R^{t_W}_{V\to V}(t) &:= (V(t) - V(t_W)) \cdot (\dot{V}(t) - \dot{V}(t_W)) \\ &:= V^{t_W}_d(t) \cdot \dot{V}^{t_W}_d(t). \end{aligned} \tag{3}$$

As W positively regulates V, the functions $W^{t_V}_d$ and $\dot{V}^{t_V}_d$ should have the same sign and thus, $R^{t_V}_{W\to V}(t) \geq 0$ throughout the cycle (Fig. 1C, left). That is, if $W^{t_V}_d = W(t) - W(t_V) \geq 0$, then $\dot{V}^{t_V}_d = \dot{V}(t) - \dot{V}(t_V) = 3(W(t) - W(t_V)) \geq 0$ (Fig. 1A). On the other hand, due to the mixture of positive and negative self-regulation of V, the relationship between the signs of $V^{t_W}_d(t)$ and $\dot{V}^{t_W}_d(t)$, and thus the sign of $R^{t_W}_{V\to V}(t)$, varies throughout the cycle (Fig. 1C,

right). As the profiles of the sign of the regulation-detection functions (Eqns. (2) and (3)) reflect the regulation type, we develop a *regulation-detection score* that quantifies the variation in the sign of the regulation-detection functions. For instance, the regulation-detection score for the regulation of W on V is defined as

$$\begin{aligned}\langle R_{W\to V}\rangle :=& \frac{\int_0^\tau R^{t_V}_{W\to V}(t)\mathrm{dt}}{\int_0^\tau |R^{t_V}_{W\to V}(t)|\mathrm{dt}} \\ =& \frac{\text{Positive Area}}{\text{Total Area}} - \frac{\text{Negative Area}}{\text{Total Area}},\end{aligned} \tag{4}$$

where τ is the period (e.g., $\tau = 1$ in Fig. 1C, left). The regulation-detection score $\langle R_{W\to V}\rangle = 1$ because W positively regulates V, and thus $R^V_{W\to V}(t) \geq 0$ (i.e., the negative area is zero) (Fig. 1C, left). On the other hand, because V both positively and negatively regulates itself, $R^{t_W}_{V\to V}(t)$ takes both positive and negative values, so $\langle R_{V\to V}\rangle = 0.6 - 0.4 = 0.2$ (Fig. 1C, right).

Similarly, we can obtain information about the regulation of V on W and the self regulation of W with the regulation-detection functions $R^{t_W}_{V\to W} := V^{t_W}_d \cdot \dot{W}^{t_W}_d(t)$ (Fig. 1D, left) and $R^{t_V}_{W\to W} := W^{t_V}_d \cdot \dot{W}^{t_V}_d(t)$ (Fig. 1D, right). Because V negatively regulates W, $R^{t_W}_{V\to W}(t) \leq 0$. Also, because the self-regulation of W is purely negative, $R^{t_V}_{W\to W}(t) \leq 0$. Thus, $\langle R_{V\to W}\rangle = -1$, and $\langle R_{W\to W}\rangle = -1$ (Fig. 1D). Taken together, in general, if X positively (negatively) regulates Y, then $\langle R_{X\to Y}\rangle = 1$ ($\langle R_{X\to Y}\rangle = -1$) (see Theorem 1 in Supplementary Information).

Next, we calculated the regulation-detection scores from experimentally measured oscillatory time-series data of two bacteria: *Paramecium* and *Didinium*, which we refer to as P and D (Fig. 1E), respectively (*48*). As P is a prey of the predator D (*48*), D is expected to negatively regulate P, and P is expected to positively regulate D. Reflecting this, $\langle R_{P\to D}\rangle = 1$ and $\langle R_{D\to P}\rangle = -1$ (Fig. 1E). Furthermore, reflecting the positive (i.e., birth) and negative (i.e., death) self-regulation of both P and D, $\langle R_{D\to D}\rangle = 0.51 - 0.49 = 0.02$ and $\langle R_{P\to P}\rangle = 0.63 - 0.37 = 0.26$ (Fig. 1E). The regulation-detection scores appear to accurately reflect types of regulation even for noisy and discrete time-series data.

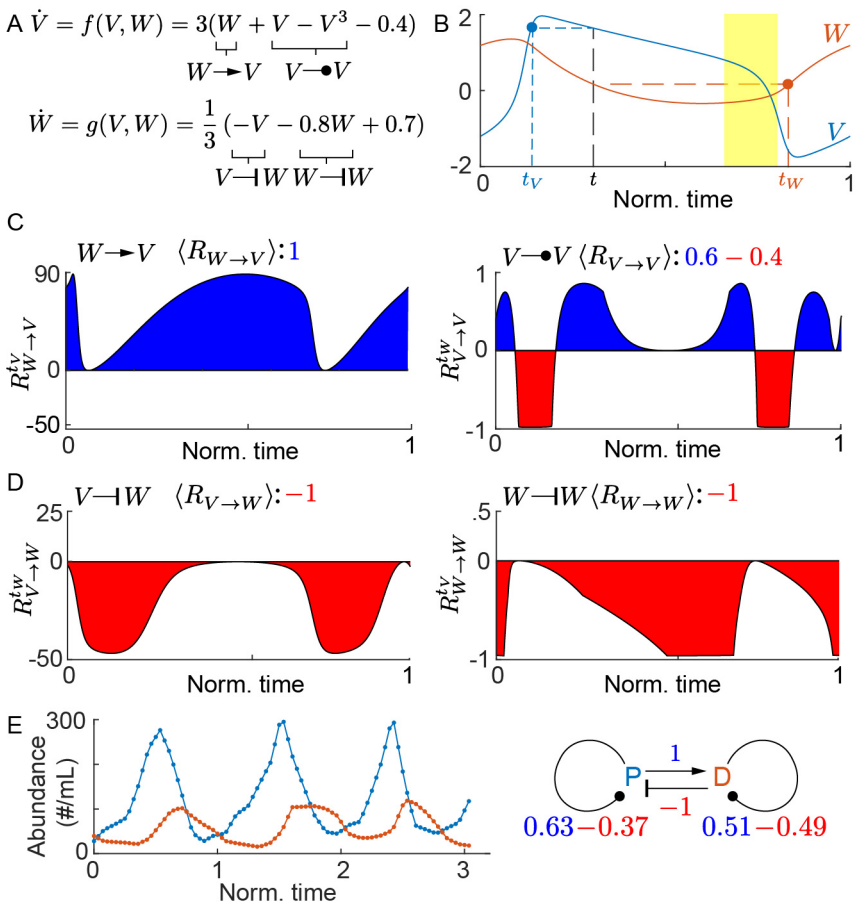


Figure 1: **Regulation-detection functions and scores reflect regulation types.** (A) The FitzHugh-Nagumo model describes the interactions between the membrane potential of a neuron (V) and the accommodation and refractoriness of the membrane (W). W positively regulates V while V negatively regulates W. In addition, V displays a mixture of positive and negative self-regulation while W negatively regulates itself. (B) Time series of one cycle simulated with the FitzHugh-Nagumo model. Although W positively regulates V (i.e., $\dot{V}$ positively depends on W), $\dot{V}$ decreases despite increasing W (yellow region) because the self-regulation of V on $\dot{V}$ masks the effect of W on $\dot{V}$. On the other hand, for the the pair of time points t and reflection time t_V, where $V(t) = V(t_V)$, if $W(t)$ is greater (less) than $W(t_V)$, then $\dot{V}(t)$ should be greater (less) than $\dot{V}(t_V)$. Similarly, as V negatively regulates W, if $V(t)$ is greater (less) than $V(t_W)$, then $\dot{W}(t)$ should be less (greater) than $\dot{W}(t_W)$ for the pair of time points t and t_W, where $W(t) = W(t_W)$. (C) The regulation-detection function $R^{t_V}_{W\rightarrow V}$ (Eqn. (2)) is positive, and thus the regulation-detection score $\langle R_{W\rightarrow V}\rangle$ (Eqn. (4)) equals one, reflecting the positive regulation of W on V. The sign of $R^{t_W}_{V\rightarrow V}(t)$ (Eqn. (3)) changes, and thus $-1 < \langle R_{V\rightarrow V}\rangle < 1$ (Eqn. (4)), reflecting the mixture of positive and negative self-regulation of V. (D) Both $R^{t_W}_{V\rightarrow W}$ and $R^{t_V}_{W\rightarrow W}$ are negative, and thus $\langle R_{V\rightarrow W}\rangle = \langle R_{W\rightarrow W}\rangle = -1$, reflecting the negative regulation of V on W and the self-regulation of W. (E) Regulation-detection scores are calculated from the time-series population data of two bacteria: *Paramecium*, P (blue), and *Didinium*, D

(red) (data taken from (*10*)). Reflecting the known predatory interaction, $\langle R_{P\to D}\rangle = 1$ and $\langle R_{D\to P}\rangle = -1$. Furthermore, reflecting that the self-regulation of both P and D consists of both positive (i.e., birth) and negative (i.e., death) regulation,$\langle R_{P\to P}\rangle = 0.26$ and $\langle R_{D\to D}\rangle = 0.02$.

Network Inference method from oscillatory time series

If X positively (negatively) regulates Y, then the reflection score $\langle R_{X\to Y}\rangle = 1$ (resp., -1). In other words, $-1 < \langle R_{X\to Y}\rangle < 1$ indicates either a mixture of positive and negative regulation of X to Y or the absence of regulation. Thus, in the system where the interactions are not mixed (i.e., monotonic), such as gene regulation by a transcription factor and predator-prey relationships, $-1 < \langle R_{X\to Y}\rangle < 1$ indicates the absence of regulation. This can be used to infer network regulations from time-series data, as positive or negative regulation is present in the network only when $\langle R_{X\to Y}\rangle = 1$ or -1, respectively. Similarly, self-regulation, which is either positive or negative, is possible only when $\langle R_{Y\to Y}\rangle = 1$ or -1. However, since the degradation of molecules or the death rate of species typically increases as its own concentration increases, self-regulation can be assumed to be negative (i.e., $\langle R_{Y\to Y}\rangle = -1$). In this case, positive or negative regulation from X to Y is possible only when $\overrightarrow{R} = (\langle R_{X\to Y}\rangle, \langle R_{Y\to Y}\rangle) = (1, -1)$ or $(-1, -1)$, and thus, $\overrightarrow{R} \neq (\pm 1, -1)$ indicates the absence of regulation (Rule 1, Fig. 2A). Furthermore, we use $\overrightarrow{R} = (1, -1)$ or $(-1, -1)$ to infer positive or negative regulation (Rules 2 and 3, Fig. 2A). Note that, if positive or mixed self-regulation is possible, as in Fig. 1E, Rules 2 and 3 can be relaxed to $\langle R_{X\to Y}\rangle = 1$ and $\langle R_{X\to Y}\rangle = -1$, respectively.

We illustrate how the three rules (Fig. 2A) can infer a network using as an example the Kim-Forger model (Fig. 2B), a simple model describing the transcriptional negative feedback loop of the mammalian circadian clock (*49, 50*). In the model, the mRNA (M) is translated into the cytosolic protein (P_C). Then, P_C is transported to the nucleus and there P inhibits the transcription of M (*49,50*). To infer the network structure (Fig. 2B, bottom), we first compute $\overrightarrow{R}$ for each possible interaction and self-regulation pair (six in total) from the time series (Fig. 2B, top). Then, using Rule 1, three regulations are inferred as absent (Fig. 2B). Furthermore, Rules

2 and 3 successfully identify the two positive regulations ($M \rightarrow P_C$ and $P_C \rightarrow P$) and the one negative regulation ($P \dashv M$), which have $\overrightarrow{R} = (1, -1)$ and $\overrightarrow{R} = (-1, -1)$, respectively. This successfully infers the negative feedback loop structure (Fig. 2B). Using the same procedure, our method also successfully infers the *Frzilator* negative feedback loop, which models the signaling circuit of *Myxococcus xanthus* (*51*) (Fig. 2C and Table S1) and a 4-state Goodwin oscillator (*52*) (Fig. 2D and Table S2).

In fact, for the single negative feedback loop models, the order of peaks and nadirs of the time series matches with the order of regulation in the feedback loop (Fig. 2B-D). For instance, the peak of M is followed by the peaks of P_C and then P (Fig. 2B). This property has been used in previous algorithms to infer single negative feedback loop structures (*36–38*). Next, we test whether our method can be applied to a more challenging case when data are merged from two independent models, specifically the Kim-Forger (Fig. 2E; solid lines) and Goodwin (Fig. 2E; dashed lines) models. After merging the time-series data, the order of peaks and nadirs cannot be used to infer the network anymore. That is, if only the order of peaks is used for this example, a single negative feedback loop with seven components is inferred. However, as our method uses the whole data set rather than just the peaks and nadirs, it successfully infers the two independent underlying networks (Fig. 2E, right and Table S3). Moreover, our inference method also successfully infers a cyclic network with output variables, which also does not adhere to the single feedback loop structure (Fig. 2F and Table S4).

While our method successfully infers various networks, Rules 2 and 3 can make false-positive inferences as $\overrightarrow{R} = (\pm 1, -1)$ is a necessary condition for positive or negative regulation, and thus $\overrightarrow{R} = (\pm 1, -1)$ can occur even in the absence of regulation. We illustrate this using a simulated repressilator data set (Fig. 2G, left). The repressilator is a single feedback loop of genetic inhibition that consists of three mRNA (Fig. 2G, left; solid lines) and three proteins (Fig. 2G, left; dashed lines) (*53, 54*). The mRNA (M_i) are translated into the respec-

tive proteins (P_i), which then repress the transcription of the next gene (e.g., P_1 represses M_2). While our method recovers the correct interactions (Fig. 2G, right; solid arrows), it also incorrectly predicts negative regulation among the proteins (Fig. 2G, right; dashed arrows). These false-positive predictions are due to the similar shape and phase of the time-series data. For instance, the shape and phase of P_1 (solid blue line) is extremely close to the shape and phase of its mRNA, M_1 (dashed blue line), e.g., their phase difference is only 2.4% of the total period. Due to their similarity, our method cannot distinguish M_1 and P_1 and thus predicts that P_3 negatively regulates not only M_1 but also P_1. For the same reason, our method falsely predicts that P_1 negatively regulates P_2, and P_2 negatively regulates P_3 (Fig. 2G and Table S5). Taken together, we caution that, in the presence of nearly identical time series in a network, our method may infer false-positive regulations, which seems unavoidable for any inference methods using time-series data.

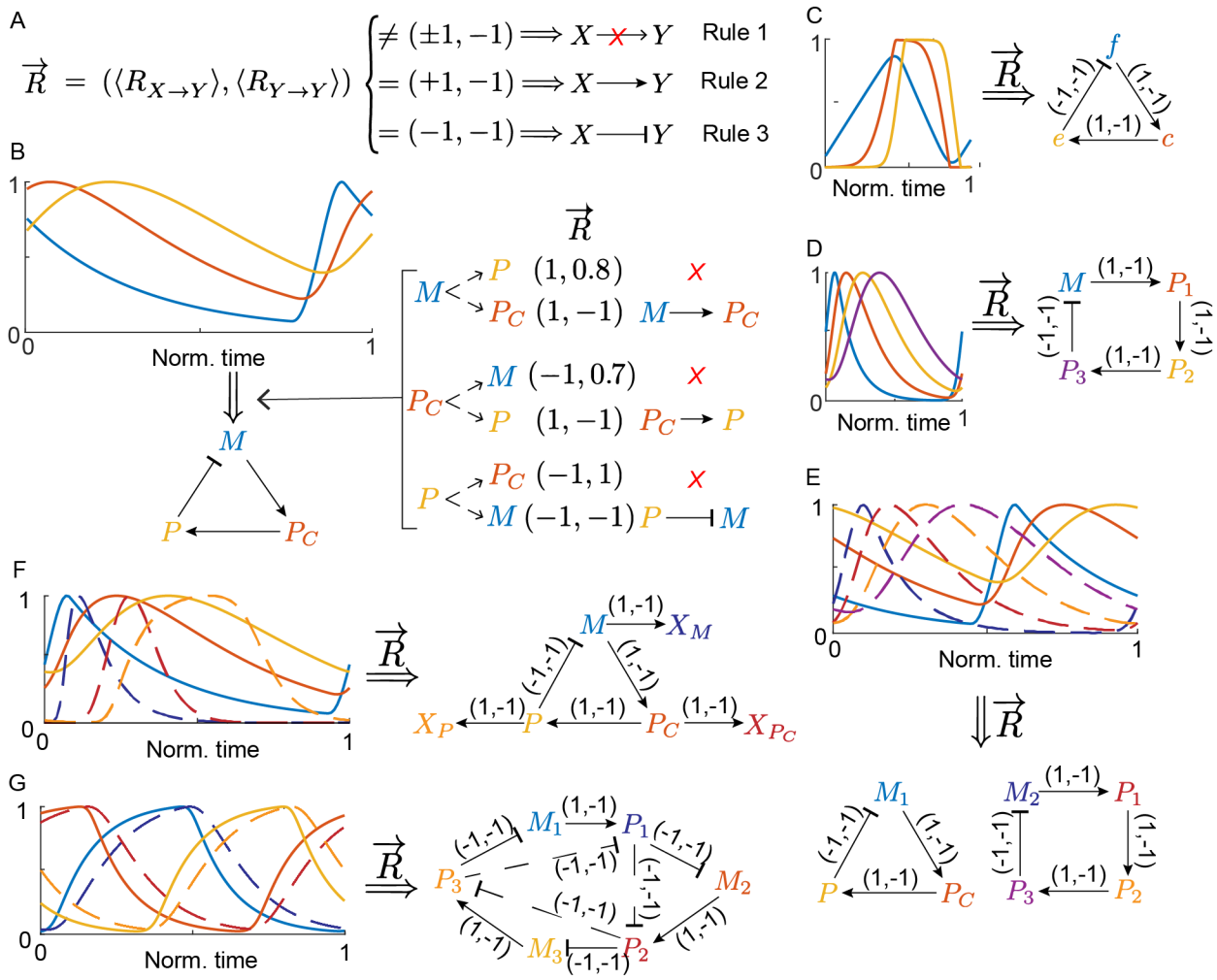


Figure 2: **The inference method successfully infers various *in silico* network structures**. (A) The three rules for network inference. $\overrightarrow{R} \neq (\pm 1, -1)$ indicates the absence of regulation and $\overrightarrow{R} = (1, -1)$ or $(-1, -1)$ indicates positive or negative regulation. (B) The three rules successfully infer the network structure of the Kim-Forger model from simulated time-series data. According to Rule 1, the three regulations $M \to P$, $P_C \to M$, and $P \to P_C$ are inferred as absent. According to Rules 2 and 3, the two positive regulations ($M \to P_C$ and $P_C \to P$) and the one negative regulation ($P \dashv M$), which have $\overrightarrow{R} = (1, -1)$ and $\overrightarrow{R} = (-1, -1)$, are inferred. (C-D) Our inference method also successfully infers the negative feedback loop of the *Frzilator* (C) and a 4-state Goodwin oscillator (D). (E-F) Our inference method also successfully infers correct regulations for more challenging cases beyond the single feedback loop structure, i.e., when time-series data are simulated with two independent models, the Kim-Forger model and Goodwin model (E) and an extended Kim-Forger model with output variables (F). (G) Our method also successfully infers regulations (solid arrows) of the repressilator from its three mRNA (solid lines) and three protein time-series data (dashed lines). However, our method also falsely predicts negative regulations among the proteins (dashed arrows) due to the

similar time series between an mRNA and its protein (e.g., M_1 and P_1). See Tables S1-S5 for the complete list of regulation-detection scores for (C)-(G) and Section 3 in Supplementary Information for the equations and parameters used to simulate the data.

Robustness of the inference method to interpolation error and noise

The calculation of $\overrightarrow{R}$, which is the key to the inference method, requires continuous time-series data. Typically, however, experimentally measured time-series data are sampled discretely. For instance, mRNA levels of circadian genes frequently are measured via PCR every three hours (*55*). For discrete data, our method uses interpolation to generate continuous data (see Methods). Accordingly, we test how sensitive our method is to interpolation error, specifically when linear interpolation is used, by using the five *in silico* data sets in Figs. 2B-F. That is, by decreasing the points measured per period from 10^2 to 10^1 (i.e., increasing the interpolation error), we quantify the accuracy of our network inference method with the score $\mathrm{F}_1 = \dfrac{\mathrm{TP}}{\mathrm{TP} + (\mathrm{FP} + \mathrm{FN})/2}$ (TP-the number of true positives, FP-false positives, and FN-false negatives). As F_1 is the harmonic mean of precision and recall, $\mathrm{F}_1 = 1$ and $\mathrm{F}_1 = 0$ indicate perfect recovery of the network and absence of correct inference, respectively. To account for interpolation error, we accept interactions based on three thresholds for $\langle R \rangle$ values: 0.99, 0.95, and 0.90. For example, a threshold of 0.99 means that we relax the condition $\overrightarrow{R} = (\pm 1, -1)$ up to ± 0.99, i.e, we accept any interaction that satisfies both $|\langle R_{X \to Y} \rangle| > 0.99$ and $\langle R_{Y \to Y} \rangle < -0.99$. We repeat this process 100 times, each time beginning the sample collection at a randomly selected time in the period (see Methods for details). Then, we investigate how the mean of the distribution of F_1 scores changes as the sampling rate decreases (Fig. 3A). For single negative feedback loops (i.e., Frzilator, Goodwin, Kim-Forger), our method accurately recovers the network even when the number of data points measured per period is relatively low, e.g., ten per cycle. For the more complicated models (i.e., the merged Goodwin and Kim-Forger and the Kim-Forger with outputs models), slightly more data points are required for inference at high accuracy. Furthermore, our method shows similar robustness across

the three thresholds, especially when the points sampled are toward the lower end.

Next, because experimental data includes noise, we test the sensitivity of our network inference procedure to noise (Fig. 3B) (see Methods for details). As we increase the level of the multiplicative noise added to the data set from 0 (no noise) to 10% multiplicative noise (sampled from $\mathrm{N}(0, 0.1^2)$), the F_1 scores decrease. In particular, the decrease occurs more dramatically when the threshold is 0.99, indicating that the high threshold leads to higher sensitivity to noise in the data. Moreover, this decrease in F_1 scores with the threshold of 0.99 is a result of an increase in false negatives (i.e., the exclusion of true interactions due to noise). Thus, we use a threshold of 0.9 when applying our inference method to experimental data (see below) as it leads to the most accurate results in the presence of noise (Fig. 3B). However, users have the option to adjust the threshold depending on the sampling rate and noise level of the data when using our computational package, ION (Fig. 4A) see Supplementary Information and Figs. S1 and S2 for a step-by-step manual).

Successful inferences from experimentally measured time series

As our inference method is quite robust to discrete data sampling and noise, we expect that our inference method can accurately infer network structures from experimentally measured time series as well. Indeed, when applied to experimentally measured abundances of the three repressilator proteins (*54*), our method successfully infers a three-gene repressilator network structure (Fig. 4D and Table S6). Note that our method recovers the repressilator network despite the absence of mRNA data because the shape and phase of the mRNA and protein profiles are expected to be similar, as in Fig. 2G, due to the short translation time in *E. coli* compared to the period (*56*). Moreover, we compare the results of our method with those of two popular model-free inference methods, Partial Cross Mapping (PCM) (*20*) and Granger Causality (GC) (*4*) (Fig. 4D). As these methods can only infer the presence of regulation, not its type (i.e., positive and negative), unlike our method, the arrows represent inferred regulations,

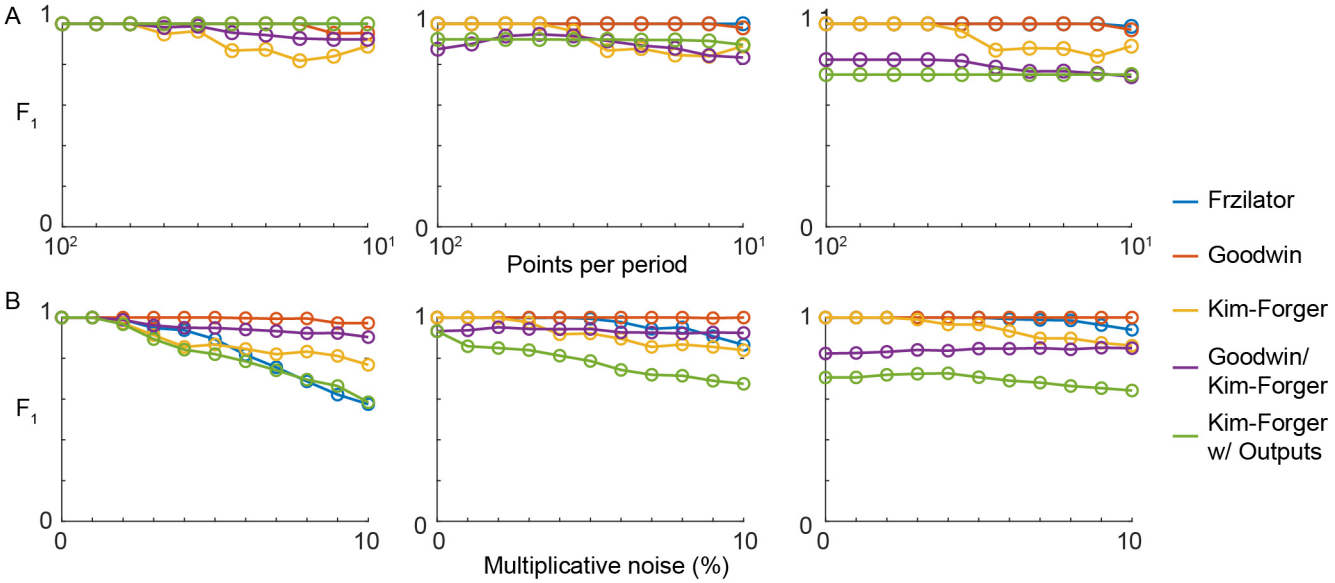


Figure 3: **Our network inference method is robust to interpolation error and to noise**. (A-B) The accuracy of our inference method when the number of points measured (A) and the level of noise (B) vary. Here, the points measured per period decreases from 10^2 to 10^1 (A) and the multiplicative noise increases from 0 to 10%, which is sampled from $\mathrm{N}(0, 0.1^2)$ (B). The mean of the F_1 score for 100 different time series, which are generated with randomly chosen phases (A) and noise levels (B), is plotted (see Methods for details). $\mathrm{F}_1 = 1$ and 0 indicate perfect recovery of the network and the absence of correct inference, respectively. Different thresholds for $\langle R \rangle$, 0.99 (left), 0.95 (middle), and 0.90 (right), are used.

which could be either positive or negative. The PCM method recovers two correct regulations, $P_2 \rightarrow P_1$ and $P_3 \rightarrow P_2$, but fails to recover the regulation $P_1 \rightarrow P_3$ and makes two false-positive predictions, $P_1 \rightarrow P_2$ and $P_3 \rightarrow P_1$. While the GC method infers all existing regulations, it makes two additional false-positive predictions, $P_1 \rightarrow P_2$ and $P_2 \rightarrow P_3$. Even for this simple three-node network, the popular model-free inference methods make false-positive predictions because the network components oscillate at the same period.

We compare the performance of our method with these model free-inference methods for a more challenging case when we combine two copies of the data set in Fig. 4D, one at the original phase and one with shifted phase (Fig. 4C and Table S7). From the combined time-series data, our method successfully infers two repressilator networks, whereas the PCM method infers two of the six correct regulations while also inferring four incorrect regulations (Fig. 4C). The GC method infers more correct regulations (five of the six); however, again, it suffers from several spurious regulations (15 false-positive interactions, Fig. 4C). Note that, even though we are using the same repressilator data set, there are inconsistencies in the PCM and GC results compared with those from Fig. 4D. These inconsistencies are a consequence of the shortened length of data used in Fig. 4C compared with that in Fig. 4D. This indicates that, in addition to the risk of false-positive inference, the PCM and GC methods are sensitive to the amount of data, unlike ours.

For time series measuring the amount of cofactors present at the estrogen-sensitive pS2 promoter after treatment with estradiol (data from (*57, 58*)), PCM and GC infer an almost fully connected network and a fully connected network, respectively (Fig. 4D). On the other hand, our method only infers two regulations, both supported by the current biological understanding of the system. That is, human ERα (hER) binds to the pS2 promoter after treatment with estradiol to recruit RNA Polymerase II to the promoter, supporting the inferred positive regulation of POLII by hER. Furthermore, TRIP1 acts as a surrogate measure for the 20S proteasome (APIS),

which promotes proteasome-mediated degradation of hER (*57*), supporting the inferred negative regulation of hER by TRIP1. However, the inferred network (Fig. 4D, Table S8) does not contain a negative feedback loop, which is required to generate sustained oscillations (*59*). Thus, there may be intermediate steps between POLII and TRIP1, TRIP1 and HDAC, and also HDAC and hER that form the negative feedback loop. Altogether, this illustrates that our method can identify direct regulations while highlighting connections that require further experimental investigation.

Discussion

We developed a model-based method that infers the network structure underlying biological oscillators. The method identifies positive or negative regulation by testing whether given oscillatory time-series data are reproducible with a general mechanistic ODE model (Eqn. (1)). In this way, our method successfully and efficiently inferred several network architectures such as single cycles (e.g., repressilator), two independent cycles, and a cycle structure with outputs. Furthermore, we provide a user-friendly computational package, ION, that applies to discrete and noisy data to infer networks of biological components that oscillate from the molecular to the population level. Our method can uncover unknown functional relationships and mechanisms that drive oscillatory behavior in biological systems when it is incorporated with evolving experimental time-series measurement methods.

Our method merges the advantages of model-based and model-free methods while mitigating the drawbacks of each. In particular, our model-based inference method does not suffer from the serious risk of false-positive prediction for biological oscillators or sensitivity to the amount of data unlike the previous model-free inference methods such as GC and PCM (Fig. 4). However, as our inference method is model-based, it runs the risk that the imposed ODE model and functional relationships create false representations of the true interactions (*21*). Our

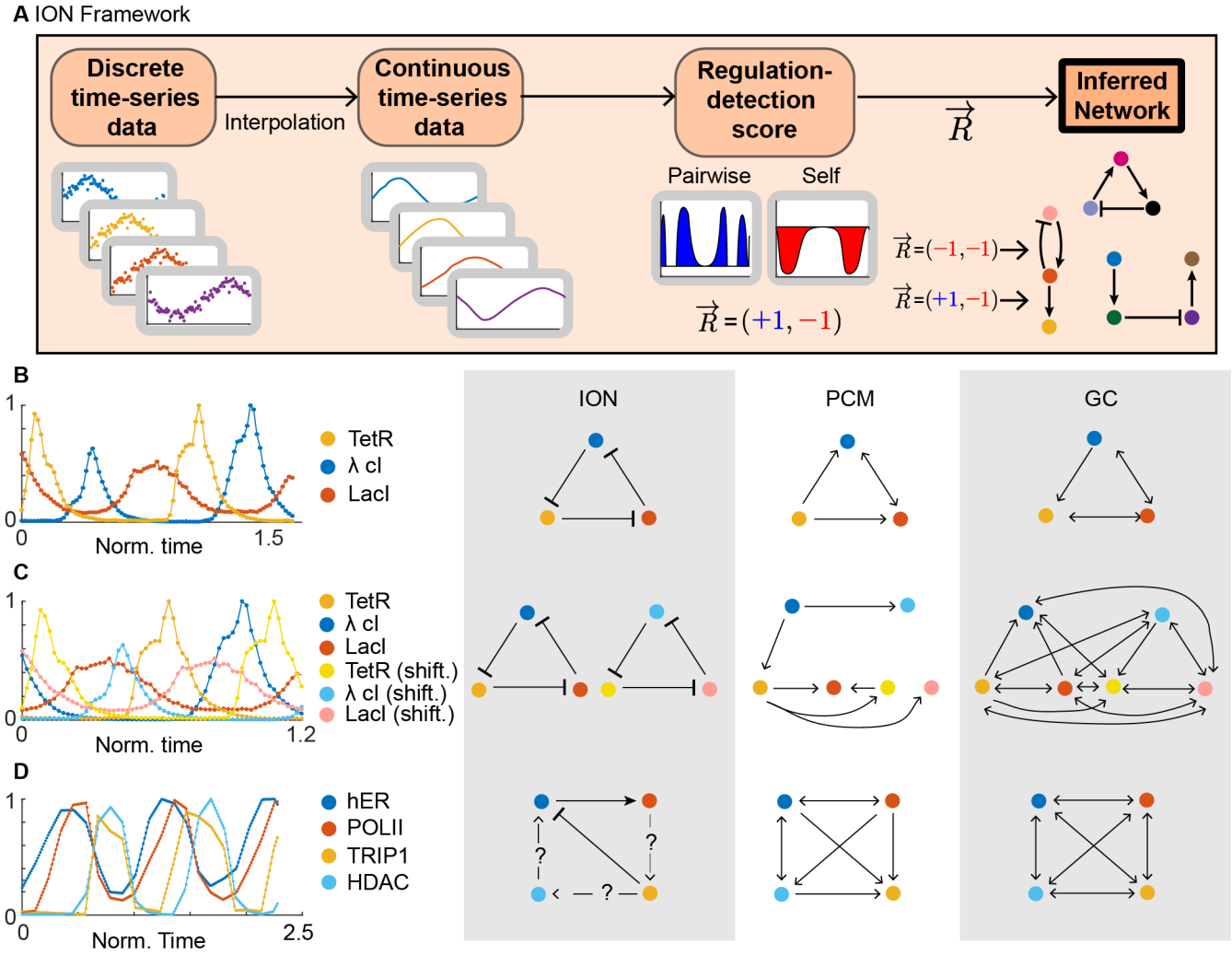


Figure 4: **The computational package ION successfully infers networks of repressilator and pS2 promoter cofactors**. (A) The ION package interpolates and smooths noisy and discrete time-series data and then computes the pairwise and self regulation-detection functions and scores ($\vec{R}$) for every pair of components. If $\vec{R}$ satisfies Rules 2 or 3 (Fig. 2A), which can be relaxed depending on the threshold specified by the user, positive or negative regulation is assigned. See Supplemental Information for a comprehensive manual. (B) Using experimentally measured oscillatory time series from (*54*), ION successfully infers a three-gene repressilator network structure. On the other hand, two popular model-free inference methods, PCM and GC, infer several false-positive regulations (e.g., λ cI regulates LacI). (C) ION also successfully infers two independent cycle when the experimental repressilator data set from (A) is duplicated and the phase is shifted by about half of the period. However, PCM and GC fail in infring independent cycles due to false-positive predictions. (D) ION infers two direct regulations and predict three hidden regulations among cofactors at the estrogen-sensitive pS2 promoter after estradiol treatment (*57*).

method minimizes this risk by using the most general form of an ODE (Eqn. (1)) to model the change in a component that is acted upon by another component and itself. In this way, we resolve the limitations of previous model-based methods that restricted the class of models, such as separable synthesis and degradation functions (*39, 41, 45*), specific types of functions (e.g., power or Hill functions) (*31, 39*), and a single negative feedback loop structure (*36–38*). Thus, we were able to uncover several varying network structures. While we considered the most general form of an ODE (Eqn. (1)) that describes the interactions between two components, an interesting future direction would be to extend our work to models that describe the interactions among multiple oscillatory components, e.g, $\frac{\mathrm{d}Y}{\mathrm{d}t} = f(X_1, ..., X_n, Y)$.

Methods

ION (Inferring Oscillatory Networks) computational package

We provide user-friendly MATLAB code (a Github repository link will be provided upon acceptance of the manuscript). The ION package can be used to infer the network structure of oscillators, which are described by Eqn. (1), across all levels of biology. Here, we briefly describe the key steps of the ION package (see Supplementary Information for a comprehensive manual).

Reflection times

For each time point t_i of the given time series $X(t) = (X(t_1), X(t_2), ..., X(t_n))$, first, the reflection time t_{iX} needs to be calculated (Fig. 1B). That is, we seek the time point t_{iX} such that $X(t_i) = X(t_{iX})$ and the signs of the slopes at $X(t_i)$ and $X(t_{iX})$ are opposite (i.e., rising and falling phase). For this, the discrete $X(t)$ is interpolated to a continuous time series $F_X(t)$ with either the 'linear' or 'fourier' interpolation method, chosen by the user. Then, t_{iX} is estimated

by identifying the closest time point to t_i among time points t satisfying the following equation:

$$F_X(t) = X(t_i) \quad \text{and} \quad \text{sign}(F'_X(t)) \neq \text{sign}(F'_X(t_i)).$$

Regulation-detection function and score

Using the estimated t_{iX}, we compute the regulation-detection function, e.g., $R^{t_{iX}}_{Y\to X}(t_i)$, for each time point t_i as follows:

$$(Y(t_{iX}) - Y(t_i))(\dot{X}(t_{iX}) - \dot{X}(t_i)).$$

If the linear method is chosen, $Y(t_{iX})$ is linearly interpolated based on the data $(Y(t_1), \ldots, Y(t_n))$, and $\dot{X}(t) = (\dot{X}(t_1), \ldots, \dot{X}(t_n))$ is estimated using a moving slope filter method. Specifically, after fitting a low-order polynomial regression model to $X(t) = (X(t_1), X(t_2), ..., X(t_n))$ in a sliding window (*60*), the derivative of the polynomial fit is used to estimate $\dot{X}(t)$ and then $\dot{X}(t_{iX})$ is linearly interpolated based on the estimated $\dot{X}(t)$. The model order and the length of the sliding window parameters can be adjusted (see Supplementary Information). On the other hand, if the fourier method is chosen, both $\dot{X}(t_i)$ and $\dot{X}(t_{iX})$ are estimated as $\dot{F}_X(t_i)$ and $\dot{F}_X(t_{iX})$, respectively, and similarly, $Y(t_i)$ and $Y(t_{iX})$ are estimated as $F_Y(t_i)$ and $F_Y(t_{iX})$, respectively, where $F_Y(t)$ is the Fourier series fit to the data $Y(t)$. Finally, in both cases, the regulation-detection score Eqn. (4) is estimated using the MATLAB function `trapz`.

Time-series data

We simulate *in silico* data using the MATLAB function `ode23tb` (Fig. 2). See Supplementary Information for the model equations and parameters. The experimental data sets of the repressilator (Fig. 4D) were obtained from (*54*). Next, to generate the duplicated experimental repressilator data set (Fig. 4C), we mixed two copies of the repressilator data set from Fig. 4D. We kept one copy at the original phase and, for the second copy, we shifted the phase by 115 minutes (almost half of the period) (Fig. 4C). Then, we removed data on the left and the right

where there was only coverage of one of the two data sets. We obtained the estradiol data set from (*57, 58*) and the Paramecium/Didinium data from (*10*).

Discrete and noisy data

To generate discretely sampled data (Fig. 3A), we select a random point in the first period to begin data extraction, and then we uniformly sample two periods worth of data at a sampling rate of 100 points per period. We repeat this process 100 times–every time randomly initializing the starting point in the first period–to generate 100 distinct data sets for every model. Then, we run our algorithm and compute F_1 scores for each of the 100 data sets. Next, from each of the 100 generated data sets, we take every other data point to reduce the number of data points (e.g., $50, 33, 25, \ldots, 10$ per period).

For the multiplicative noise analysis (Fig. 3B), we begin with two periods worth of data sampled at 100 points per period. Then, we add multiplicative noise sampled randomly from a normal distribution with mean 0 and standard deviation given by the percentage. For example, at 10% multiplicative noise, we add the noise $X(t_i) \cdot \epsilon$ to $X(t_i)$, where ϵ is sampled randomly from $\mathrm{N}(0, 0.1^2)$.

PCM and GC

We ran the PCM method with an embedding dimension of 3, $\tau = 1$, and a max delay of 3, and used a threshold of 0.5684 as recommended in (*20*) by using the code provided in (*20*). We ran the GC using the code provided in (*61*) and specified a max delay of 3 as we did with the PCM method and a significance level of 95%. We rejected the null hypothesis that Y does not Granger cause X, and thereby inferred direct regulations if the value of the F-statistic was greater than the critical value from the F-distribution (*4*).

Acknowledgments. We thank Anne Shiu and Seokjoo Chae for valuable comments. This work

was supported by a National Institutes of Health Training Grant (T32 HL007622) and Samsung Science and Technology Foundation under Project Number SSTF-BA1902-01 (to J.K.K.).

Competing Interests. The authors declare that they have no competing interests.

Author Contributions. JT, DF, and JKK designed the research. JT and JKK conceptualized the theoretical approach, performed the research, and developed the computational platform. JT, DF, and JKK wrote the manuscript.

Competing Interests. The authors declare that they have no competing interests.

References

1. M. M. Saint-Antoine, A. Singh, *Current Opinion in Biotechnology* **63**, 89 (2020). Nanobiotechnology Systems Biology.

2. R. Zhang, N. F. Lahens, H. I. Ballance, M. E. Hughes, J. B. Hogenesch, *Proceedings of the National Academy of Sciences* **111**, 16219 (2014).

3. L. S. Mure, *et al.*, *Science* **359** (2018).

4. C. W. J. Granger, *Econometrica* **37**, 424 (1969).

5. P. A. Stokes, P. L. Purdon, *Proceedings of the National Academy of Sciences* **114**, E7063 (2017).

6. H. Lütkepohl, *New Introduction to Multiple Time Series Analysis* (Springer, 2005).

7. A. Pourzanjani, E. D. Herzog, L. R. Petzold, *PLOS ONE* **10**, 1 (2015).

8. J. H. Abel, *et al.*, *Proceedings of the National Academy of Sciences* **113**, 4512 (2016).

9. A. C. Yang, C.-K. Peng, N. E. Huang, *Nature Communications* **9**, 3378 (2018).

10. G. Sugihara, *et al.*, *Science* **338**, 496 (2012).

11. E. R. Deyle, G. Sugihara, *PLOS ONE* **6**, 1 (2011).

12. E. R. Deyle, *et al.*, *Proceedings of the National Academy of Sciences* **110**, 6430 (2013).

13. A. A. Tsonis, *et al.*, *Proceedings of the National Academy of Sciences* **112**, 3253 (2015).

14. H. Ye, E. R. Deyle, L. J. Gilarranz, G. Sugihara, *Scientific Reports* **5**, 14750 (2015).

15. E. R. Deyle, M. C. Maher, R. D. Hernandez, S. Basu, G. Sugihara, *Proceedings of the National Academy of Sciences* **113**, 13081 (2016).

16. H. Ma, *et al.*, *Phys. Rev. E* **96**, 012221 (2017).

17. J. Runge, *et al.*, *Nature Communications* **10**, 2553 (2019).

18. N. Tani, *et al.*, *Scientific Reports* **10**, 650 (2020).

19. J.-Y. Wang, T.-C. Kuo, C.-h. Hsieh, *Nature Communications* **11**, 2635 (2020).

20. S. Leng, *et al.*, *Nature Communications* **11**, 2632 (2020).

21. S. Cobey, E. B. Baskerville, *PLOS ONE* **11**, 1 (2016).

22. P. Mhaskar, M. A. Hjortsø, M. A. Henson, *Biotechnology Progress* **18**, 1010 (2002).

23. E. Balsa-Canto, M. Peifer, J. R. Banga, J. Timmer, C. Fleck, *BMC Systems Biology* **2**, 26 (2008).

24. N. Radde, L. Kaderali, *Discrete Applied Mathematics* **157**, 2285 (2009). Networks in Computational Biology.

25. T. Toni, D. Welch, N. Strelkowa, A. Ipsen, M. P. Stumpf, *Journal of The Royal Society Interface* **6**, 187 (2009).

26. N. Geva-Zatorsky, E. Dekel, E. Batchelor, G. Lahav, U. Alon, *Proceedings of the National Academy of Sciences* **107**, 13550 (2010).

27. G. Lillacci, M. Khammash, *PLOS Computational Biology* **6**, 1 (2010).

28. B. Wang, W. Enright, *SIAM Journal on Scientific Computing* **35**, A2718 (2013).

29. M. Stražar, M. Mraz, N. Zimic, M. Moškon, *Natural Computing* **13**, 119 (2014).

30. D. Trejo Banos, A. J. Millar, G. Sanguinetti, *Bioinformatics* **31**, 3617 (2015).

31. T. Gotoh, *et al.*, *Proceedings of the National Academy of Sciences* (2016).

32. D. McBride, L. Petzold, *Journal of Biological Rhythms* **33**, 515 (2018).

33. S. Wang, *et al.*, *Proceedings of the National Academy of Sciences* **115**, 9300 (2018).

34. T. Firman, A. Amgalan, K. Ghosh, *The Journal of Physical Chemistry B* **123**, 343 (2019).

35. J. A. Pitt, J. R. Banga, *BMC Bioinformatics* **20**, 82 (2019).

36. S. Pigolotti, S. Krishna, M. H. Jensen, *Proceedings of the National Academy of Sciences* **104**, 6533 (2007).

37. S. Pigolotti, S. Krishna, M. H. Jensen, *Phys. Rev. Lett.* **102**, 088701 (2009).

38. M. H. Jensen, S. Pigolotti, S. Krishna, *The European Physical Journal Special Topics* **178**, 45 (2009).

39. T. Konopka, M. Rooman, *BMC Systems Biology* **4**, 123 (2010).

40. T. Konopka, *Bioinformatics* **27**, 961 (2011).

41. J. K. Kim, D. B. Forger, *SIAM Journal on Applied Mathematics* **72**, 1842 (2012).

42. S. L. Brunton, J. L. Proctor, J. N. Kutz, *Proceedings of the National Academy of Sciences* **113**, 3932 (2016).

43. N. M. Mangan, S. L. Brunton, J. L. Proctor, J. N. Kutz, *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* **2**, 52 (2016).

44. K. A. McGoff, *et al.*, *Genome Biology* **17**, 214 (2016).

45. H.-H. Jo, *et al.*, *Communications Biology* **1**, 207 (2018).

46. R. FitzHugh, *Biophysical Journal* **1**, 445 (1961).

47. J. Nagumo, S. Arimoto, S. Yoshizawa, *Proceedings of the IRE* **50**, 2061 (1962).

48. B. Veilleux, The analysis of a predatory interaction between didinium and paramesium, Master's thesis, University of Alberta (1976).

49. J. K. Kim, D. B. Forger, *Molecular Systems Biology* **8**, 630 (2012).

50. J. K. Kim, *IET Syst. Biol.* **10**, 125 (2016).

51. O. A. Igoshin, A. Goldbeter, D. Kaiser, G. Oster, *Proceedings of the National Academy of Sciences* **101**, 15760 (2004).

52. B. C. Goodwin, *Advanced Enzyme Regulation* **3**, 425 (1965).

53. M. B. Elowitz, S. Leibler, *Nature* **403**, 335 (2000).

54. L. Potvin-Trottier, N. D. Lord, G. Vinnicombe, J. Paulsson, *Nature* **538**, 514 (2016).

55. H. R. Ueda, *et al.*, *Nature Genetics* **37**, 187 (2005).

56. B. Choi, *et al.*, *Bioinformatics* **36**, 586 (2019).

57. R. Métivier, *et al.*, *Cell* **115**, 751 (2003).

58. V. Lemaire, C. F. Lee, J. Lei, R. Métivier, L. Glass, *Phys. Rev. Lett.* **96**, 198102 (2006).

59. B. Novák, J. J. Tyson, *Nature Reviews Molecular Cell Biology* **9**, 981 (2008).

60. A. V. Oppenheim, R. W. Schafer, J. R. Buck, *Discrete-time Signal Processing*, Technology and Engineering (Prentice Hall, 1999).

61. Chandler, Granger causality test (2020).

Supplementary materials

Supplementary Text

Figs. S1 and S2

Tables S1 to S8

References

Supplemental Information for Inferring causality in biological oscillators

Jonathan Tyler, Daniel Forger, Jae Kyoung Kim*

*To whom correspondence should be addressed; E-mail: jaekkim@kaist.ac.kr

Contents

1 Theoretical Foundations **1**

2 Manual for the ION computational package **3**

3 Description of the *in silico* models **8**

3.1 Kim-Forger Model 8

3.2 Frzilator 10

3.3 Goodwin Oscillator 10

3.4 Kim-Forger Model with Outputs 11

3.5 Repressilator 11

4 Complete results of *in silico* and experimental network inference (Figs. 2 and 4) **12**

1 Theoretical Foundations

Here, we describe the theoretical foundations of our main result. In particular, we formally define the reflection time t_Y (Fig. 1B) and prove that, if X positively regulates Y, then the

regulation-detection score $\langle R_{X\to Y}\rangle = 1$. Similar arguments can be used for negative regulations.

Definition 1. *If $Y(t)$ is a smooth time series with one maximum (t_M) and one minimum (t_m) over a period, then for $t \neq t_M, t_m$, we define the reflection time t_Y such that $Y(t_Y) = Y(t)$ and $t_Y \neq t$. If $t = t_M, t_m$, then $t_Y := t$.*

Theorem 1. *Let $X(t)$ and $Y(t)$ be smooth time series with period τ that have one maximum and one minimum over a period τ and solve the ODE*

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = f(X, Y). \tag{1}$$

If f is strictly monotonically increasing in X (i.e., X positively regulates Y), then $R^{t_Y}_{X\to Y}(t) = X^{t_Y}_d(t) \cdot \dot{Y}^{t_Y}_d(t) \geq 0$ for all t, where

$$X^{t_Y}_d(t) = X(t) - X(t_Y)$$

$$\dot{Y}^{t_Y}_d(t) = \dot{Y}(t) - \dot{Y}(t_Y).$$

Moreover, $\langle R_{X\to Y}\rangle = 1$, where

$$\langle R_{X\to Y}\rangle := \frac{\int_0^\tau R^{t_Y}_{X\to Y}(t)\mathrm{d}t}{\int_0^\tau |R^{t_Y}_{X\to Y}(t)|\mathrm{d}t}.$$

Proof. First,

$$\begin{aligned} R^{t_Y}_{X\to Y}(t) &= X^{t_Y}_d(t) \cdot \dot{Y}^{t_Y}_d(t) \\ &= (X(t) - X(t_Y))(\dot{Y}(t) - \dot{Y}(t_Y)) \\ &= (X(t) - X(t_Y))\cdot \\ &\qquad (f(X(t), Y(t)) - f(X(t_Y), Y(t_Y))) \geq 0. \end{aligned}$$

The last inequality holds because, by assumption, $f(X, Y)$ is a monotonically increasing function of X when Y has the same value (i.e., $Y(t) = Y(t_Y)$). Furthermore, $R^{t_Y}_{X\to Y}(t)$ cannot be 0 for all t. For the sake of contradiction, assume that $R^{t_Y}_{X\to Y}(t) = 0$ for all t. If $R^{t_Y}_{X\to Y}(t) = X^{t_Y}_d(t) \cdot \dot{Y}^{t_Y}_d(t) = 0$, then $X^{t_Y}_d(t) = 0$ or $\dot{Y}^{t_Y}_d(t) = f(X(t), Y(t)) - f(X(t_Y), Y(t_Y)) = 0$. If

$f(X(t), Y(t)) - f(X(t_Y), Y(t_Y)) = 0$, then $X(t) - X(t_Y) = 0$ as well because f is strictly monotonically increasing in X. Therefore, if $R^{t_Y}_{X\to Y}(t) = 0$ for all t, then

$$X(t) = X(t_Y) \quad \text{for all } t \tag{2}$$

Next, from (1) and (2),

$$\dot{Y}(t) = f(X(t), Y(t)) = f(X(t_Y), Y(t_Y)) = \dot{Y}(t_Y), \tag{3}$$

for all t. This implies that $\dot{Y}(t) = \dot{Y}(t_Y) = 0$ given the nature of t_Y (e.g, if Y is increasing at t, then Y is decreasing at t_Y). Therefore, $\dot{Y}(t) = 0$ for all t, i.e., Y is constant. However, this contradicts the assumption that Y is a smooth oscillating time series. Since $R^{t_Y}_{X\to Y}(t) \geq 0$ and is a nonzero function, $\langle R_{X\to Y}\rangle = 1$.

□

2 Manual for the ION computational package

As described in the main text, we developed a computational package, ION (Inferring Oscillatory Networks), to infer networks of biological oscillators (Fig. 4A). We have published the ION package in MATLAB at the link: (a Github repository will be provided upon acceptance of the manuscript). ION can be used as follows.

1. *Generate the data.csv file*. The first column of the data.csv file should be the time points at which the measurements were taken. The subsequent columns should be the data for each variable at the respective time points. If the data are stored in an excel spreadsheet, order the file as time points in column 1, data for variable 1 in column 2, data for variable 2 in column 3, etc., and then save it as a .csv file in the same directory as the code. Be sure that there are no variable names at the top of the data.csv file (see Input 1 in Fig. S1 for the structure of the data.csv file).

2. *Update the* `regulation_detection_inputs.m` *file*. Users need to specify the threshold used to infer regulation and the data interpolation method, in particular, either 'linear' or 'fourier'. The linear method will linearly interpolate the discrete data given in the data.csv file to create a continuous data set. The fourier method will fit a Fourier series to the discrete data to create a continuous data set.

   (a) *Update the threshold variable* (Input 2 in Fig. S1). The variable *threshold* determines the threshold for the $\overrightarrow{R}$ values accepted as interactions. For example, a threshold of 0.99 means that we relax the condition $\overrightarrow{R} = (\pm 1, -1)$ (Fig. 2A) up to ± 0.99, i.e, we accept any interaction that satisfies both $|\langle R_{X \to Y} \rangle| > 0.99$ and $\langle R_{Y \to Y} \rangle < -0.99$. Based on our analysis in Fig. 3, we recommend a threshold of 0.9, which we use for inferring interactions given experimental data in Fig. 4.

   (b) *If linear method is chosen, update the variables 'supportlength' and 'modelorder'* (Input 2 in Fig. S1). They need to be chosen so that the function `movingslope.m` accurately estimates the derivatives at each time point of the data set (*1*). The variable 'supportlength' determines the number of points used for the moving window average estimation of the derivative, so it should be at least 2 and no more than the number of data points. If 'supportlength' is odd, then the derivative is estimated using a central sliding window. If 'supportlength' is even, then the moving window will be moved backward by one element. For example, a 'supportlength' of 2 is equal to a difference quotient, computed for instance using the MATLAB function `diff`. Next, the variable 'modelorder' defines the order of the windowed model used to estimate the slope. If 'model order' is 1 or less than the 'supportlength'-1, then the model is linear or a regression, respectively. If 'modelorder' is equal to 'supportlength'-1, then the method will be a sliding Lagrange interpolant.

(c) *If fourier method is chosen, update the 'num_fourier' variable* (Input 2 in Fig. S2). The variable 'num_fourier' determines the number of Fourier series coefficients and must be at least one and at most 8. For example, if 'num_fourier' $= 1$, then the algorithm fits the parameters a_0, a_1, b_1, and ω to the data for each variable based on the Fourier series of the form $F(t) = a_0 + a_1 \cos(2\pi \cdot \omega \cdot t) + b_1 \sin(2\pi \cdot \omega \cdot t)$. The algorithm fits the Fourier series with the specified order using a nonlinear least-squares method, specifically using the Levenberg-Marquardt method. To manually adjust the Fourier fitting method and options (e.g., the method, function tolerance, etc.), users need to update the function `createFit.m`.

3. *If linear method is chosen, create extrema value files for each variable and save them in the same directory as the scripts* (Input 3 in Fig. S1). If the fourier method is chosen, the interpolated time series has one maximum and minimum per cycle. However, if the linear method is chosen, local extrema can occur due to noise. To distinguish such local extrema from the global extrema per cycle, which is required to compute the reflection time (see Fig. 1B), users need to input extrema value files for each variable in the data set. The files should be saved as ek.csv where k corresponds to the variable. For example, the extrema value file for variable 1 (the first data column) should be saved as e1.csv. The extrema value files should have time points in the first column and either 1 or -1 in the second column for a global maximum and a global minimum in that period at the time point, respectively. Users can generate these files by using the MATLAB function `findpeaks`. The function `findpeaks` returns all local minima and maxima, which users can use to decide the global minimum and maximum in each period.

4. *Run the* `main.m` *function*. The `main.m` function will read the inputs from the `regulation_detection_inputs.m` file and run the algorithm with the user-specified

method. If the linear (fourier) method was chosen, the `main.m` function calls the `main_linear.m` (`main_fourier.m`) script to run the algorithm. Users will see several figures being generated as well as several output files generated in a new "Outputs" directory, which will be created within the same directory that the script is run:

(a) If the linear method is chosen, a derivative.fig file that plots the estimated derivative values for all variables (Output 1 in Fig. S1).

(b) If the fourier method is chosen, files labeled Var$k$.fig, where k corresponds to the variable number, showing the fourier fits for each variable (Output 1 in Fig. S2).

(c) Figures plotting the regulation-detection functions for each possible interaction (Output 2 in Fig. S1 and S2). For example, the file Reg_detect_i_onto_i_fix_j.fig plots the regulation-detection function $R^{t_{V_j}}_{V_i \to V_i}(t)$ (see Fig. 1C), where V_i is variable i. Similarly, the file Reg_detect_i_onto_j.fig plots the regulation-detection function $R^{t_{V_j}}_{V_i \to V_j}(t)$ (see Fig. 1C).

(d) A .mat file with the regulation-detection scores for each possible self-regulation ('R_self' variable) and cross-regulation ('R_cross' variable) (Output 3 in Fig. S1 and S2).

(e) A figure plotting the inferred network structure, inferred_network_graph.fig (Output 4 in Fig. S1 and S2). If R_self$(i, j) < -$threshold and —R_cross$(i, j)| >$ threshold, then the algorithm infers the interaction from Variable j to Variable i. Moreover, if R_cross(i, j) is positive (negative), then it is a positive (negative) regulation.

(f) A Final_results.pdf document that reports each output listed above for reference.

Input 1 - data.csv

Time	Var 1	Var 2	Var 3
0,	0.01,	0.58,	0.10
5,	0.01,	0.54,	0.27
10,	0.012,	0.48,	0.43
15,	0.012,	0.42,	0.70
20,	0.012,	0.36,	0.93
25,	0.013,	0.32,	0.88
⋮	⋮	⋮	⋮

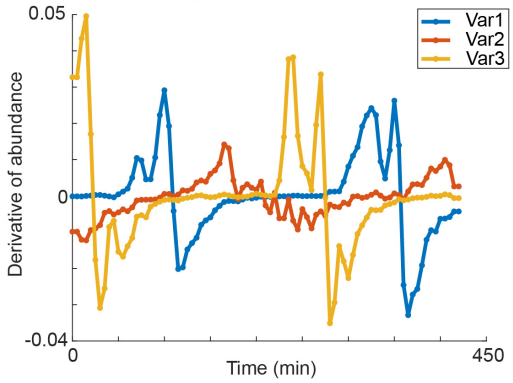
Input 2 - regulation_detection_inputs.m

```
% Regulation detection inputs

threshold = 0.9; % The threshold used to infer direct
% regulations.

method = 'linear'; %For linear interpolation or fourer
% interpolation.  Please choose either 'linear' or 'fourier'.

supportlength = 3; % Defines the number of points used
% for the moving window. Update when using the
% linear method.

modelorder = 1; % Defines the order of the windowed
% model used to estimate the derivative.
% Update when using the linear method.

num_fourier = 8; % If the 'fourier' method is chosen
% above, then this is the number of fourier series
% coefficients to fit.  Update when using fourier.
```

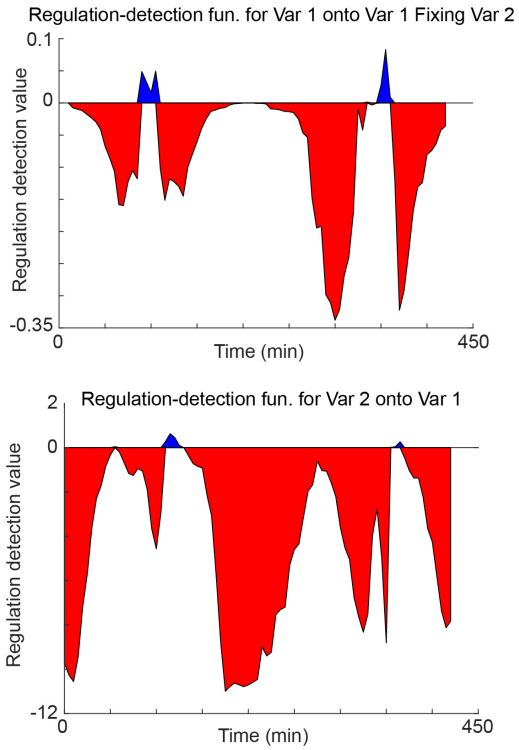
Input 3 - Extrema Value Files

e1.csv		e2.csv		e3.csv	
50,	-1	95,	-1	20,	1
110,	1	210,	1	195,	-1
270,	-1	365,	1	285,	1
355,	1				

Output 1 - Derivative



Output 2 - Regulation-detection functions



Output 3 - Regulation-detection scores

Self-Regulation scores			Regulation Scores		
-,	-0.92,	-0.006	-,	-0.99,	0.21
0.71,	-,	-0.98	0.75,	-,	-1
-0.93,	-0.04,	-	-1,	0.98,	-

Output 4 - Inferred Network

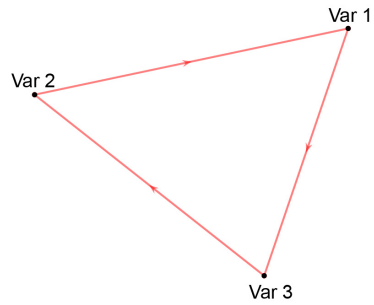


Figure S1: **Sample input and output files for the ION package if the linear method is chosen based on the experimental repressilator example (Fig. 4B)**. The data.csv (Input 1) file contains the time points in the first column and the data for each variable in subsequent columns, separated by commas. The regulation_detection_inputs.m file (Input 2) is already in the directory and should be updated according to the preferences of the user. If the user selects the 'linear' method, then the user must choose appropriate values for the 'supportlength' and

'modelorder' variables as well as generate extrema value files that list the time points at which the global maxima (1) and minima (-1) happen for each period (Input 3). The algorithm outputs the estimated derivatives for each variable (Output 1), plots of the regulation-detection functions (Output 2), the regulation-detection scores (Output 3), and the inferred network structure (Output 4).

3 Description of the *in silico* models

3.1 Kim-Forger Model

The Kim-Forger model describes the core transcriptional-translational feedback loop in the molecular circadian clock (Fig. 2B) (*2*). It is similar to the Goodwin model in that a protein product represses the transcription of its *mRNA*. However, the repression mechanism is not a Hill-type repression mechanism as in the Goodwin oscillator, but rather a protein sequestration mechanism where tight binding of the activator and repressor sequesters the activator protein, which in turn represses its activity. The model is given by the following system of ODEs.

$$\begin{aligned}
\frac{\mathrm{d}M}{\mathrm{d}t} &= k_1 f(P) - k_2 M \\
\frac{\mathrm{d}P_C}{\mathrm{d}t} &= k_1 M - k_3 P_C \\
\frac{\mathrm{d}P}{\mathrm{d}t} &= k_1 P_C - k_4 P \\
f(z) &= \frac{A_t - K_d - z + \sqrt{(A_t - K_d - P)^2 - 4A_t K_d}}{2A_t}.
\end{aligned}$$

Here, the variable M is the repressor mRNA concentration; P_C is the cytosolic repressor protein concentration; P is the nuclear repressor protein concentration. The parameter A_t is the total activator concentration, and K_d is the dissociation constant between the activator and repressor (closer to 0 means tighter binding). As in (*3*), we simulate the Kim-Forger model with the parameters $k_1 = 1$, $k_2 = 0.16$, $k_3 = 0.29$, $k_4 = 0.3$, $A_t = 0.6$, and $K_d = 10^{-5}$.

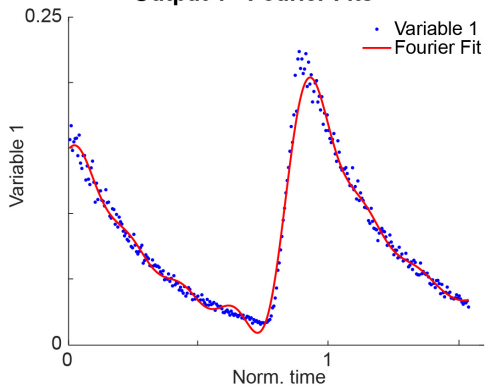
Input 1 - data.csv

Time	Var 1	Var 2	Var 3
0.0051,	0.156,	0.436,	0.849
0.0103,	0.167,	0.447,	0.919
0.0154,	0.148,	0.439,	0.968
0.0206,	0.155,	0.485,	0.905
0.0257,	0.157,	0.435,	0.879
0.0308,	0.158,	0.485,	0.969
0.0359,	0.144,	0.485,	0.937
0.0411,	0.160,	0.470,	0.947
0.0463,	0.138,	0.481,	1.033
⋮	⋮	⋮	⋮

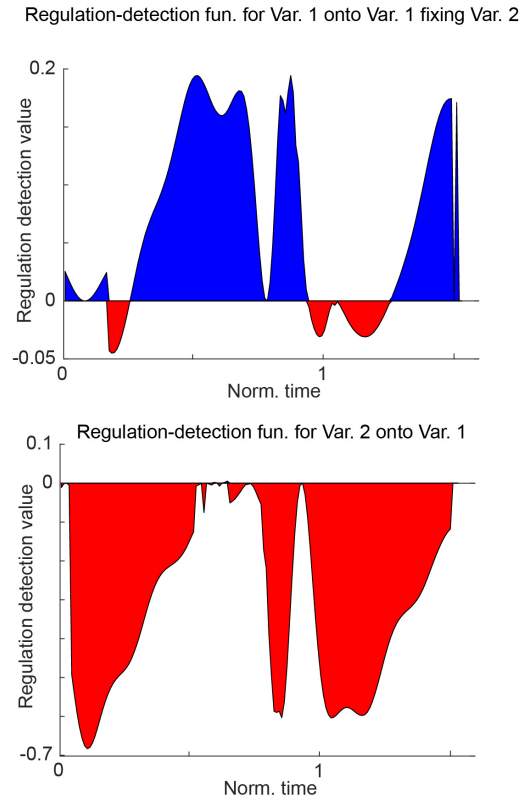
Input 2 - regulation_detection_inputs.m

```
% Regulation detection inputs

threshold = 0.9; % The threshold used to infer direct
% regulations.

method = 'linear'; %For linear interpolation or fourer
% interpolation.  Please choose either 'linear' or 'fourier'.

supportlength = 3; % Defines the number of points used
% for the moving window. Update when using the
% linear method.

modelorder = 1; % Defines the order of the windowed
% model used to estimate the derivative.
% Update when using the linear method.

num_fourier = 8; % If the 'fourier' method is chosen
% above, then this is the number of fourier series
% coefficients to fit.  Update when using fourier.
```

Output 1 - Fourier Fits



Output 2 - Regulation-detection functions



Output 3 - Regulation-detection scores

Self-Regulation scores			Regulation Scores		
-,	0.85,	-0.980	-,	-0.99,	0.21
-1,	-,	1	0.75,	-,	-1
0.89,	-1,	-	-1,	0.98,	-

Output 4 - Inferred Network

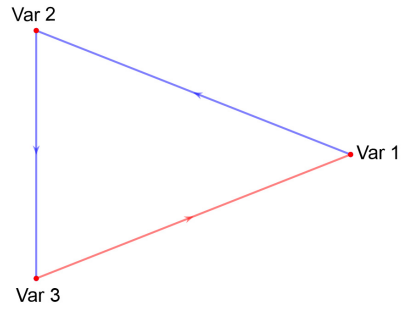


Figure S2: **Sample input and output files for the ION package if the fourier method is chosen based on the Kim-Forger model with added noise (Fig. 2B)**. The data.csv (Input 1) file contains the time points in the first column and the data for each variable in subsequent columns, separated by commas. The regulation_detection_inputs.m file (Input 2) is already in the directory and should be updated according to the preferences of the user. If the user selects the 'fourier' method, then the algorithm fits a Fourier series to each variable with the number

of coefficients specified by the variable 'num_fourier' variable (Input 2). The algorithm outputs the Fourier fits for each variable (Output 1), plots of the regulation-detection functions (Output 2), the regulation-detection scores (Output 3), and the inferred network structure (Output 4).

3.2 Frzilator

The *frzilator* model consists of a negative feedback loop that models the oscillations in a cascade of covalent modifications in *Myxococcus xanthus* (*4*) (Fig. 2C). The mathematical model uses Michaelis-Menten dynamics to model the negative feedback and is given by the equations

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}t} &= \phi\left(\frac{1-f}{0.01+(1-f)}\right) - d_f\left(\frac{f}{0.005+f}\right)e \\
\frac{\mathrm{d}c}{\mathrm{d}t} &= k_c\left(\frac{1-c}{0.005+(1-c)}\right)f - d_c\left(\frac{c}{0.005+c}\right) \\
\frac{\mathrm{d}c}{\mathrm{d}t} &= k_e\left(\frac{1-e}{0.005+1-e}\right) - d_e\left(\frac{e}{0.005+e}\right).
\end{aligned}$$

As in (*3*), we simulate the system with the nominal parameter values $\phi = 0.08$, $k_c = 4$, $k_e = 4$, $d_f = 1$, $d_c = 2$, and $d_e = 2$.

3.3 Goodwin Oscillator

The Goodwin model describes the action of a protein product repressing its *mRNA* (*5*) (Fig. 2D). The transcriptional regulation is described by a Hill function, where the Hill coefficient is an upper bound for the number of repressor proteins that bind to the promoter (*6*). A high Hill coefficient is required for the system to generate sustained oscillations. We simulated the Goodwin oscillator from the following system of ODEs.

$$\begin{aligned}
\frac{\mathrm{d}M}{\mathrm{d}t} &= \frac{1}{1+P_3^{10}} - 0.4M \\
\frac{\mathrm{d}P_1}{\mathrm{d}t} &= M - 0.4P_1 \\
\frac{\mathrm{d}P_2}{\mathrm{d}t} &= P_1 - 0.4P_2 \\
\frac{\mathrm{d}P_3}{\mathrm{d}t} &= P_2 - 0.4P_3
\end{aligned}$$

3.4 Kim-Forger Model with Outputs

We augment the original Kim-Forger model (Section 3.1) so that each component promotes the production of one output, e.g., M positively regulates X_M (Fig. 2F). The system of ODEs is given below.

$$\begin{aligned}
\frac{\mathrm{d}M}{\mathrm{d}t} &= k_1 f(P) - k_2 M \\
\frac{\mathrm{d}P_C}{\mathrm{d}t} &= k_1 M - k_3 P_C \\
\frac{\mathrm{d}P}{\mathrm{d}t} &= k_1 P_C - k_4 P \\
\frac{\mathrm{d}X_M}{\mathrm{d}t} &= k_5 \frac{M^{n_1}}{1 + M^{n_1}} - k_6 X_M \\
\frac{\mathrm{d}X_{P_C}}{\mathrm{d}t} &= k_7 \frac{P_C^{n_2}}{1 + P_C^{n_2}} - k_8 X_{P_C} \\
\frac{\mathrm{d}X_P}{\mathrm{d}t} &= k_9 \frac{P^{n_3}}{1 + P^{n_3}} - k_{10} X_P \\
f(z) &= \frac{A_t - K_d - z + \sqrt{(A_t - K_d - P)^2 - 4A_t K_d}}{2A_t}.
\end{aligned}$$

Here, the parameters k_1 through k_4 are the same as in Section 3.1. We simulate the model with the parameters $k_5 = 0.5$, $k_6 = 0.59$, $k_7 = 0.23$, $k_8 = 0.98$, $k_9 = 0.08$ and $k_{10} = 0.63$ and Hill coefficients $n_1 = 10$, $n_2 = 15$, and $n_3 = 20$.

3.5 Repressilator

The repressilator is a synthetic feedback loop that consists of three genes and three proteins where the mRNAs translate to the respective proteins, which in turn repress the transcription of the next mRNA (Fig. 2G) (*7*). The model is given by the following ODEs.

$$
\begin{aligned}
\frac{\mathrm{d}M_1}{\mathrm{d}t} &= \frac{\alpha}{1+P_3^n} - M_1 \\
\frac{\mathrm{d}P_1}{\mathrm{d}t} &= \beta(M_1 - P_1) \\
\frac{\mathrm{d}M_2}{\mathrm{d}t} &= \frac{\alpha}{1+P_1^n} - M_2 \\
\frac{\mathrm{d}P_2}{\mathrm{d}t} &= \beta(M_2 - P_2) \\
\frac{\mathrm{d}M_3}{\mathrm{d}t} &= \frac{\alpha}{1+P_2^n} - M_3 \\
\frac{\mathrm{d}P_3}{\mathrm{d}t} &= \beta(M_3 - P_3),
\end{aligned}
$$

where $\alpha = 7$, $\beta = 3$, and $n = 10$.

4 Complete results of *in silico* and experimental network inference (Figs. 2 and 4)

In Tables S1-S5, we report the complete results of our network inference procedure applied to the *in silico* examples in Fig. 2. In Tables S6-S8, we report the complete results of our network inference procedure applied to the experimental data sets (Fig. 4). The inferred interactions are the interactions that pass Rules 1-3 (Fig. 2A) with a threshold value of 0.90. Each table cell is $(\langle R_{X\to Y}\rangle, \langle R_{Y\to Y}\rangle)$ where X is the column variable and Y is the row variable.

	f	c	e
f	–	$(-1, 1)$	$(-1, -1)$
c	$(1, -1)$	–	$(-1, 0.99)$
e	$(1, -0.71)$	$(1, -1)$	–

Table S1: Calculated regulation-detection scores for the Frzilator oscillator example (Fig. 2C).

	M	P_1	P_2	P_3
M	–	$(-1, 0.98)$	$(-1, 0.61)$	$(-1, -1)$
P_1	$(1, -1)$	–	$(-1, 0.92)$	$(-1, 0.21)$
P_2	$(1, -0.30)$	$(1, -1)$	–	$(-1, 0.64)$
P_3	$(1, 0.06)$	$(1, -0.23)$	$(1, -1)$	–

Table S2: Calculated regulation-detection scores for the Goodwin oscillator example (Fig. 2D).

	M_1	P_C	P	M_2	P_1	P_2	P_3
M_1	–	$(-1, 0.68)$	$(-1, -1)$	$(-0.90, -0.56)$	$(-0.17, -0.39)$	$(0.87, -0.22)$	$(1, -0.23)$
P_C	$(1, -1)$	–	$(-1, 1)$	$(-1, -0.84)$	$(-0.90, -0.54)$	$(0.11, 0.04)$	$(0.98, 0.81)$
P	$(1, 0.76)$	$(1, -1)$	–	$(-1, -0.18)$	$(-1, -0.87)$	$(-0.97, -0.92)$	$(0.03, -0.15)$
M_2	$(0.75, -0.25)$	$(0.99, -0.17)$	$(1, -0.23)$	–	$(-1, 0.91)$	$(-1, -0.02)$	$(-1, -1)$
P_1	$(-0.28, -0.26)$	$(0.82, 0.18)$	$(1, 0.43)$	$(1, -1)$	–	$(-1, 1)$	$(-1, 0.36)$
P_2	$(-0.92, -0.85)$	$(-0.22, -0.54)$	$(0.97, 0.68)$	$(1, -0.42)$	$(1, -1)$	–	$(-1, 1)$
P_3	$(-1, -0.97)$	$(-0.97, -0.97)$	$(-0.03, -0.55)$	$(1, 0.76)$	$(1, -0.68)$	$(1, -1)$	–

Table S3: Calculated regulation-detection scores for the Kim-Forger, Goodwin independent cycle example (Fig. 2E).

	M_1	P_C	P	X_M	X_{P_C}	X_P
M_1	–	$(-1, 0.65)$	$(-1, -1)$	$(-1, 1)$	$(-1, -0.75)$	$(-0.99, -0.91)$
P_C	$(1, -1)$	–	$(-1, 1)$	$(1, -0.92)$	$(-1, 0.87)$	$(-1, 0.51)$
P	$(1, 0.87)$	$(1, -1)$	–	$(1, -0.87)$	$(1, -0.97)$	$(-1, 1)$
X_M	$(1, -1)$	$(-1, 0.45)$	$(-1, -0.40)$	–	$(-1, -0.05)$	$(-1, -0.97)$
X_{P_C}	$(1, -0.03)$	$(1, -1)$	$(-1, 0.69)$	$(1, -0.08)$	–	$(-1, 0.35)$
X_P	$(0.98, 0.31)$	$(1, 0.12)$	$(1, -1)$	$(0.99, 0.21)$	$(1, -0.51)$	–

Table S4: Calculated regulation-detection scores for the Kim-Forger with outputs example (Fig. 2F).

	M_1	P_1	M_2	P_2	M_3	P_3
M_1	–	$(-1, 1)$	$(1, 0.70)$	$(1, 0.17)$	$(-1, -0.85)$	$(-1, -1)$
P_1	$(1, -1)$	–	$(1, 0.98)$	$(1, 0.90)$	$(-1, -0.23)$	$(-1, -1)$
M_2	$(-1, -0.82)$	$(-1, -1)$	–	$(-1, 1)$	$(1, 0.71)$	(0.22)
P_2	$(-1, -0.15)$	$(-1, -1)$	$(1, -1)$	–	$(1, 0.98)$	$(1, 0.90)$
M_3	$(1, 0.79)$	$(1, 0.33)$	$(-1, -0.80)$	$(-1, -1)$	–	$(-1, 1)$
P_3	$(1, 0.99)$	$(1, 0.93)$	$(-1, -0.13)$	$(-1, -1)$	$(1, -1)$	–

Table S5: Calculated regulation-detection scores for the repressilator example (Fig. 2G).

	P_1	P_2	P_3
P_1	–	$(-0.99, -0.92)$	$(0.21, -0.006)$
P_2	$(0.75, 0.71)$	–	$(-1, -0.98)$
P_3	$(-1, -0.93)$	$(0.98, -0.04)$	–

Table S6: Calculated regulation-detection scores for the experimental repressilator example (Fig. 4B).

	P_1^1	P_2^1	P_3^1	P_1^2	P_2^2	P_3^2
P_1^1	–	$(-1, -0.96)$	$(0.56, -0.18)$	$(-0.83, -0.01)$	$(0.97, -0.57)$	$(-1, 0.37)$
P_2^1	$(0.83, 0.88)$	–	$(-1, -0.97)$	$(-0.65, 0.99)$	$(-0.19, 0.18)$	$(0.96, 0.14)$
P_3^1	$(-1, -1)$	$(0.96, 0.02)$	–	$(1, -0.03)$	$(-1, 0.16)$	$(-0.96, 0.05)$
P_1^2	$(-0.69, 0.37)$	$(0.93, -0.74)$	$(-1, -0.24)$	–	$(-0.99, -0.91)$	$(-0.04, -0.09)$
P_2^2	$(-0.97, 0.98)$	$(0.15, 0.38)$	$(1, -0.65)$	$(0.76, 0.61)$	–	$(-1, -0.96)$
P_3^2	$(1, -0.31)$	$(-1, -0.11)$	$(-0.67, 0.23)$	$(-1, -0.91)$	$(0.97, -0.04)$	–

Table S7: Calculated regulation-detection scores for the mixed repressilator example (Fig. 4C).

	hER	$POLII$	$TRIP1$	$HDAC$
hER	–	$(-0.78, 1)$	$(-0.94, -0.92)$	$(-0.41, -0.92)$
$POLII$	$(0.91, -0.99)$	–	$(-0.99, -0.49)$	$(-0.97, -0.72)$
$TRIP1$	$(0.87, 0.58)$	$(0.93, -0.10)$	–	$(-0.73, 0.66)$
$HDAC$	$(0.80, 1)$	$(0.92, 0.86)$	$(0.88, -0.23)$	–

Table S8: Calculated regulation-detection scores for the estradiol example (Fig. 4D).

References

1. J. D'Errico, Movingslope (2020).

2. J. K. Kim, D. B. Forger, *Molecular Systems Biology* **8**, 630 (2012).

3. V. Bokka, *IET Systems Biology* **12**, 190 (2018).

4. O. A. Igoshin, A. Goldbeter, D. Kaiser, G. Oster, *Proceedings of the National Academy of Sciences* **101**, 15760 (2004).

5. B. C. Goodwin, *Advanced Enzyme Regulation* **3**, 425 (1965).

6. Santillán, M., *Math. Model. Nat. Phenom.* **3**, 85 (2008).

7. M. B. Elowitz, S. Leibler, *Nature* **403**, 335 (2000).