Supporting Information

Automated refinement and inference of analytical models for metabolic networks

Michael D Schmidt[1], Ravishankar R Vallabhajosyula[2], Jerry W Jenkins[3], Jonathan E Hood[2], Abhishek S Soni[2], John P Wikswo[4], and Hod Lipson[5]

[1]Cornell Computational Systems Laboratory, Cornell University, Ithaca, NY
[2]CFD Research Corporation, Huntsville, AL
[3]HudsonAlpha Institute, Huntsville, AL
[4]Departments of Biomedical Engineering, Molecular Physiology & Biophysics, and Physics & Astronomy, and the Vanderbilt Institute for Integrative Biosystems Research and Education, Vanderbilt University, Nashville, TN
[5]School of Mechanical & Aerospace Engineering and the Department of Computing & Information Science, Cornell University, Ithaca, NY

Symbolic regression and automated experimental design to identify differential equations

Figure S1 summarizes the high-level symbolic regression of differential equations and the automated experiment control of the proposed algorithm – to use noisy metabolic time-series data to infer analytical differential equations that describe the underlying nonlinear dynamics *automatically* and without any *prior knowledge* of the metabolic system under study.

Symbolic regression algorithm settings

The core algorithm we used is described in detail in [1]. As described in the manuscript, the symbolic regression algorithm is based on genetic programming. We search for each ODE equation independently. The initial set of equations for each search is a population of randomly generated equations that are generated by filling their genetic encoding with random values, operations, and building-blocks.

The algorithm parameters are similar to a standard evolutionary algorithm, with a few new parameters needed for coevolution. The key algorithm settings are as follows: We use a population size of 512, distributed over eight CPUs/cores. We use the deterministic crowding selection method, with 5% mutation probability and 75% crossover probability. The encoding is an operation list acyclic graph with a maximum of 32 operations/nodes. Single-point crossover exchanges operations in the operation list at a random split. The operation set contains addition, subtraction, multiplication, and division algebraic operations. The fitness predictor population contains 128 predictors, distributed over eight CPUs/cores. The fitness predictor subset size is 16 indices to the full training data set. Predictors are also evolved using deterministic crowding, but with 10% mutation and 50% crossover.

These settings were chosen empirically and we have not tried to optimize them – it is likely there are more optimal settings. However, these settings appear to work well for many problems [1]. The fitness predictor population uses different settings because the genotype of the predictor is much smaller.

We calculate fitness using the correlation coefficient between the candidate solution's predicted derivative values and the numerically estimated derivatives from the training data. We

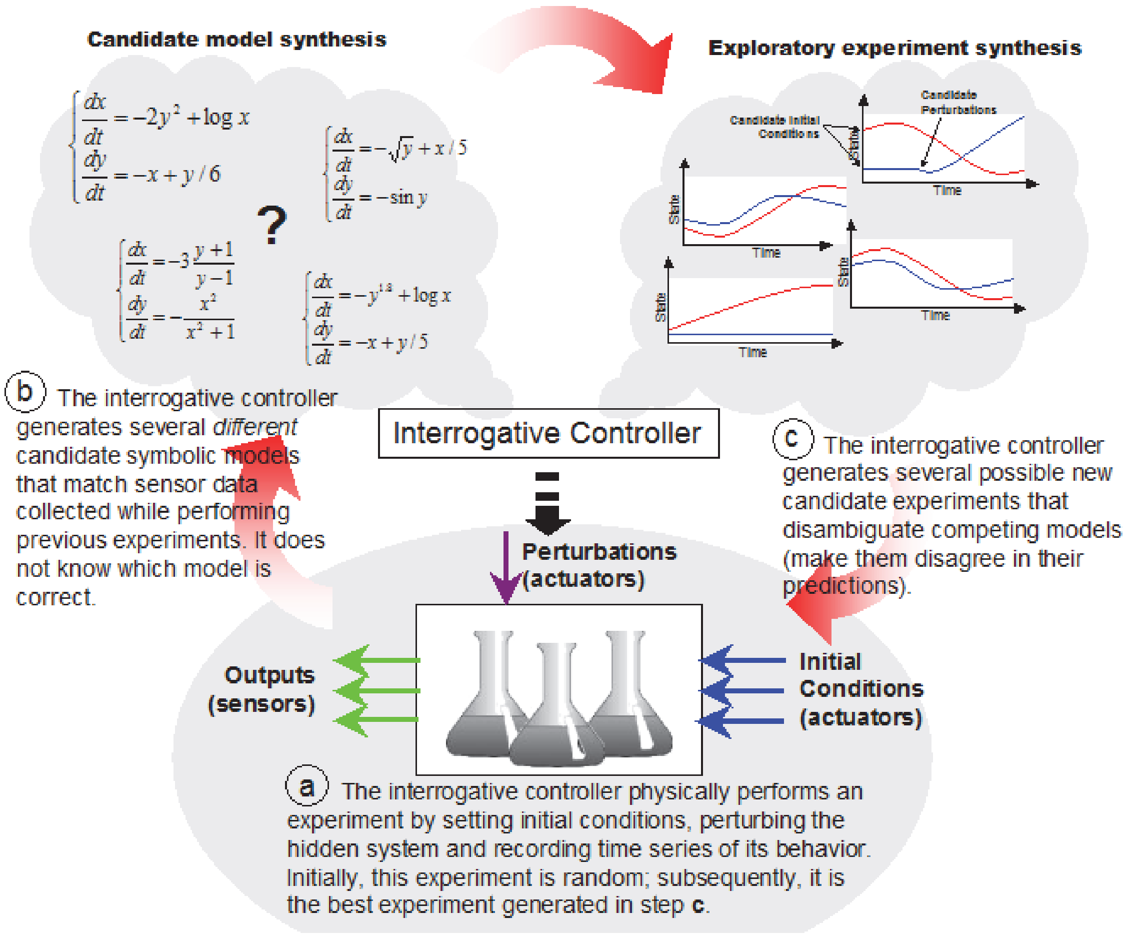

Candidate model synthesis
$\frac{dx}{dt} = -2y^2 + \log x$, $\frac{dy}{dt} = -x + y/6$
$\frac{dx}{dt} = -\sqrt{y} + x/5$, $\frac{dy}{dt} = -\sin y$
?
$\frac{dx}{dt} = -3\frac{y+1}{y-1}$, $\frac{dy}{dt} = -\frac{x^2}{x^2+1}$
$\frac{dx}{dt} = -y^{1.8} + \log x$, $\frac{dy}{dt} = -x + y/5$
Exploratory experiment synthesis
Candidate Initial Conditions
Candidate Perturbations
State
Time
b The interrogative controller generates several *different* candidate symbolic models that match sensor data collected while performing previous experiments. It does not know which model is correct.
Interrogative Controller
c The interrogative controller generates several possible new candidate experiments that disambiguate competing models (make them disagree in their predictions).
Perturbations (actuators)
Outputs (sensors)
Initial Conditions (actuators)
a The interrogative controller physically performs an experiment by setting initial conditions, perturbing the hidden system and recording time series of its behavior. Initially, this experiment is random; subsequently, it is the best experiment generated in step **c**.

Figure S1. The coevolution of models through symbolic regression and fitness prediction, and experiments by the estimation-exploration algorithm. Candidate solutions compete to explain current experimental data, and experimental initial conditions compete to maximize disagreement in the predictions of the various solutions. This process of synthesizing coherent models and controversial experiments continues until a single dominant solution.

also include a small absolute error term to provide a weak gradient to match the scale and offset of the data. The fitness function for a solution s is therefore

$$fitness(s) = \frac{\text{cov}(x,y)}{\sigma_x \sigma_y} - \varepsilon \cdot \frac{1}{n}\sum |x-y| \ ,$$

where s is a candidate differential equation, x is the model's predicted derivative values, y is the numerical derivative from the training data, σ_x and σ_y are their respective standard deviations, and cov(x,y) is the covariance of x and y. The summation is the small mean-absolute-error term, with ϵ equal to 10^{-6}. When calculating the exact fitness of a candidate solution, x and y values cover the entire training data. When predicting fitness, the x and y values cover only data samples referenced by the best current fitness predictor.

Acyclic graph encoding of the model

The acyclic graph (illustrated in Figure 3 of the paper) that represents symbolic equations was encoded internally as floating-point assembly code. The encoding consists of a list of floating-point operations and parameter values. Operations can load an input variable or a parameter value (*set* command), or perform a floating-point operation on any previous operation outputs (*add/sub/mul/div* commands). Essentially, each operation represents a leaf or parent node in the acyclic graph. The graph is rooted by the final operation in the list. Table S1 shows several raw encodings generated by the algorithm after regressing the yeast glycolysis model.

We can construct the graph by tracing backward from the last operation recursively. One notable consequence of this encoding is that some operations are unconnected in the graph – no operations branching from the output node may reference certain nodes. In effect, these vestigial sections are free to drift during regression since they have no impact on the equation (phenotype). These sections are omitted in Table S1.

We initialize the algorithm with random equations by generating a random list of floating-point operations, limited to 32 operations. We introduce variation using point mutation and crossover. A point mutation can randomly change the type of the floating-point operation (for example, flipping an *add* operation to a *multiply* or an *add* to an input variable), or randomly change the parameter constant associated with that operation (if it is used). The crossover operation recombines two existing equations to form a new equation. To perform crossover, we select a random location in the list, and copy all operation and parameter values to the left of this point from the first parent and remaining operations and parameters to the right from the second parent.

In our experiments, we are effectively searching the rational functions (seven-variable quotients of polynomials) of at most 32 operations (nodes in an acyclic graph representation). This limits the total number of parameters also to 32. The discrete search space size, neglecting real-valued parameters, is thus 6^{32} – or roughly 1025 parameterized functions.

Equation-space and accuracy/complexity tradeoff

For any given system, there is a potentially infinite set of equations that closely fit any finite set of experimentally collected data. Therefore, it is important to have some qualitative understanding of what the domain of reaction rate equations looks like. For example, a 1000th order polynomial can perfectly fit any data set of 1000 or fewer unique time samples. Therefore, it is important to understand the qualitative features of the equation-space which can also help us distinguish between true intrinsic models and coincidental fits.

Consider the relationship between equation complexity and accuracy of fitting the experimental data. Qualitatively, there exist extremely complex equations (*e.g.*, Taylor series, neural networks, and Fourier series) with near perfect accuracy as well as simple, single-parameter models with baseline accuracy (*e.g.,* the mean reaction rate). The behavior of equations in between these two extremes is more interesting.

Figure S2 shows the Pareto front of equation accuracy versus equation complexity for modeling the reaction rate of S_1 and demonstrates a clear inflection in the trade-off between model accuracy and complexity. Starting at the lower right corner of the figure and increasing the model complexity by moving to the left, there is a certain complexity where model accuracy jumps dramatically and then plateaus. In other words, there is a relatively simple equation that can model the system's behavior accurately (but perhaps not perfectly). By parsimony arguments, we can reason this equation to be the most likely model of the system. The equation at the inflection in this example is indeed the correct S_1 model, supporting this assumption.

Table S1. Raw acyclic graph encodings of differential equations found for the glycolysis yeast model. Unconnected operations are omitted.

S_1	S_2	S_3	S_4
(0) ← set <A3> (1) ← set [-7.15469] (2) ← set <S1> (3) ← mul (1) (2) (4) ← set [-10.6171] (6) ← div (4) (3) (10) ← set <S1> (12) ← set <S3> (13) ← div (3) (12) (15) ← sub (6) (10) (16) ← div (13) (15) (17) ← sub (16) (0) (18) ← sub (16) (17) (22) ← mul (18) (18) (23) ← set [0.0708605] (24) ← div (23) (0) (25) ← mul (18) (22) (26) ← add (24) (25) (27) ← div (3) (26) (28) ← set [-2.46936] (31) ← sub (27) (28) return (31)	(0) ← set [-0.234998] (1) ← set [-6.00913] (2) ← set <S2> (3) ← mul (1) (2) (4) ← add (0) (3) (5) ← set [-6.70044] (7) ← mul (5) (2) (8) ← set <N2> (9) ← mul (7) (8) (10) ← add (4) (9) (11) ← set [14.6053] (12) ← set <S1> (13) ← mul (11) (12) (14) ← set [0.071048] (15) ← set <A3> (16) ← div (14) (15) (19) ← mul (15) (15) (21) ← mul (19) (15) (22) ← add (16) (21) (23) ← div (13) (22) (24) ← add (10) (23) (25) ← set [-0.194275] (26) ← add (24) (25) (27) ← set [-0.466312] (28) ← sub (26) (27) (29) ← set [1.01609] (31) ← div (28) (29) return (31)	(0) ← set [6.01392] (1) ← set <S2> (2) ← mul (0) (1) (3) ← set [-64.187] (4) ← set <S3> (5) ← mul (3) (4) (6) ← add (2) (5) (7) ← set [16.0479] (9) ← mul (7) (4) (10) ← set <A3> (11) ← mul (9) (10) (12) ← add (6) (11) (13) ← set [-6.00042] (14) ← set <S2> (15) ← mul (13) (14) (16) ← set <N2> (17) ← mul (15) (16) (28) ← add (12) (17) (29) ← set [1] (31) ← div (28) (29) return (31)	(0) ← set [-0.0267483] (1) ← set [62.8684] (2) ← set <S3> (3) ← mul (1) (2) (4) ← add (3) (0) (5) ← set [-12.727] (6) ← set <S4> (7) ← mul (5) (6) (8) ← add (4) (7) (9) ← set [12.7542] (10) ← set <S5> (11) ← mul (9) (10) (12) ← add (8) (11) (13) ← set [-98.4028] (15) ← mul (13) (6) (16) ← set <N2> (17) ← mul (15) (16) (18) ← add (12) (17) (19) ← set [-15.7122] (20) ← set <S3> (21) ← mul (19) (20) (22) ← set <A3> (23) ← mul (21) (22) (24) ← add (18) (23) (25) ← set [1.01302] (26) ← mul (24) (25) (27) ← set [1.00701] (28) ← mul (26) (27) (29) ← set [0.021349] (31) ← add (28) (29) return (31)
N_2	A_3	S_5	
(1) ← set [5.95097] (2) ← set <S2> (3) ← mul (1) (2) (5) ← set [-17.8537] (6) ← set <S2> (7) ← mul (5) (6) (8) ← set <N2> (9) ← mul (7) (8) (10) ← add (3) (9) (11) ← set [-99.1305] (12) ← set <S4> (13) ← mul (11) (12) (15) ← mul (13) (8) (16) ← add (10) (15) (17) ← set [0.984067] (18) ← mul (16) (17) (19) ← set [0.984102] (20) ← div (18) (19) (27) ← set [-0.000345771] (28) ← add (20) (27) (29) ← set [1.01106] (31) ← mul (28) (29) return (31)	(0) ← set [0.0859636] (1) ← set [128.854] (2) ← set <S3> (3) ← mul (1) (2) (4) ← add (0) (3) (5) ← set [-1.37961] (6) ← set <A3> (7) ← mul (5) (6) (8) ← add (4) (7) (9) ← set [-32.0337] (11) ← mul (9) (2) (13) ← mul (11) (6) (14) ← add (8) (13) (15) ← set [-14.5328] (16) ← set <S1> (17) ← mul (15) (16) (18) ← set [0.0714486] (19) ← set <A3> (20) ← div (18) (6) (23) ← mul (6) (6) (25) ← mul (23) (19) (26) ← add (20) (25) (27) ← div (17) (26) (28) ← add (14) (27) (29) ← set [0.99359] (31) ← mul (28) (29) return (31)	(0) ← set [1.30265] (1) ← set <S4> (2) ← mul (1) (0) (3) ← set [-3.1032] (4) ← set <S5> (5) ← mul (3) (4) (6) ← add (2) (5) (25) ← set [-2265.46] (26) ← add (6) (25) (28) ← sub (26) (25) (29) ← set [-0.000194254] (31) ← add (28) (29) return (31)	

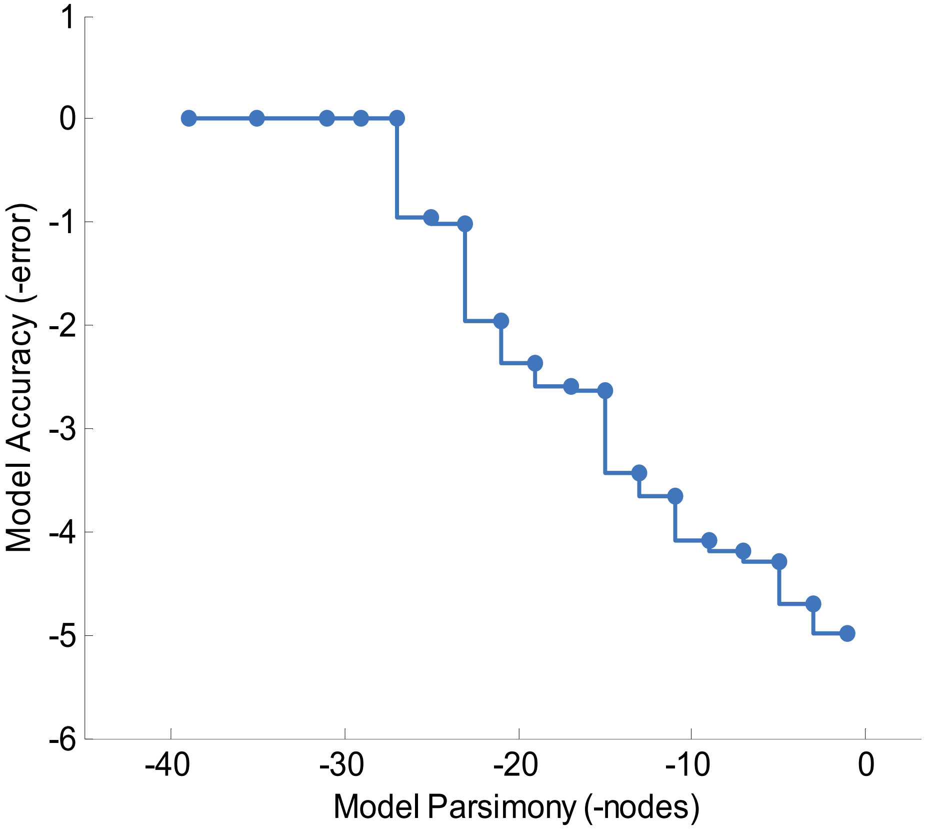


Figure S2. The Pareto front of model accuracy versus its simplicity for variable S_1. There is an inherent trade-off between complexity and accuracy to the training data. Many complex functions have very high accuracy; however, the exact solution lies at the sharp inflection near -28 nodes, balancing high accuracy and simplicity.

Distributed computation

Genetic programs are readily parallelizable to several computers and server clusters where available. We distributed the symbolic regression evolution over four computers and eight total logical processors using the island distributed computation method [2]. The island model partitions the population of solutions into separated smaller populations residing on each computer (or core). We spread a population of 512 individuals over eight CPU cores; therefore, each population has 64 individuals.

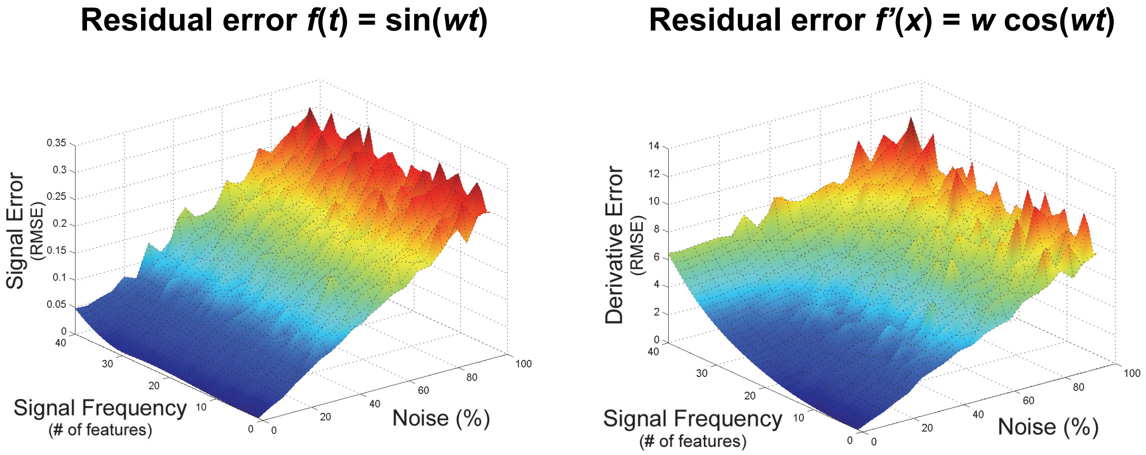
The island model populations are faster to evolve because there are fewer individuals and less work to calculate fitness values per population. We migrate solutions between populations at regular intervals. Every 10,000 iterations (averaged over all populations), we randomly shuffle all solutions among random pairs of populations.

Noise effects on estimating the gradient

Noise can make inference tasks significantly more difficult. In particular, noise makes approximating the gradient (numerical derivatives) more difficult because derivatives can be highly sensitive to noise. We used LOESS smoothing [3], a non-parametric fitting which can overcome a significant amount of noise, up to a point depending on the noise strength and frequency.

LOESS smoothing updates each sample in the data set by fitting a small order polynomial to the sample and its nearest neighbors. If the neighbor size is significantly wider than the sample rate, the polynomial will remove high-frequency noise. Other methods, such as filtering and convolution, also reduce high-frequency noise, but they do not readily produce estimates of the signal derivative. Using LOESS smoothing, we can obtain the numerical derivative directly from the smoothing procedure by evaluating the symbolic derivative of the local polynomial fit at each data sample.

In Figure S3, we can see the effect of LOESS smoothing for calculating the numerical derivative versus the amplitude of the noise and its frequency relative to the sampling rate. These graphs come from smoothing the signal $f(t)=\sin(wt)$ over $t=[0,2\pi]$. The number of features (of the data set) is defined as $2\pi w$ (the number of periods in the data set). We can see that error on the signal itself is most affected by the noise amplitude. In contrast, the error of the numerical derivative using LOESS smoothing is affected by both noise amplitude and the number of features in the data set (frequency of the signal).


Residual error *f*(*t*) = sin(*wt*)

Residual error *f'*(*x*) = *w* cos(*wt*)

Figure S3. The residual error in the signal and its derivative. The residual squared-error after LOESS smoothing versus the magnitude of the noise and the density of features relative to the noise frequency (sample rate) for (left) a sine-wave signal and (right) its numerical derivative. The signal error is most sensitive to the noise magnitude but more robust to the number of features. In contrast, the error on the numerical derivative has much higher sensitivity to the number of features. The state of the art of what the symbolic regression algorithm can handle with LOESS smoothing is roughly the medium-blue to dark-blue regions.

This result suggests that smoothing cannot remove all noise from data, even for small amounts, and that smoothing breaks down for the numerical derivative values for high-frequency features in the data. Empirically, we can also estimate the domain of system noises that the symbolic regression algorithm can handle for explicit and differential equations (roughly the medium-blue to dark-blue regions).

The glycolytic oscillation models

In Table S2 we list the chemical species and their rate/mass balance equations and initial conditions, and in Table S3, the associated reaction fluxes and kinetic coefficients.

In anaerobic metabolism in yeast, the underlying dynamics have been experimentally observed to be strongly dependent on the kinetics of the enzyme phosphofructokinase (PFK) [4]. This enzyme is activated by its substrate fructose-6-phosphate (F6P), adenosine diphosphate (ADP), and its product fructose-1,6-bisphosphate. At higher concentrations, ATP acts as an inhibitor, whereas adenosine monophosphate (AMP) activates PFK. Physical models proposed to date [5] to explain the dynamics of metabolic oscillations are based on: (a) activation of fructose-1,2-bisphosphate and ADP [6–8], or (b) inclusion of activation by AMP and inhibition by ATP [9–11]. The inclusion of adenosine nucleotides as system variables necessitates the inclusion of ATP production by the second half of glycolysis, as well as cellular ATP usage by energy-consuming processes (ATPases).

The individual reaction rates are considered to be irreversible, and are described by linear and bilinear functions of the concentrations of their substrates, except for v_1, where inhibition by ATP is taken into account. The input flux *JG* is constant and irreversible, while the output flux of *S4* is reversible and counted positive if $S_4 > S_5$. Note that the reaction v_1 effectively takes a 6 carbon compound and splits it into two 3 carbon compounds producing two S_2 species. Reaction v_1 also uses two ATPs and produces two ADPs. Reaction v_3 uses two ADPs and produces two ATPs because it is lumping the action of phosphoglycerate kinase and pyruvate kinase, both producers of ATP. Subtleties in the nonlinear terms of the model are discussed elsewhere [4–6,8–12].

Regression procedure for all methods

We also record performance on a third validation data set. The validation data set (same size and phase distribution as the training data) is used only to choose the best point during regression that maximizes generalization (a method known as “early-stopping”) for display in Figure S4.

Reverse engineering glycolytic oscillation in yeast

When looking at the experimental tests the algorithm chose during regression, it is not immediately obvious what data and initial conditions are most informative in a seven-dimensional domain. However, we can pick out some basic empirical trends. Figure S5 shows the most differentiating data points among the population of equations within a single time series. Figure S5A provides a phase-space representation of each variable and its time-derivative. The points in these trajectories are color-coded by the frequency of their use in calculating and comparing equations (via the fitness prediction). In a single time series, the importance (frequency of references by the fitness predictors) of a given point is not necessarily those system states with high derivative magnitudes. Instead, heavy importance tends to lie near inflections around the limit cycle for most variables.

Table S2. The chemical species in the model (NM, IM, and OS are the normal, impaired, and overspecified models, respectively). Initial conditions for the limit cycle are provided. In the algorithm, however, initial conditions are chosen randomly at first, and then based on the current models found in the search, subject to constraints listed in Table 1.

Variable	Description	Model	Species rate or mass balance	Initial conditions
A_2	ADP	All	$A_2 + A_3 = A$	1.525 mM
A_3	ATP	All	$\dot{A}_3 = -2v_1 + 2v_3 - v_5$	2.475 mM
N_1	NAD$^+$	All	$N_1 + N_2 = N$	0.923 mM
N_2	NADH	NM OS	$\dot{N}_2 = v_2 - v_4 - v_6$	0.077 mM
		IM	$\dot{N}_2 = v_2 - J_P$	
S_1	Glucose	All	$\dot{S}_1 = J_G - v_1$	1.187 mM
S_2	Glyceraldehydes-3-phosphate and dihydroxyacetone phosphate pool	NM OS	$\dot{S}_2 = 2v_1 - v_2 - v_6$	0.193 mM
		IM	$\dot{S}_2 = 2v_1 - v_2$	
S_3	1,3-bisphosphoglycerate	All	$\dot{S}_3 = v_2 - v_3$	0.050 mM
S_4	Cytosolic pyruvate and acetaldehyde pool	NM	$\dot{S}_4 = v_3 - v_4 - J_P$	0.115 mM
		IM	$\dot{S}_4 = v_3 - J_P$	
		OS	$\dot{S}_4 = v_3 - v_4 - J_P - v_{sink}$	
S_5	Extracellular concentration of S_4	All	$\dot{S}_5 = \varphi(J_P - v_7)$	0.077 mM $\varphi = 0.10$

Table S3. Description of the reaction fluxes shown in Figure 2 in the main text and the values of their kinetic coefficients (NM, IM, and OS are the normal, impaired, and overspecified models, respectively).

Reaction enzymes or processes represented	Model	Reaction	Coefficient value
Incoming flux of glucose across cell membrane	All	$J_G = \text{constant}$	$J_G = 2.5\,\text{mM/min}$
Hexokinase, phosphoglucoisomerase, and phosphofructokinase, where K_I is the inhibition constant and the exponent 'q' is the cooperativity coefficient of ATP inhibition	All	$v_1 = \dfrac{k_1 S_1 A_3}{1 + \left(\dfrac{A_3}{K_I}\right)^q}$	$k_1 = 100\ \text{mM/min}$ $K_I = 0.52\ \text{mM}$ $q = 4.0$
Glyceraldehydes-3-phosphate dehydrogenase	All	$v_2 = k_2 S_2 N_1$	$k_2 = 6.0\ \text{mM/min}$
Phosphoglycerate kinase, phosphoglycerate mutase, enolase, and pyruvate kinase	All	$v_3 = k_3 S_3 A_2$	$k_3 = 16.0\,\text{mM/min}$
Alcohol dehydrogenase	NM OS	$v_4 = k_4 S_4 N_2$	$k_4 = 100\ \text{mM/min}$
	IM	**Absent**	
Nonglycolytic ATP consumption	All	$v_5 = k_5 A_3$	$k_5 = 1.28\ \text{min}^{-1}$
Formation of glycerol from triose phosphates	NM OS	$v_6 = k_6 S_2 N_2$	$k_6 = 12.0\,mM/\text{min}$
	IM	**Absent**	
Degradation of pyruvate and acetaldehyde in the extracellular space	All	$v_7 = kS_5$	$k = 1.8\ \text{min}^{-1}$
Carbon sink term to the pyruvate pool accounting for the carbon loss to cellular synthetic processes (fatty acid biosynthesis, amino acid production)	OS	$v_{sink} = \dfrac{k_{sink} S_4}{1 + \left(\dfrac{A_3}{K_{IATP}}\right)^3}$	$k_{sink} = 20\ \text{mM/min}$ $K_{IATP} = 0.52\ \text{mM}$
Membrane transport of pyruvate and acetaldehyde into extracellular space (A_s = membrane surface, P = membrane permeability, and V = cellular volume)	NM OS	$J_P = \left(\dfrac{A_S P}{V}\right)(S_4 - S_5)$	$\left(\dfrac{A_S P}{V}\right) = 13.0\ \text{min}^{-1}$
	IM	$J_P = \left(\dfrac{A_S P}{V}\right)(S_4)$ (Note A)	

Note: In the case of the impaired model, mammalian cells do not typically take in lactate from the extracellular space, so the dependence on S_5 was eliminated to ensure that the model would act like a mammalian cell.

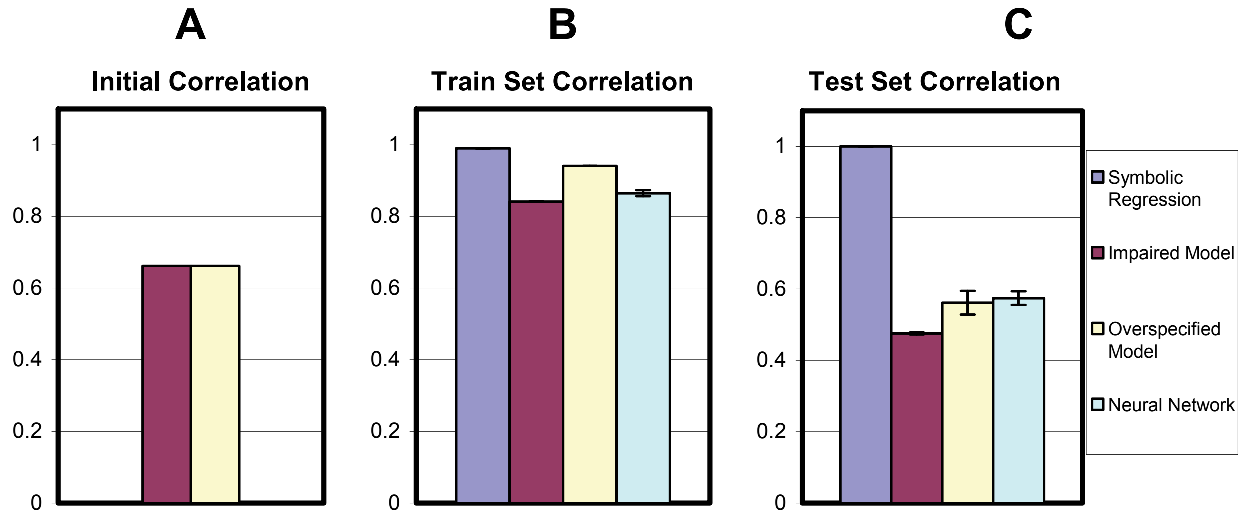


Figure S4. Correlations of the various regressions averaged over 100 trials on equation *S₄*. Error bars represent the standard error. (A) The correlations between the training data and each initial model before the model is regressed to the training data by the corresponding algorithm. Symbolic regression and neural network regression must model the system from scratch and initially have zero correlation. The impaired and overspecified models are close approximations to the exact model and therefore have positive correlations. (B) The mean correlation of the best solution from ten runs of each algorithm to the training data. The training data contain 10% random noise, which results in slight variances – most notably in the neural networks. The best solution from each algorithm correlates well to the training data with low standard error. (C) The mean correlation of each method to the test data. The assumed structures of the impaired and overspecified models appear to limit their ability to model a wider phase domain. The neural network appears limited by noise in the system, but does achieve a slightly higher correlation on average with the test set than do the impaired and overspecified models.

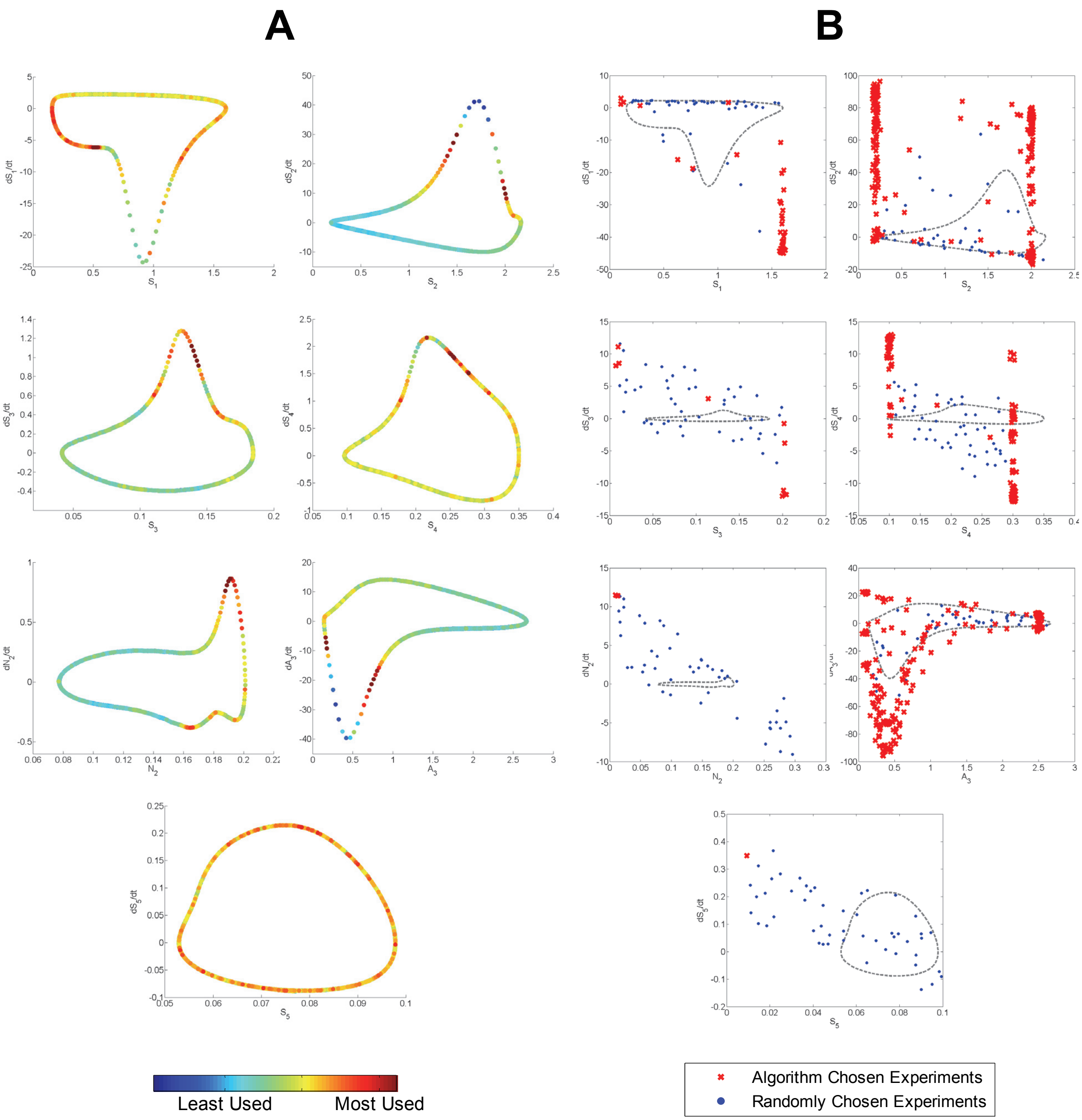


Figure S5. Performance of the calculation in the phase plane for each variable. The left set of panes (A) shows the glycolysis system near the stable limit cycle in the course of a single experiment, with colors representing frequency with which the fitness predictor examines each point within a single time series. The right set (B) shows the initial condition experiments (red) chosen by the algorithm to differentiate solutions in comparison to a random distribution of initial conditions (blue). The algorithm tends to focus on nonlinear states away from the limit cycle (dashed black line) within the experimental constraints imposed upon the estimation-exploration algorithm.

Figure S5B shows the range of initial conditions (red) chosen by the estimation-exploration algorithm (EEA) as it suggests new experiments for each iteration in the series. The blue points show the range of derivative values for randomly chosen states. The dashed line shows the limit cycle for each variable. With the exception of A_3, the EEA preferentially chooses new experiment initial conditions near extremities of the allowed range of each variable (see “Methods”), away from the limit cycle. These initial conditions are more likely to amplify nonlinear features of the system, which is also consistent with the observed behavior of the fitness predictor. Therefore, the maximum disagreement criterion for new experiments may in effect reveal information about the nonlinear terms, which appear to be the most used data points for synthesizing models within single trajectories. Initial conditions and measurements on the limit cycle provide much less information than ones that lie outside the limit cycle for which the system must descend into the limit.

The amount of noise in the system affects the frequency of finding the exact differential equation for each state-variable differently.

Figure S6 shows the rate of convergence (success rate) for each equation within one hour of regression. The most complicated differential equations (S_1, S_2, and A_3) are also the most sensitive to noise. We have found that noise obscures subtle features in these equations, resulting in partial regression of the exact equations. For example, in the solution for S_2 in Table 2, the v_1 reaction term is found exactly, but the v_4+v_6 reactions are approximated.

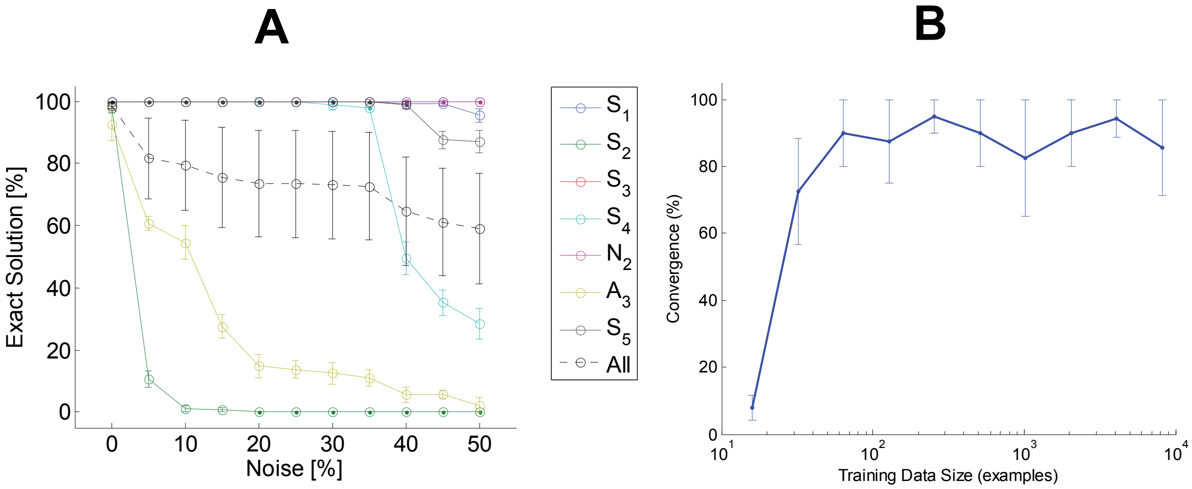


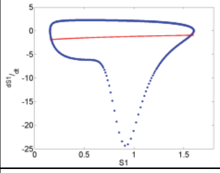
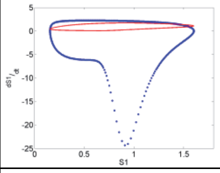
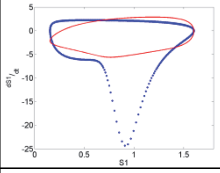
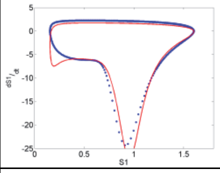
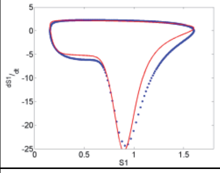
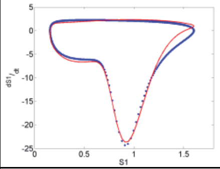
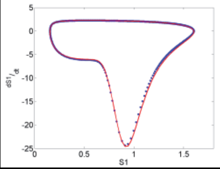
Figure S6. The effects of noise and training data set size on convergence. A: The rate of successful inference of the exact differential equation for each state-variable versus the observation noise in the system after one hour of regression. The convergence rate is calculated from ten independent trials on each equation at each noise level. B: The rate of successful inference of the exact differential equation for all variables versus the total amount of data given to the system after one hour. The error bars indicate the standard deviation in convergence among the seven variables.

Sequence of solutions

Since symbolic regression begins with randomly generated solutions (differential equations), it is interesting to observe the evolutionary path these solutions take toward the final model. Table S4 shows one evolutionary sequence for the variable S1. The solutions tend to grow gradually in complexity from the initially random solutions. The fit to the data improves incrementally. Finally, a solution that contains most of the exact model emerges, and the solution prunes down as it fits the last remaining features.

Table S4. Seven snapshots of the best solution during regression of S_1. The solution is plotted in red and the systems limit cycle is shown in blue.

O = number of operators in the equations; P = number of coefficients/parameters.

Generation	Fit to Limit Cycle	Current Best Model	O	P
2		$\frac{dS_2}{dt}=\frac{-2.5028}{S_3+S_2+1}$	3	2
190		$\frac{dS_2}{dt}=\frac{A_3}{S_3+1.4659}$	2	1
2,605		$\frac{dS_2}{dt}=\frac{5.9310(A_3-1.6763)}{S_2-S_1+A_3+0.1587}$	6	3
316,029		$\frac{dS_2}{dt}=\frac{S_4+2S_1+2N_2-2A_3(A_3-0.9450)-0.9950}{A_3(0.9450-A_3)-0.2948}$	13	7
407,083		$\frac{dS_2}{dt}=2S_5+\frac{2S_1}{(0.7557-A_3)\cdot A_3-0.2046}+\frac{2N_2}{A_3}+2.1192$	11	6
2,835,858		$\frac{dS_2}{dt}=\Big((2.1623-A_3)\cdot\big((-(2.1623-A_3)^3)-1.079\big)\Big)\cdot S_1+1.8$	8	4
4,444,185		$\frac{dS_2}{dt}=2.5308-\frac{42.3825S_1}{5.4326A_3^3+\frac{0.4290}{A_3}}$	5	4

Comparison to other methods

As shown in Table S5, we tested the performance of each method on modeling the time-derivative of S_4, the equation that differs the most between the impaired and overspecified nonlinear regression models. The symbolic regression algorithm must search for and fit the equation from scratch, whereas the nonlinear regression and neural network modeling must tune parameters. The training data are static and were generated using the exact model – there were no algorithmically chosen experiments. As before, the test dataset has an upper-bound constraint that is twice that used for the training data set. Additionally, the training data set again contains 10% random noise on every measurement. We stop regression after the solutions stop improving when evaluated on the test data set.

Table S5. The equations for S$_4$ (pyruvate and acetaldehyde pool) for the exact, impaired, and overspecified models shown in Figure 3 in the main text. The exact values for the parameters are k_3 = 16, k_4 = 100, and A_sP/V = 13.

Model name	Differential equation	Regressed parameters
Exact model	$\frac{dS_4}{dt} = k_3S_3A_2 - k_4S_4N_2 - \left(\frac{A_SP}{V}\right)\left(S_4 - S_5\right)$	Symbolic Regression of Exact Test Set: $k_3 = 16.03,\ 16.01$ $k_4 = 100.11$ $\frac{A_SP}{V} = 13.21,\ 13.03$
Impaired model	$\frac{dS_4}{dt} = k_3S_3A_2 - \left(\frac{A_SP}{V}\right)S_4$	Nonlinear Regression of Exact Test Set to Impaired Model: $k_3 = 13.76635$ $\frac{A_SP}{V} = 21.2331,$
Over-specified model	$\frac{dS_4}{dt} = k_3S_3A_2 - k_4S_4N_2 - \left(\frac{A_SP}{V}\right)\left(S_4 - S_5\right) - \frac{k_{sink}S_4}{1+\left(\frac{A_3}{K_{IATP}}\right)^3}$	Nonlinear Regression of Exact Test Set to Overspecified Model: $k_3 = 13.76635$ $k_3 = 15.8508$ $k_4 = 94.812$ $\frac{A_SP}{V} = 12.0785$ $k_{\text{sink}} = 0.411579$ $k_{ATP} = 0.5264$

Computing quantities of interest from the invariant expression

The invariant constructed in this work using symbolic regression enables the computation of quantities of interest in the biological system such as rate expression and numerical sensitivities. This approach utilizes the algebraic structure of the invariant expression to approximate these quantities. Rearranging the terms of the invariant and solving for the reaction rate v_1 yields an expression which closely matches the analytical rate expression from the model to a very high degree of accuracy. This was carried out as follows, starting with the expression for S_1, which was obtained by symbolic regression:

$$S_1 = (v_1\ (A_3^4\,(k_1 - k_2 N_2) + k_3))/A_3\,.$$

The corresponding invariant expression can then be written as

$$Invariant = S_1 - (v_1\ (A_3^4\,(k_1 - k_2 N_2) + k_3))/A_3\,.$$

The magnitude of the invariant can be made sufficiently small by requiring the symbolic regression algorithm to find suitable coefficients k_1, k_2, and k_3 to the desired accuracy. We then approximated the reaction rate by setting the value of the invariant to zero and solving for v_1. The resultant expression obtained in this manner is given as

$$v_1 \approx (A_3 S_1)/[\,A_3^4\,(k_1 - k_2 N_2) + k_3]\ .$$

A plot of v_1 obtained using this expression for three temperatures is shown in Figure S7 a, c, and d, along with the error in approximation arising from the accuracy of the invariant coefficients and the corresponding magnitude of the invariant in Figure S7 b, d, and f. This is indicative of the temperature dependency of the coefficients of the invariant in a manner equivalent to the Arrhenius form dependency of the rate constants.

The algebraic form of the invariant also simplifies the task of computing the numerical sensitivities that provide insight into the dynamics of the system and its control. In this case, we have computed the scaled numerical sensitivity of v_1 with respect to both A_3 and N_2. These numerical sensitivities are given as

$$\varepsilon_{A_3}^{v_1} = \frac{A_3}{v_1}\frac{\partial v_1}{\partial A_3} \text{ and } \varepsilon_{N_2}^{v_1} = \frac{N_2}{v_1}\frac{\partial v_1}{\partial N_2} \text{ , respectively.}$$

These sensitivities have been abbreviated as eAv and eNv in Figure S8 and Figure S9. They were computed by differentiating the algebraic form of v_1 and substituting the implicit derivatives such as $\partial S_1/\partial A_3$ and $\partial S_1/\partial N_2$ by their numerical approximations, which were evaluated by means of central differences. This approach yielded a simple means of computing the numerical sensitivities. The numerical sensitivities play an important role in variables in the systems that can be targeted by appropriate means.

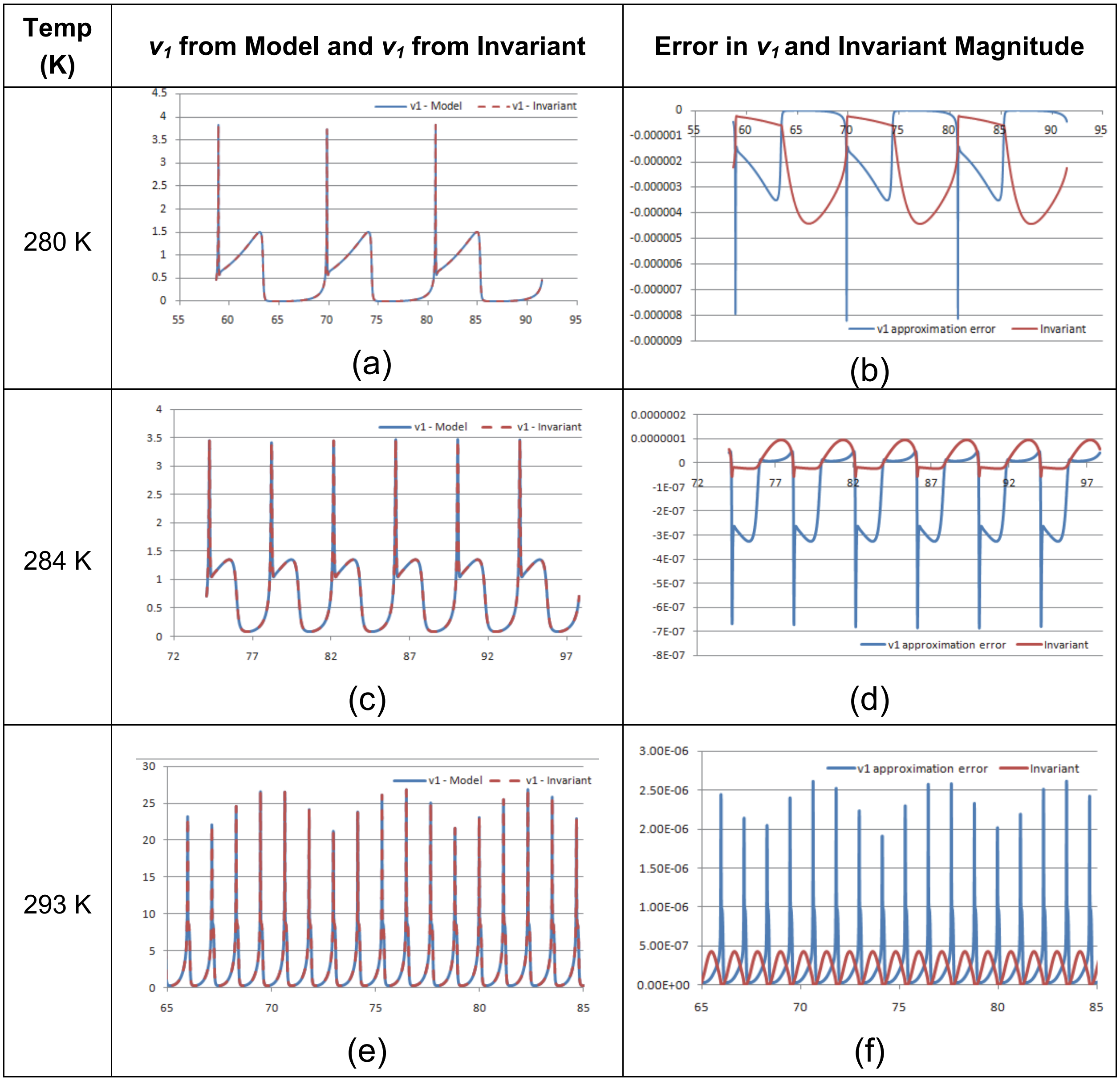
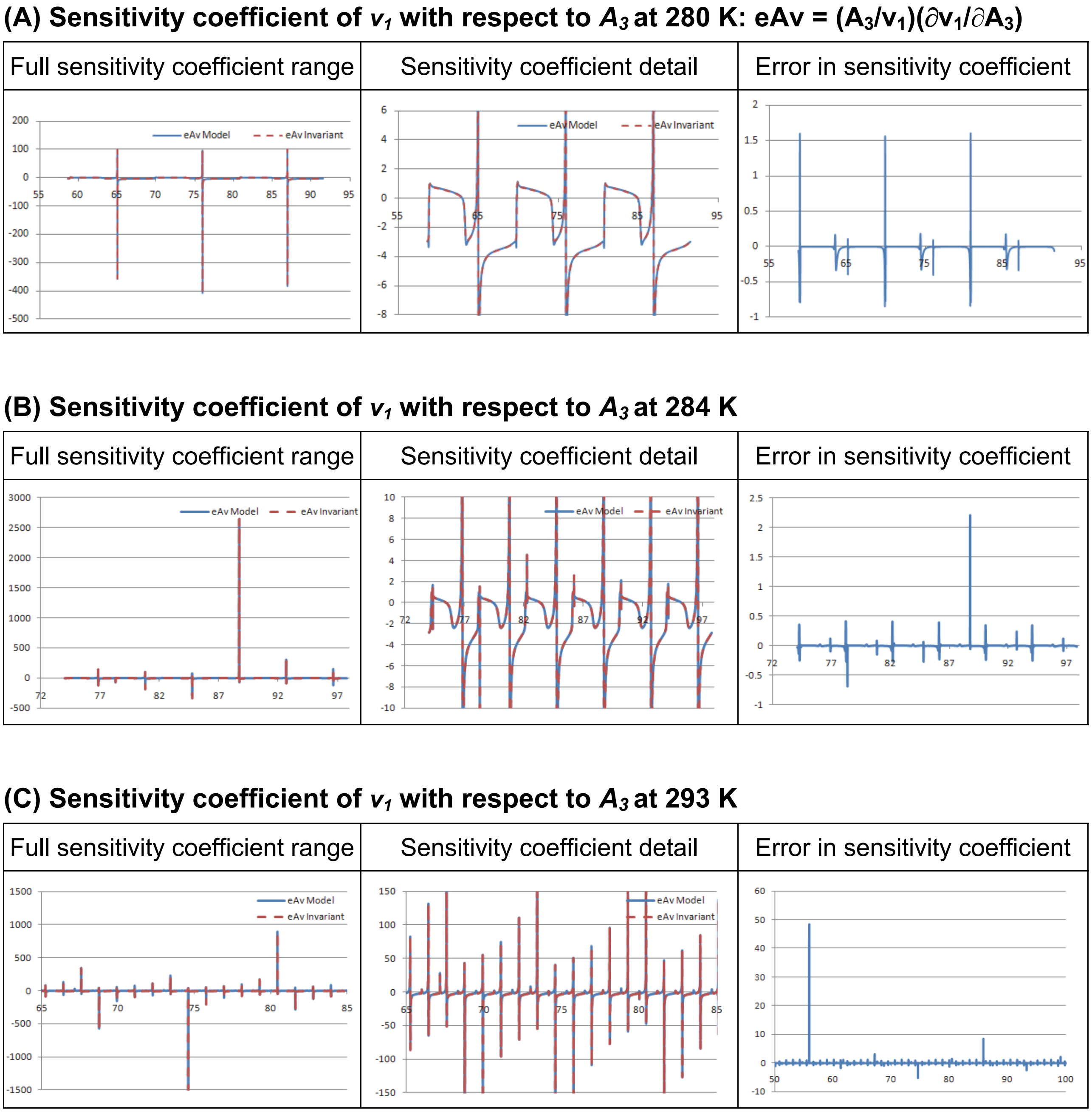


Figure S7. Comparison of the analytic form of the rate expression v_1 from the model with that obtained from the invariant. (A) The comparison at 280 K. (C) The comparison at 284 K. (E) The comparison at 293 K. The corresponding approximation error between the model and invariant v_1 is compared with the magnitude of the invariant at each temperature in (B) for 280 K, in (D) for 284 K, and in (F) for 293 K.



(A) Sensitivity coefficient of v_1 with respect to A_3 at 280 K: eAv = $(A_3/v_1)(\partial v_1/\partial A_3)$

Full sensitivity coefficient range | Sensitivity coefficient detail | Error in sensitivity coefficient

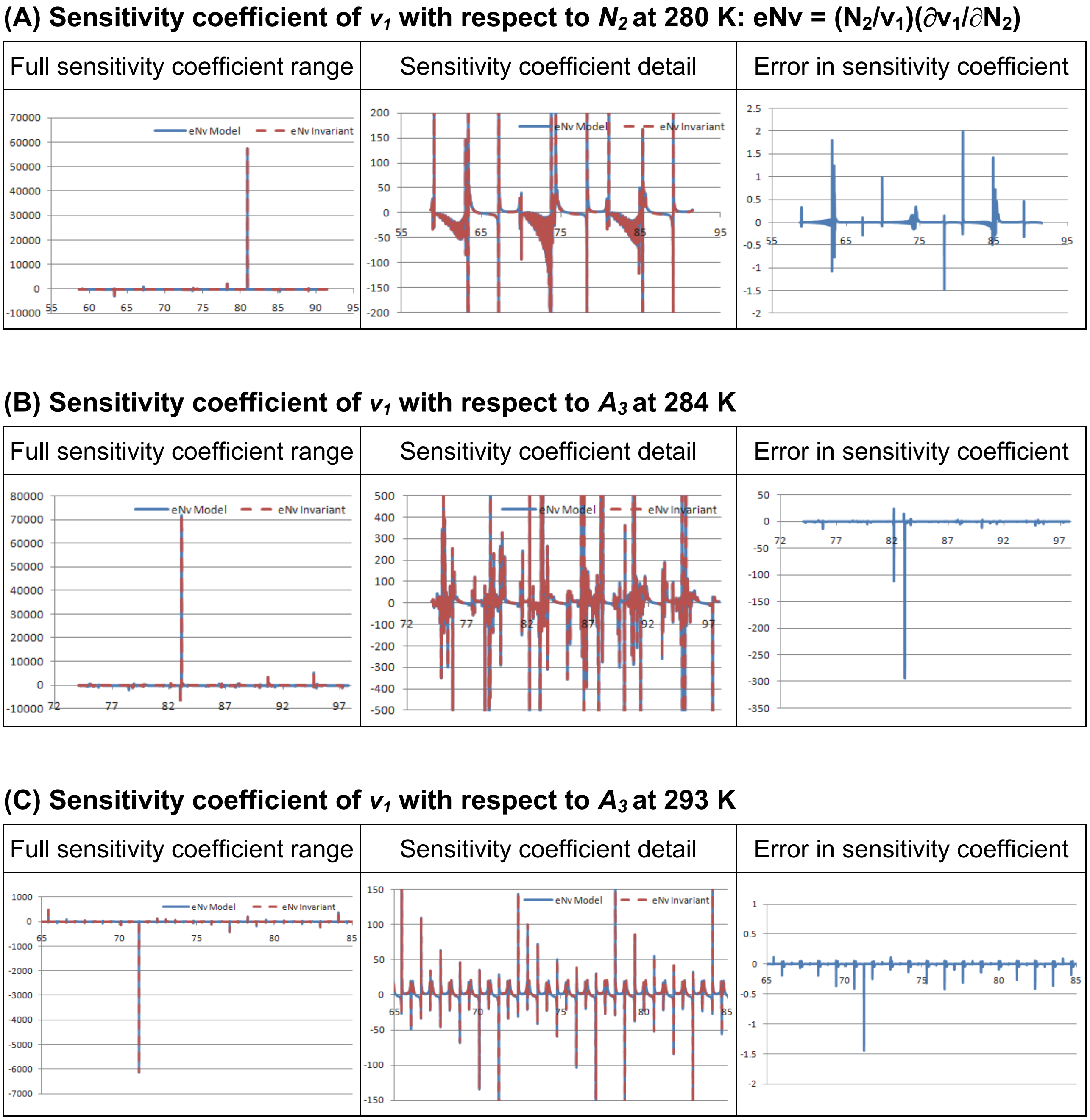
(B) Sensitivity coefficient of v_1 with respect to A_3 at 284 K

Full sensitivity coefficient range | Sensitivity coefficient detail | Error in sensitivity coefficient

(C) Sensitivity coefficient of v_1 with respect to A_3 at 293 K

Full sensitivity coefficient range | Sensitivity coefficient detail | Error in sensitivity coefficient

Figure S8. Sensitivity coefficient of v_1 with respect to A3 shown for three temperatures: (A) for 280 K, (B) for 284 K, and (C) for 293 K. For each temperature, the left panel shows the full range of both the model-derived as well as the invariant-derived sensitivity coefficients. The middle panel shows the area closer to the origin, where it can be seen that the invariant-derived sensitivity coefficient tracks the actual sensitivity coefficient very well. The right panel for each temperature shows the error between the two sensitivity coefficients.



(A) Sensitivity coefficient of v_1 with respect to N_2 at 280 K: eNv = $(N_2/v_1)(\partial v_1/\partial N_2)$

Full sensitivity coefficient range | Sensitivity coefficient detail | Error in sensitivity coefficient

(B) Sensitivity coefficient of v_1 with respect to A_3 at 284 K

Full sensitivity coefficient range | Sensitivity coefficient detail | Error in sensitivity coefficient

(C) Sensitivity coefficient of v_1 with respect to A_3 at 293 K

Full sensitivity coefficient range | Sensitivity coefficient detail | Error in sensitivity coefficient

Figure S9. Sensitivity coefficient of *v_1* with respect to N2 shown for three temperatures: (A) for 280 K, (B) for 284 K, and (C) for 293 K. For each temperature, the left panel shows the full range of both the model-derived as well as the invariant-derived sensitivity coefficients. The middle panel shows the area closer to the origin where it can be seen that the invariant-derived sensitivity coefficient tracks the actual sensitivity coefficient very well. The right panel for each temperature shows the error between the two sensitivity coefficients.

References

1. Schmidt MD, Lipson H (2008) Coevolution of fitness predictors. IEEE Trans Evol Comput 12: 736-749.

2. Fernandez F, Tomassini M, Vanneschi L (2003) An empirical study of multipopulation genetic programming. Genetic Programming and Evolvable Machines 4: 21-51.

3. Cleveland WS, Devlin SJ (1988) Locally weighted regression - an approach to regression-analysis by local fitting. J Am Stat Assoc 83: 596-610.

4. Goldbeter A (1996) Biochemical oscillations and cellular rhythms: the molecular bases of periodic and chaotic behaviour. Cambridge and New York: Cambridge University Press.

5. Wolf J, Heinrich R (2000) Effect of cellular interaction on glycolytic oscillations in yeast: a theoretical investigation. Biochem J 345: 321-334.

6. Goldbeter A, Lefever R (1972) Dissipative structures for an allosteric model - application to glycolytic oscillations. Biophys J 12: 1302-1315.

7. Selkov EE (1968) Self-oscillations in glycolysis 1. A simple kinetic model. Eur J Biochem 4: 79-86.

8. Higgins J (1964) A chemical mechanism for oscillation of glycolytic intermediates in yeast cells. PNAS 51: 989-994.

9. Richter O, Betz A, Giersch C (1975) Response of oscillating glycolysis to perturbations in Nadh-Nad system - comparison between experiments and a computer model. BioSystems 7: 137-146.

10. Termonia Y, Ross J (1981) Oscillations and control features in glycolysis - numerical-analysis of a comprehensive model. PNAS (US) 78: 2952-2956.

11. Smolen P (1995) A model for glycolytic oscillations based on skeletal-muscle phosphofructokinase kinetics. J Theor Biol 174: 137-148.

12. Ruoff P, Christensen MK, Wolf J, Heinrich R (2003) Temperature dependency and temperature compensation in a model of yeast glycolytic oscillations. Biophys Chem 106: 179-192.