Koopman Operator Methods for Analysis & Design of Synthetic and Natural Gene Networks

SIAM Conference on the Life Sciences (LS22) Tutorial Session MT1, Room 328, 4-6 pm EST

Professor Enoch Yeung

Center for Control, Dynamical Systems, and Computation Biomolecular Science and Engineering Program Department of Mechanical Engineering University of California Santa Barbara
Outline of Tutorial:

- Introduction to Koopman Operators
- Introduction to deep dynamic mode decomposition
- Examples of deepDMD applied to biological systems:
  - Glycolysis model/network
  - Genetic toggle switch
- Code usage
- Research Application 1: Engineering organism persistence
- Research Application 2: Extracting novel genetic sensors
Koopman Operators Enable Discovery of Predictive Models Directly From Data

Koopman Operators Enable Discovery of Predictive Models Directly From Data


Outline of Tutorial:

• Introduction to Koopman Operators

• **Introduction to deep dynamic mode decomposition**

• Examples of deepDMD applied to biological systems:
  • Glycolysis model/network
  • Genetic toggle switch

• Code usage

• Research Application 1: Engineering organism persistence

• Research Application 2: Extracting novel genetic sensors
Traditional Methods for Learning Koopman Operators Rely on Manual Dictionary Curation
Traditional Methods for Learning Koopman Operators Rely on Manual Dictionary Curation

Extended Dynamic Mode Decomposition

\[
D = \begin{bmatrix}
\psi_1(y_n) & \psi_2(y_n) & \cdots & \psi_m(y_n)
\end{bmatrix}
\]

\[
\psi(y_n) \rightarrow K \rightarrow \psi(y_{n+1})
\]
Traditional Methods for Learning Koopman Operators
Rely on Manual Dictionary Curation
Our Proposed Approach: Deep Dynamic Mode Decomposition

Deep Dynamic Mode Decomposition

\[
D_{NN} = \begin{bmatrix}
\psi_1(y_n) & \psi_2(y_n) & \cdots & \psi_m(y_n)
\end{bmatrix}
\]

Time-series data

Automated dictionary learning

Loss Function
Deep Dynamic Mode Decomposition

\[ D_{\text{NNN}}^{\text{NNN}} = \begin{bmatrix} \psi_1(y_n) & \psi_2(y_n) & \cdots & \psi_m(y_n) \end{bmatrix} \]

**Time-series data**

\[ \psi(y_n) \rightarrow y_{n+1} \]

\[ \psi_{\text{est}}(y_{n+1}) \]

**Automated dictionary learning**

\[ \sum \]

\[ \varepsilon(\psi_{\text{act}}, \psi_{\text{est}}) \]

\[ \text{Loss Function} \]
Deep Dynamic Mode Decomposition

\[ D_{\mathcal{NN}} = \begin{bmatrix} \psi_1(y_n) & \psi_2(y_n) & \cdots & \psi_m(y_n) \end{bmatrix} \]

See also:
Li et al. 2017
Yeung, Kundu, and Hodas (arXiv), 2017
N. Takeishi, Y. Kawahara, T. Yairi, 2017
Lusch, Kutz, and Brunton 2018
A summary of deep dynamic mode decomposition:

\[
\min_{K, \theta} \left\| N_{\Psi}(x_{n+1}, \theta) - KN_{\Psi}(x_n, \theta) \right\|_2 + \lambda_1 \left\| K \right\|_2 + \lambda_2 \left\| (\theta) \right\|_1
\]

where the weights of the neural network are expressed as:

\[
\theta = (W_1, \ldots, W_d, b_1, \ldots, b_d)
\]

and the neural network of depth \(d\) can be written as

\[
N_{\Psi}(x_n) = h_d \circ h_{d-1} \circ \ldots \circ h_1(x_n)
\]

with

\[
h_j(h_{j-1}) = \sigma_j(W_{j-1}h_{j-1} + b_{j-1}).
\]
A summary of deep dynamic mode decomposition:

The deep dynamic mode decomposition can be written as the non-convex optimization problem:

\[
\min_{K, \theta} \| N_\Psi(x_{n+1}, \theta) - KN_\Psi(x_n, \theta) \|_2 + \lambda_1 \| K \|_2 + \lambda_2 \| (\theta) \|_1
\]

Notationally, the neural network data matrices mirror the standard structure of regularized E-DMD

\[
\min_{K} \| Y_f - KY_p \|_2 + \lambda \| K \|_{2,1}
\]

with

\[
Y_f = \begin{bmatrix}
N_\Psi(x_{n+1}^{(0)}) & \cdots & N_\Psi(x_1^{(0)}) \\
\vdots & \ddots & \vdots \\
N_\Psi(x_{n+1}^{(p)}) & \cdots & N_\Psi(x_1^{(p)})
\end{bmatrix} \quad Y_p = \begin{bmatrix}
N_\Psi(x_n^{(0)}) & \cdots & N_\Psi(x_0^{(0)}) \\
\vdots & \ddots & \vdots \\
N_\Psi(x_n^{(p)}) & \cdots & N_\Psi(x_0^{(p)})
\end{bmatrix}
\]
Outline of Tutorial:

- Introduction to Koopman Operators
- Introduction to deep dynamic mode decomposition
- **Examples of deepDMD applied to biological systems:**
  - Glycolysis model/network
  - Genetic toggle switch
- Code usage
- Research Application 1: Engineering organism persistence
- Research Application 2: Extracting novel genetic sensors
Deep Dynamic Mode Decomposition on the Glycolytic Oscillator

Glycolysis example:

\[
\begin{align*}
\frac{dS_1}{dt} &= J_0 - \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q}, \\
\frac{dS_2}{dt} &= 2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5, \\
\frac{dS_3}{dt} &= k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6), \\
\frac{dS_4}{dt} &= k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7), \\
\frac{dS_5}{dt} &= k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5, \\
\frac{dS_6}{dt} &= -2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} + 2k_3 S_3 (A - S_6) - k_5 S_6, \\
\frac{dS_7}{dt} &= \mu \kappa (S_4 - S_7) - k_7,
\end{align*}
\]
Deep Dynamic Mode Decomposition on the Glycolytic Oscillator

Glycolysis example:

\[ \frac{dS_1}{dt} = J_0 - \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q}, \]
\[ \frac{dS_2}{dt} = 2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5, \]
\[ \frac{dS_3}{dt} = k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6), \]
\[ \frac{dS_4}{dt} = k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7), \]
\[ \frac{dS_5}{dt} = k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5, \]
\[ \frac{dS_6}{dt} = -2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} + 2k_3 S_3 (A - S_6) - k_5 S_6, \]
\[ \frac{dS_7}{dt} = \mu \kappa (S_4 - S_7) - k S_7, \]
Deep Dynamic Mode Decomposition on the Glycolytic Oscillator

Glycolysis example:

\[
\begin{align*}
\frac{dS_1}{dt} &= J_0 - \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q}, \\
\frac{dS_2}{dt} &= 2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5, \\
\frac{dS_3}{dt} &= k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6), \\
\frac{dS_4}{dt} &= k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7), \\
\frac{dS_5}{dt} &= k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5, \\
\frac{dS_6}{dt} &= -2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} + 2 k_3 S_3 (A - S_6) - k_5 S_6, \\
\frac{dS_7}{dt} &= \mu \kappa (S_4 - S_7) - k S_7,
\end{align*}
\]

Deep DMD

- actual
- predicted
Deep Dynamic Mode Decomposition on the Glycolytic Oscillator

Glycolysis example:

\[
\begin{align*}
\frac{dS_1}{dt} &= J_0 - \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q}, \\
\frac{dS_2}{dt} &= 2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5, \\
\frac{dS_3}{dt} &= k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6), \\
\frac{dS_4}{dt} &= k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7), \\
\frac{dS_5}{dt} &= k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5, \\
\frac{dS_6}{dt} &= -2 \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} + 2k_3 S_3 (A - S_6) - k_5 S_6, \\
\frac{dS_7}{dt} &= \mu \kappa (S_4 - S_7) - k S_7,
\end{align*}
\]

Extended DMD

Deep DMD

eeyeung@ucsb.edu
DeepDMD Recovers Predictive Models of the Glycolytic Oscillator with Unmeasured States

Extended DMD
- Actual
- Predicted

Deep DMD
- Actual
- Predicted

Specifically, we are interested in power systems transient dynamics, which models the evolution of rotor angles of the synchronous machines at sub-second to a couple of seconds timescales. At this timescale, the dynamics of interest can be modeled as the classical swing dynamics:

\[
\dot{\phi}_i = \omega_i, \\
\dot{\omega}_i = \frac{1}{M_i} (D_i \omega_i + P_{m,i}) - \frac{1}{2} \sum_{j=1}^{n} V_j (G_{ij} \cos(\phi_i - \phi_j) + B_{ij} \sin(\phi_i - \phi_j)),
\]

(13)

where \(i = 1, 2, \ldots, n\). \(\phi_i\) and \(\omega_i\) represent the generator rotor angle and speed, respectively. Rotational inertia (\(M_i\)) and damping (\(D_i\)) are the parameters associated with each synchronous machine, while \(P_{m,i}\) is its mechanical power input. \(P_{e,i}\) is the network term that represents the total electrical power output from the generator terminal into the power grid. In the Kron-reduced network representation (each node in the network is a machine), the electrical output is a sum of the total power flowing from the generator to all
Outline of Tutorial:

- Introduction to Koopman Operators
- Introduction to deep dynamic mode decomposition
- **Examples of deepDMD applied to biological systems:**
  - Glycolysis model/network
  - Genetic toggle switch
- Code usage
- Research Application 1: Engineering organism persistence
- Research Application 2: Extracting novel genetic sensors
Deep Dynamic Mode Decomposition Recovers Predictive Models for Multi-Stable Systems

The genetic toggle switch has bistable dynamics:

$$\frac{dx_1}{dt} = \frac{\alpha_1}{1 + x_2^\beta} - \kappa_1 x_1$$
$$\frac{dx_2}{dt} = \frac{\alpha_2}{1 + x_1^\gamma} - \kappa_2 x_2$$

DeepDMD is able to predict bistable behavior as long as it sees at least one trajectory from each basin of attraction.
Deep Input Dynamic Mode Decomposition for Prediction of Operational Envelopes in Synthetic Biology

**Toggle Switch Model with Inputs for Temperature & Chemical Inducers**

\[
\dot{P}(X(t), U(t)) = A(\theta)P(X, U)
\]

\[
\sum_{(x,u)\in\mathcal{X}} \gamma(x)\dot{P}(X, U) = \sum_{(x,u)\in\mathcal{X}} \gamma(x)A(\theta, x)P(X, U)
\]

\[
\frac{d}{dt}E_{P_t}[\gamma(x)] = E_{P_t}[\gamma(x)A(\theta, x)]
\]

where the latter equation can be expressed as:

\[
\frac{d}{dt}E_{P_t}[\gamma(x)] = \sum_{(x,u)\in\mathcal{X}} \gamma(x) \left( \sum_{j=1}^{m} a_j(x - \xi_j, \theta)P \left( \begin{bmatrix} X(t) \\ U(t) \end{bmatrix} = \begin{bmatrix} x \\ u \end{bmatrix} - \xi_j \right) - \sum_{j=1}^{m} a_j(x, \theta)P \left( \begin{bmatrix} X(t) \\ U(t) \end{bmatrix} = \begin{bmatrix} x \\ u \end{bmatrix} \right) \right)
\]

\[
= F(x(t), u(t), \theta)
\]
Deep Input Dynamic Mode Decomposition for Prediction of Operational Envelopes in Synthetic Biology

A new data-driven algorithm for scalable discovery of nonlinear dynamical models with experimental inputs

Discovery of Nonlinear Temperature-Dependent Moment Dynamics from Flow Data

Yeung & Egbert, qBio 2018
Deep Input Dynamic Mode Decomposition for Prediction of Operational Envelopes in Synthetic Biology

Given the dataset

\[ D = \{ x(\theta_1, u_1, t_1), x(\theta_2, u_2, t_2), \ldots, x(\theta_n, u_n, t_n) \} \]

and performance criteria

\[ F(x(u, t, \theta)) \geq b \]

Predict if

\[ F(x^*(u^*, t^*, \theta^*)) \geq b \]

Performance Specification

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>2428</td>
</tr>
<tr>
<td>Dysfunctional</td>
<td>644</td>
</tr>
</tbody>
</table>
Input-Koopman Operators Exist for Certain Classes of Nonlinear Systems

Consider a system of the form:

\[ x_{t+1} = F(x_t, w_t) \]

**Assumptions:**

1) \( F(x_t, w_t) \) is analytic,
2) \( w_t \) are memoryless and independent of \( x_t \)

**Proposition:**

There exists an input-Koopman representation for the system of the form:

\[ \psi_x(x_{t+1}) = K_x \psi_x(x_t) + K_u \psi_u(u_t) \]

where \( u_t \) is a vector function consisting of univariate terms in \( w_t \) and multivariate \( w_t, x_t \) terms.
Deep Input Dynamic Mode Decomposition for Controller Synthesis in the Presence of Uncertainty

Price spoofing strategy:

\[ h(u) = u + \alpha \bar{u}, \]

with \( \alpha > 0 \) and \( \bar{u} \) an unknown price offset.

We suppose market, spoofing, generator, and power network dynamics are unknown and wholly represented by the data:
Outline of Tutorial:

- Introduction to Koopman Operators
- Introduction to deep dynamic mode decomposition
- Examples of deepDMD applied to biological systems:
  - Glycolysis model/network
  - Genetic toggle switch
- Code usage
  - Research Application 1: Engineering organism persistence
  - Research Application 2: Extracting novel genetic sensors
The Code and How to Use It

0) Go to the Yeung Repo:

https://github.com/YeungRepo/darpa-sd2/tree/master/deepDMD

1) Save a binary file DATA.pickle with your snapshots

2) Run python3 gen_control_Koopman.py DATA.pickle DEPTH WIDTH DD_SIZE EPOCHS BATCHSIZE PLOT_BASIS OUTPUT_DIRECTORY_PATH DEPTH_CONTROL DDSIZE_CONTROL

(also can change tf.device('/cpu:0') to tf.device('/gpu:[1-8]'))

3) Access your Tensorflow session and visualize interactively with Gen_Pred_plots.ipynb

Deep H2 and H-oo synthesis code in development, release to come out soon (see H2synth_LMI.ipynb for an early version)
Outline of Tutorial:

• Introduction to Koopman Operators
• Introduction to deep dynamic mode decomposition
• Examples of deepDMD applied to biological systems:
  • Glycolysis model/network
  • Genetic toggle switch
• Code usage
• Research Application 1: Engineering organism persistence
• Research Application 2: Extracting novel genetic sensors
Engineering organism persistence in complex communities is an open problem.
How do we quantify organism fitness as a function of environmental inputs and genetic
How do we quantify organism fitness as a function of environmental inputs and genetic factors?

- **Nutrient Rich**: Initialization → Early-Mid Log → Saturation → High Fitness
- **Nutrient Starved**: Initialization → Early-Mid Log → Saturation → Restrained Fitness
How do we quantify organism fitness as a function of environmental inputs and genetic factors?

Nutrient Rich

Nutrient Starved

Adversarial Conditions

Initialization → Early-Mid Log → Saturation

High Fitness

Restrained Fitness

Reduced Fitness
How do we quantify organism fitness as a function of environmental inputs and genetic factors?

Environment with quantified chemical cues

Input A + Input B

Initialization → Early-Mid Log → Saturation

Unknown Fitness
Can we use data-driven models that predict genetic control knobs for cell fitness?
Data-driven systems-level models capture whole system response

Given a system model:

\[ x_{t+1} = Ax_t + Bu_t \]

where:
- \( x_t \) is a vector of RNA counts
- \( u_t \) is a vector of input conditions

We use input dynamic mode decomposition to estimate a model:

\[
\min_{A,B} \|X_f - (AX_p + BU_p)\|_F + \lambda \|[A \ B]\|_1
\]

subject to:

\[
X_p = [x_0 \ x_1 \ \ldots \ x_{N-1}]
\]

\[
X_f = [x_1 \ x_2 \ \ldots \ x_N]
\]

\[
U_p = [u_0 \ u_1 \ \ldots \ u_{N-1}]
\]

Varied condition space to excite system modes

Time-series transcriptomics data

Data-driven models
Organism fitness assays reveal conditions that alter fitness.

Optical density kinetics reveal maximum & minimum growth conditions.
Can we identify *some* causal genetic control knobs from *mostly* correlated states?
Sensor fusion of RNAseq and fitness curves constrains a full state, input-output model

Replicates are grown in identical conditions as fitness assays and pooled for RNAseq

<table>
<thead>
<tr>
<th></th>
<th>MAX</th>
<th>NC</th>
<th>MIN</th>
</tr>
</thead>
</table>
| [Casein] (g/)| 3.5 | 0   | 112.
| [Glucose] (g/)| 0.14| 0   | 15  |

Distinct measurements with conformal experimental conditions
Sensor fusion of RNAseq and fitness curves constrains a full state, input-output model

Next generation sequencing reveals condition-specific transcriptomic response to differing levels of fitness

Distinct measurements with conformal experimental conditions

Replicates are grown in identical conditions as fitness assays and pooled for RNAseq

<table>
<thead>
<tr>
<th>Casein (g/)</th>
<th>MAX</th>
<th>NC</th>
<th>MIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glucose (g/)</td>
<td>3.5</td>
<td>0</td>
<td>112.</td>
</tr>
<tr>
<td>0.14</td>
<td>0</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

Maximum Growth Rate Condition: 2797 genes

Minimum Growth Rate Condition: 2797 genes

[Graph showing growth curves and RNAseq data]
Sensor fusion of RNAseq and fitness curves constrains a full state, input-output model

Fitness-constrained input dynamic mode decomposition:

**Stage 1 Optimization:**

\[
\min_{A,B} \| X_f - (AX_p + BU_p) \|_F + \lambda_1 \| [A \ B] \|_1 \\
\text{s.t.} \quad X_p = [x_0 \ x_1 \ \ldots \ x_{N-1}] \\
X_f = [x_1 \ x_2 \ \ldots \ x_N] \\
U_p = [u_0 \ u_1 \ \ldots \ u_{N-1}]
\]

**Stage 2 Optimization:**

\[
\min_C \| Y_{tot} - CX_{tot} \|_F + \lambda_2 \| C \|_1 \\
\text{s.t.} \quad Y_{tot} = [y_0 \ y_1 \ y_2 \ \ldots \ y_N] \\
X_{tot} = [x_0 \ x_1 \ x_2 \ \ldots \ x_N]
\]

Distinct replicates are grown in identical conditions as fitness assays and pooled for RNAseq.

2797 genes sorted by abundance in min condition.

Next generation sequencing reveals condition-specific transcriptomic response to differing levels of fitness.

Distinct measurements with conformal experimental conditions.

Fitness-constrained dynamical network model.
Forward simulation allows us to rank genetic targets for engineering fitness.

Fitness boost & defect scores can be assigned as L2 deviations from the wild-type strain.
Forward simulation allows us to rank genetic targets for engineering fitness.

Fitness boost & defect scores can be assigned as L2 deviations from the wild-type strain.

We computed 317 model predictions (<672 µs) of single gene knockouts for boosting or reducing fitness; the top 20 are displayed here.

Ontology & functional roles of identified targets:

- PP_0813 cycB: Aerobic respiration
- PP_5313 hupA: DNA binding
- PP_2438 atx
- PP_4189 sucA: TCA cycle
- PP_4947 putA: Oxidizes proline to glutamate (source of C and N)
- PP_2388 LysE: Amino acid transport
- PP_0295 cbcW: Transmembrane transport
- PP_2664: Phosphorelay sensor kinase activity
- PP_1033: Sulfuric ester hydrolase activity
- PP_2673

<table>
<thead>
<tr>
<th>Locus tag</th>
<th>Gene</th>
<th>Gene function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP_0813</td>
<td>cycB</td>
<td>Aerobic respiration</td>
</tr>
<tr>
<td>PP_5313</td>
<td>hupA</td>
<td>DNA binding</td>
</tr>
<tr>
<td>PP_2438</td>
<td>atx</td>
<td></td>
</tr>
<tr>
<td>PP_4189</td>
<td>sucA</td>
<td>TCA cycle</td>
</tr>
<tr>
<td>PP_4947</td>
<td>putA</td>
<td>Oxidizes proline to glutamate (source of C and N)</td>
</tr>
<tr>
<td>PP_2388</td>
<td>LysE</td>
<td>Amino acid transport</td>
</tr>
<tr>
<td>PP_0295</td>
<td>cbcW</td>
<td>Transmembrane transport</td>
</tr>
<tr>
<td>PP_2664</td>
<td></td>
<td>Phosphorelay sensor kinase activity</td>
</tr>
<tr>
<td>PP_1033</td>
<td></td>
<td>Sulfuric ester hydrolase activity</td>
</tr>
<tr>
<td>PP_2673</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Time-series DNAseq of randomly barcoded transposon libraries allow us to quantify relative model fitness of gene knockouts

Deutschbauer RB-TnSeq library of 150,000 barcoded mutants of *Pseudomonas putida KT2440*

*from Wetmore et al. 2015*
Deutschbauer RB-TnSeq library of 150,000 barcoded mutants of *Pseudomonas putida KT2440*

Time-series DNAseq on 140 samples of the entire library allows us to quantify *actual* relative fitness

---

*Time-series DNAseq of randomly barcoded transposon libraries allow us to quantify relative model fitness of gene knockouts*

---

*from Wetmore et al. 2015*
Time-series DNAseq of randomly barcoded transposon libraries allow us to quantify relative model fitness of gene knockouts.

Deutschbauer RB-TnSeq library of 150,000 barcoded mutants of *Pseudomonas putida KT2440*

Time-series DNAseq on 140 samples of the entire library allows us to quantify actual relative fitness.

>40x increase in rate of identification of successful over random selection.

Data-driven model:
- p(Booster): 90%
- p(Defect): 87.5%

Random selection:
- p(Booster): 0.45%
- p(Defect): 2.4%

from Wetmore et al. 2015
Outline of Tutorial:

- Introduction to Koopman Operators
- Introduction to deep dynamic mode decomposition
- Examples of deepDMD applied to biological systems:
  - Glycolysis model/network
  - Genetic toggle switch
- Code usage
- Research Application 1: Engineering organism persistence
- Research Application 2: Extracting novel genetic sensors
Transcriptional sensors underlying response to environmental perturbations can be extracted with DMD.

**Input**
- Time-series culture harvesting
- Transcriptome perturbation
- Measure transcript abundance

**Data-driven observer design**
- Dynamic mode decomposition
  \[ \approx \mathbf{K} \]
  \[ t = 0 : m - 1 \]
  \[ t = 1 : m \]
- Maximize observability
  \[ \mathbf{y}_t = \mathbf{w}^T \times \mathbf{K}^t \times \mathbf{Z}_0 \]
  Gene sampling weights
  \( N \) synthetic initial conditions

**Cell state reconstruction**

**Outcomes**
- Synthetic sensor promoter library
  - Reporter\(_1\)
  - Reporter\(_2\)
- Reporter activity
- Reporter\(_N\) → Time
- Virtual sensor
Dynamic Mode Decomposition provides interpretable and predictive model of gene expression dynamics.
Observability maximization on the model provides a ranking scheme for prioritizing genetic sensing elements.
We identified and designed 15 genetic sensors, extracted from the genome of *Pseudomonas fluorescens SBW25*
We identified and designed 15 genetic sensors, extracted from the genome of *Pseudomonas fluorescens SBW25*.
Transfer curves of the 15 genetic sensors show a range of responses that can be used for biosensing applications.
Virtual superposition of the 15 experimental strains realizes “idealized” sensors.
DMD-Extracted Biosensors Detect Malathion Amid Pesticide Runoff in Environmental Samples

**Diagram:**
- **a:** Illustration of the process of collecting irrigation flow-through and filtering samples.
- **b:** Graphs showing the working concentration of Spectracide and inferred malathion concentration.
- **c:** Bar charts depicting the concentration levels of different proteins.

**Text:**
- Malathion reporters
- Collect irrigation flow-through
- Filter and induce
- Infer malathion concentration

**Graphs and Legends:**
- Working concentration of Spectracide
- Inferred malathion concentration (μM)
Summary of Tutorial Activities

- Introduction to Koopman Operators
- Introduction to deep dynamic mode decomposition
- Code usage
- Examples of deepDMD applied to biological systems:
  - Glycolysis model/network
  - Genetic toggle switch
- Research Application 1: Engineering organism persistence
- Research Application 2: Extracting novel genetic sensors
**Acknowledgements**

The Biological Control, Computing, and Learning Laboratory (BCCL)

**Aqib Hasnain**  
Alec Taylor  
**Dennis Joshy**  
**Charles Johnson**  
Jamiree Harrison  
**Shara Balakrishnan**  
Ines Bilkic (alumnus)  
Jackson Bright (alumnus)  
YiLing Yang (alumnus)  
Dean Passanisi (alumnus)  
Andy Cai (alumnus)  
Josh Marquardt (alumnus)

**Collaborators**

Robert Egbert (PNNL)  
Adam Deutschbauer (LBNL)  
Subhrajit Sinha (PNNL)  
Diveena Becker (Ginkgo Bioworks)  
Paul Maschoff (Ginkgo Bioworks)  
Narendra Mahesri (Ginkgo Bioworks)  
Chris Voigt (MIT)  
Yuval Dorfan (Zvi Meitar Institute, Israeli Syn Bio Institute)  
Amin Espah Borujeni (MIT)  
Ben Gordon (MIT/Broad Foundry)  
Joshua Urrutia (Texas Advanced Computing Center, UT Austin)

**Funding**