Data-Based Control

Egemen Kolemen
Associate Professor
Princeton University / PPPL
And ITER Fellow in Control
7 Heavenly Bodies
Armillary Sphere (Geocentric)
Better System Identification: Geocentric ➔ Heliocentric

Retrograde motion of Mars

Epicycles (~Fourier)
Better System Identification: Geocentric ➔ Heliocentric
System Id: Identify the internal dynamics and effect of the actuator on the state of a system.

\[ y = f(x, u) \quad \text{or} \quad \dot{x} = f(x, u) \]

\[ y = g(x, u) \]

\[ \Rightarrow \ddot{r} = \frac{GM}{r^2} \]
If the Aim is Control: >99% use PID! ➔ Need Data to Tune P, I and D only.

- For >99% industrial control PID is enough!
- For most control, you need enough info (System Identification) to tune a PID.
- System Id to get enough data/info to tune P, I and D.
• For >99% industrial control PID is enough! For most control, you need enough info (System Identification) to tune a PID.
• System Id: First order ODE with time delay
  \[ \dot{y}(t)T + y(t) = Ku(t - L) \]
  \[ y(t) = K \cdot (1 - e^{-\frac{(t-L)}{T}}) \]

• Classic Reaction Curve Method

• Problem:
  – Many shots needed
  – Need the actuator in open loop.
For >99% industrial control PID is enough! For most control, you need enough info (System Identification) to tune a PID.

System Id: First order ODE with time delay

\[ \dot{y}(t)T + y(t) = K u(t - L) \]

\[ y(t) = K \cdot (1 - e^{-(t-L)/T}) \]

• Classic Reaction Curve Method

• Problem:
  - Many shots needed
  - Need the actuator in open loop.
• [P,I,D] control can be designed are functions of K, L, T.

\[ \dot{y}(t) T + y(t) = K u(t - L) \]

\[ y(t) = K \cdot (1 - e^{-(t-L)/T}) \]

**Example Tuning Chart**

<table>
<thead>
<tr>
<th>Cohen Coon</th>
<th>( K_c )</th>
<th>( T_i )</th>
<th>( T_d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>( \frac{1}{a} \left( \frac{1 + 0.35\tau}{1 - \tau} \right) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI</td>
<td>( \frac{0.9}{a} \left( \frac{1 + 0.92\tau}{1 - \tau} \right) )</td>
<td>( \frac{3.3 - 3.0\tau}{1 + 12\tau} )</td>
<td>( L )</td>
</tr>
<tr>
<td>PD</td>
<td>( \frac{1.24}{a} \left( \frac{1 + 0.13\tau}{1 - \tau} \right) )</td>
<td>( \frac{2.5 - 2.0\tau}{1 - 0.39\tau} )</td>
<td>( L )</td>
</tr>
<tr>
<td>PID</td>
<td>( \frac{1.35}{a} \left( \frac{1 + 0.18\tau}{1 - \tau} \right) )</td>
<td>( \frac{2.5 - 2.0\tau}{1 - 0.39\tau} )</td>
<td>( L )</td>
</tr>
</tbody>
</table>

\( \tau = \frac{L}{(L+T)} \); \( a = K_{\text{process}} \frac{L}{T} \)
Closed Loop System ID: Closed Loop Auto-tune PID with Relay Feedback

- The closed-loop plant response period ($P_u$) & amplitude ($A$) give (for example):

$$[P,I,D]=4h/(\pi A)[0.6, 2/P_u, P_u/8]$$

- Advantages:
  - Only a single experiment is needed to tune many different regimes.
  - Closed loop:
    1. More stable
    2. Enable tuning for actuator that can’t be open loop (e.g.: Vertical Ctrl, EFC). Methods exist to join with the existing control
• Pseudo-random binary sequence
• In essence multiple reaction curves
• SISO $\Rightarrow$ MIMO
• Less noise, but plasma might be evolving

Shot # 93298

D. Moreau, “Model-Predictive Kinetic Control Experiments on EAST”, IAEA, 2021
MIMO Linear System Experimental System ID

- Obtain $u$ and $y$ measurements
- MIMO Linear State Space Model can then be obtained by least squares methods
- model = ssest(data,n); %Matlab command
- Then use a linear control (e.g. LQR)

\[
\begin{align*}
x_{k+1} &= Ax_k + Bu_k \\
y_k &= Cx_k + Du_k
\end{align*}
\]
Data-Based Control for Fusion

- Fusion plasmas/reactors has very complicated nonlinear physics
- There is a lot of diagnostic measurement
- Prime target for data-based control design!
Fusion Has Huge Amounts of Nonlinear Data: How to Utilize This for Control?

- How can we bring this immense information into control?
  - Many not available or usable in real-time (RT)
  - Too much data to pass to a central CPU
  - Mostly not automated: Post-discharge analysis by physicists

- Machine Learning ➞ RT data featurization + automated analysis + control design
Machine Learning for Real-time Fusion Plasma Behavior Prediction and Manipulation

- DOE ML for Fusion report identified ML for Fusion Control as priority
- A multi-institutional collaboration of a) ML/AI Scientists, b) Diagnosticians and c) Fusion Control
- Experts:
  - CMU, PPPL, PU, SLAC and UW-M
Extend RT control to fast plasma dynamics and fluctuation diagnostics

- Fluctuation diagnostics capture fast plasma dynamics with MHz sampling
- RT calculations on ~10-100 signals at ~1 MHz is not feasible on CPU/GPU → requires FPGA at sensors – "Edge ML"
  - Highest throughput on FPGA imposes restrictions on algorithm/model architecture
  - Calculation output captures information about fast plasma dynamics and output is available to downstream plasma control system
- Diagnostics: BES, interferometers, ECE
- Physics applications: ELM onset prediction, characterization of turbulence and confinement mode, Alfven eigenmodes
![What is machine learning?](image)

Consider \( y = f(x) \)

<table>
<thead>
<tr>
<th></th>
<th>Known</th>
<th>Known</th>
<th>Want to Find</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Standard” problem</td>
<td>( f(x) )</td>
<td>( x )</td>
<td>( y )</td>
</tr>
<tr>
<td></td>
<td>(function)</td>
<td>(input)</td>
<td>(output)</td>
</tr>
<tr>
<td>“Inverse” problem</td>
<td>( f(x) )</td>
<td>( y )</td>
<td>( x )</td>
</tr>
<tr>
<td></td>
<td>(function)</td>
<td>(output)</td>
<td>(input)</td>
</tr>
<tr>
<td>“Learning” problem</td>
<td>( x )</td>
<td>( y )</td>
<td>( f(x) )</td>
</tr>
<tr>
<td></td>
<td>(input)</td>
<td>(output)</td>
<td>(function)</td>
</tr>
</tbody>
</table>

- **Know data + function, want to find more data**
- **Only know data, need to find the right function**
So you have some data
Linear modeling
And it mostly worked!
A bit harder
Linear fit: not so great...
Oh hey it looks quadratic:
Gray-box system identification, give the structure
It really is!
Let’s go really nonlinear
Polynomial fit: nope
Sine wave: better, but still nope
Maybe a Neural Network?

Weights / Parameters

\[ z_1 = a_1 x + b_1 \]
\[ z_2 = a_2 z_1 + b_2 \]
\[ z_3 = a_3 z_2 + b_3 \]
\[ y = a_4 z_3 + b_4 \]

\[ y = f(x) \]
Maybe a Neural Network?

Weights / Parameters

\[ \begin{align*}
    z_1 &= a_1 x + b_1 \\
    z_2 &= a_2 z_1 + b_2 \\
    z_3 &= a_3 z_2 + b_3 \\
    y &= a_4 z_3 + b_4
\end{align*} \]

Activation Functions

- **Sigmoid**: \( \sigma(x) = \frac{1}{1+e^{-x}} \)
- **tanh**: \( \tanh(x) \)
- **ReLU**: \( \max(0, x) \)
- **Leaky ReLU**: \( \max(0.1x, x) \)
- **Maxout**: \( \max(w_1^T x + b_1, w_2^T x + b_2) \)
- **ELU**: \( \begin{cases} 
    x & x \geq 0 \\
    \alpha(e^x - 1) & x < 0 
\end{cases} \)

\[ y = f(x) \]
Works well!
Works well!
What is Machine Learning?

- **Traditional programming**

  Data → Computer → Label

- **Machine Learning programming**

  Data → Computer → Program → Label
Neural network architectures (famous ones)

**Multilayer perceptron (MLP)**
- Fully connected layers
- One or more hidden layers

**Convolutional neural network (CNN)**
- Convolve multiple kernels over image to build feature space
- Use feature space for classification with MLP (a common recipe in DL)

**Autoencoder (AE)**
- Output is a reconstruction of input
- Latent space provides low-dimensional representation

**Recurrent neural network (RNN)**
- History-dependent internal state provides a mechanism for memory
- Long short-term memory (LSTM), gated recurrent unit (GRU)
Machine Learning Types

- **Supervised**: learning with labeled data
  - Example: image classification
  - Example: regression for predicting real-valued outputs

- **Unsupervised**: discover patterns in unlabeled data
  - Example: cluster similar data points

- **Reinforcement learning**: learn to act based on feedback/reward
  - Example: learn to play Go

Slide credit: Ismini Lourentzou – Introduction to Deep Learning
1. Sample image and label
Training the NN

1. Sample image and label
2. Pass image through network to get loss (forward)
1. Sample image and label
2. Pass image through network to get loss (forward)
3. Backpropagate to get gradients (backward)
Training the NN

1. Sample image and label
2. Pass image through network to get loss (forward)
3. Backpropagate to get gradients (backward)
4. Take step along negative gradients to update weights
Training the NN

1. Sample image and label
2. Pass image through network to get loss (forward)
3. Backpropagate to get gradients (backward)
4. Take step along negative gradients to update weights
5. Repeat!
Example Classifier NN

Is this a **Cat** or **Dog**?

![Image of a cat and a dog in a laundry basket](https://www.cs.umd.edu/class/fall2018/cmsc320)
ML Control for Fusion: ML as a measurement going into human designed control.

Diagram:
- Desired output
- Controller
- Plant/Process
- Actual output
- Measurement Device

ML as eyes
ML use in Fusion: Event Prediction

- Simplest and first ML use were Event Prediction
- $x=0D$ signals, $y=\text{Disruption } [0,1]$
- Convert to probability

---

### Signal description

- Electron density, $n_e \text{ (m}^{-3})$
- Plasma current, $I_p \text{ (A)}$
- Squareness, $\zeta$
- Plasma minor radius, $a$
- Normalised internal inductance, $\xi_i$
- Stored plasma energy, $W_{MHD} \text{ (J)}$
- Elongation, $\kappa$
- Triangularity, $\delta$

---

DIII-D Shot = 180808, Time = 5.150s
$I_p=1.252 \text{ MA, } W_{MHD}=0.469 \text{ MJ}$

---

# 174724

---

Prediction time

Boyer, NF 2022 + Fu PoP 2020
ML to Predict + Feedforward Control (Rampdown Scenario Change)

- ML algorithm to find the disruption time.
- Use it to change the off-normal response:
  - New ramp down sequence of the plasma (feedforward) when disruption is expected
  - Reduce disruption impact ($I_p$ at disruption)

Fu et al. PoP 2020
ML gives a Feedback Control Target:
ML Predict Stability (Tearing) ➔ Optimize Performance

1. Predict Instability using ML
2. Instability Starts
3. Disruption Occurs

- Instability prediction gives lots more time than disruption prediction
- Enough time to control the plasma and avoid shutdown
- Choose a stability level to operate at (say %1)
- Then Machine Learning Controls the NBI/ECH for highest performance at that stability level
- PCS Algorithm (Kolemen, Yu, Boyer, Erickson)
- Fu et al. PoP 2020
NN-based AE detector: Detecting Alfvén Eigenmode Using ECE

- Input: **40 raw ECE signals** (No preprocessing!)
- Output: Score of five AE modes (LFM, BAE, EAE, RSAE, TAE) per time step
- ML model: Reservoir Computing Network (RCN)
NN-based AE detector: Detecting Alfvén Eigenmode Using ECE

True Positive Rate: 91%
False Positive Rate: 7%
NN-based AE localizer:
Detecting and Locating Alfvén Eigenmode Using ECE

- Input: Spectrogram of each ECE channel
- Output: Probability of AE modes per ECE per time step

A. Kaptanoglu et. al., 2022
Nucl. Fusion (Accepted)
NN-based AE localizer: Detecting and Locating Alfvén Eigenmode Using ECE

A. Kaptanoglu et al., 2022
Nucl. Fusion (Accepted)
NN-based AE localizer:
Detecting and Locating Alfvén Eigenmode Using ECE

- Control aim ECH to the location to suppress AE
NN-based AE localizer: Detecting and Locating Alfvén Eigenmode Using ECE

Shot # 146079, fr# 0/3905, t:0ms

TOP: Original spec., BOTTOM: Denoised spec.
Plasma Evolution Modeling: Plasma Profile

- Define the state of the plasma by 1D profiles.
- Due to symmetry and high transport along flux sections.

Full state of plasma determined by 1D profiles:
- Pressure ($P$)
- Current ($J$)
- Electron temperature and density ($T_e, n_e$)
- Ion temperature and density ($T_i, n_i$)
- Rotation ($\Omega$)
ML as Plasma Evolution Model:
ML Profile Prediction

- **ML-Based Prediction of Profile Evolution.**
- **Input:** DIII-D Historic Data -- 2013-2019 (5 Profiles, Shape, NBI, Density,...)
- **Output:** Profile Evolution predicting NN

Abbate, Conlin NF 2021
Replaying a test set shot for BetaN, Li prediction (Ian Char, CMU)

Ground Truth

Predictions
Replaying a more interesting test shot (Ian Char, CMU)
Realtime Adaptive ML Plasma Model: Reservoir Computing Network (RCN)

A **recurrent** neural network with **random** and **sparsely connected** early layers. Only the last layer is trained using **linear regression**.

**Specifications of RCN:**
- Projects the inputs to a random very high-dimensional space.
- Ability to process **temporal information** (time-series data analysis)
- Much **faster** and **easier training** procedure compared to DNNs.
  - LSTM: **5 hours** on GPU
  - RCN (with similar performance to LSTM): **4 Minutes** on CPU
  - Easy & fast training makes “in-situ” **model adaptation** possible
Adaptive Data-driven Profile Prediction Model

<table>
<thead>
<tr>
<th></th>
<th>CNN/LSTM</th>
<th>RCN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training</strong></td>
<td>5h on GPU</td>
<td>45s on CPU</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>SOTA</td>
<td>Close to DNN</td>
</tr>
<tr>
<td><strong>Adaptation</strong></td>
<td>Difficult</td>
<td>Easy (100ms)</td>
</tr>
</tbody>
</table>

Real-time Adapting every 500ms

ML as a Surrogate for Physics code: EQNet, RT Equilibrium Reconstruction with ML

- Data does not have to be experimental! It can simulation as well.
- We developed a NN that represents G-S Reconstruction for NSTX-U
- More accurate than rt-EFIT (<1ms)

ML as a Surrogate for Physics code: Faster, More Robust

- Neural net responds better to dynamic changes and induced vessel currents than online method.
- Faster: Removes 5ms phase delay during oscillations – improves controller chance for recovery.
- Robust: Trained with all good and bad sensors. Losing a sensor degrades prediction BUT does not fail.
- ML can output Linear State Space System which can then be used in control.
ML Control for Fusion: ML to produce model for control

Desired output → Controller → Plant/Process → Actual output

Measurement Device

ML as model to use in Control
Plasma dynamics are nonlinear - use ML to get linear model

- Traditional linearization uses Jacobian of $f$
  - Only valid locally (perhaps very locally)
- Alternative - Learn nonlinear embedding to linear space
  - Linearly Recurrent Autoencoder Network (LRAN)
- Can be valid globally
  - S. E. Otto and C. W. Rowley, 2019
Model trained on experimental data from DIII-D 2013-2018 give similar prediction as before

Linear model can be used in control
Model Predictive Control: Convert cost function to the “standard form” of a quadratic program

Control system formulation

\[ J_k = \sum_{i=1}^{N} x_{k+i}^T Q x_{k+i} + u_{k+i-1}^T R u_{k+i-1} \]

Subject to:
- Dynamics constraint
  \[ x_{i+1} = Ax_i + Bu_i \]
- Input constraints
- Output constraints

LQR is a special case of this with \( N = \infty \) and no input or output constraints

Standard form for quadratic programs

\[ J_k(z) = z^T H z + f^T z \]

Subject to:
\[ A_{eq} z = b_{eq} \]
\[ A_{ineq} z \leq b_{ineq} \]
Model Predictive Control: Convert cost function to the “standard form” of a quadratic program

- Predict state
- Find the new constraints
- Solve Quadratic Program
- Find the discrete control action
- Calculations can be done fast enough in tokamaks
Example: inverted pendulum on a cart with wall constraint

Motivated by Steve Brunton Control Bootcamp (youtube)
Example: MPC for shape and secondary x-point control

MPC controller guides x-point while maintaining power supply and boundary constraints

### Activated Constraints

<table>
<thead>
<tr>
<th>Coil #</th>
<th>PF Coils</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I (kA)</td>
<td>V (kV)</td>
</tr>
<tr>
<td>PF1</td>
<td>&lt; 48</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>PF2</td>
<td>&lt; 55</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>PF3</td>
<td>&lt; 55</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>PF4</td>
<td>&lt; 55</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>PF5</td>
<td>&lt; 52</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>PF6</td>
<td>&lt; 52</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>CS1L</td>
<td>&lt; 45</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>CS1U</td>
<td>&lt; 45</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>CS2L</td>
<td>&lt; 45</td>
<td>&lt; 1.5</td>
</tr>
<tr>
<td>CS3L</td>
<td>&lt; 45</td>
<td>&lt; 1.5</td>
</tr>
</tbody>
</table>

---

ML Control for Fusion:
ML as end-to-end control

Desired output → Controller + Actual output

Controller → Plant/Process

Measurement Device

Reinforcement Learning
ML does everything: measurement + prediction + control
Learning to Control From Data: Model Predictive Control (MPC) vs Reinforcement Learning (RL)

**Model Predictive Control (MPC)**

- Observe current state of the tokamak
- Search for action that has most benefit for the next few timesteps
- Tokamak

**Reinforcement Learning (RL)**

- Collect experience from the model and find the best actions.
- Neural Net Controller
- Model

**Beforehand**

- Observe state
- Neural Net Controller
- Neural net forward pass to get action

**Online**

- Tokamak
Reinforcement learning

- How do we (humans) learn to solve problems?
  - Trial and error interaction with the environment

- Reinforcement learning (RL) is a general framework to express how this process is performed.

- There are two important aspects to the paradigm
  - It allows us to specify the goal (Reward function)
  - It can deal with long-term dependencies

- [Links to various deep learning blogs and articles](https://www.deepmind.com/blog)
Reinforcement learning versus other learning

- Reinforcement learning: Learn by trial-and-error how to act on an environment to achieve high reward
  - Exploration to gather experience + learning from the experience

Recent applications of RL to tokamak control:
- 0D parameter control in KSTAR [Seo et al, NF 2021]
- Safety factor control [Wakatsuki et al, NF 2019]
- Ion profile control [Wakatsuki et al, NF 2021]
- Beta, Profile control [Char APS'21, Mehta APS'21]
- Shape Control
- [J. Degrave, F. Felici et al. Nature 2022]

See also: [Sutton and Barto, Reinforcement Learning, an Introduction. MIT Press]
Reinforcement Learning

• Deep reinforcement learning (RL) [1] “Finding the best decision-making policy”

<table>
<thead>
<tr>
<th>Action</th>
<th>Bad</th>
<th>Good</th>
<th>Bad</th>
<th>Good</th>
<th>Good</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reward</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>

- The AI takes the action that would yield a higher reward.
- It gradually learns the optimal decision-making policy via experiences.

• Difficulties of RL application in fusion research

- RL requires a reliable training environment (simulation).
  
  But we don’t have a perfect all-in-one simulator including gyro-kinetic, MHD stability, H&CD, …

- RL typically requires $>10^5$ simulation iterations to train.
  
  But reliable theory-based simulation (TGLF, EPED, …) takes minutes to hours for a single step calculation.
Use ML Plasma Evolution Model: DIII-D training a policy for high $\beta_N$

- How does this work? Simple example
- Train a policy using to maximize $\beta_N$
- This controller observes all scalar information about the plasma and is allowed to control the beams, current, density, and shape parameters.
- Rotation profile control is tested on DIII-D.
- J. Schneider, I. Char, V. Mehta et al.

Red lines are RL Policy
Blue lines are historical replay
Use the ML Plasma Model for RL at KSTAR

- RL validation in the exp. databased simulation

- After enough training, the AI determines reasonable solution of $I_p$ and the boundary shape to reach the target of multiple parameters.

- You can try it out at https://github.com/jaemin-seo/AI_tokamak_control if interested.
KSTAR Operation Design with RL

- Validation in the KSTAR experiment

Target setting

\((\beta_p, q_{95}, l_i)\)

= \(1.8, 6.0, 0.9\)

& \(1.2, 4.0, 1.0\)

- After we set the target, the AI determines the operation trajectory. Then Jaemin an the experiment with that trajectory.

- The plasma response followed the preset targets.

Simulation based RL for Fusion Control: Free boundary Shape Control

- Free-boundary simulator
- Solve coupled system of equations:
  - Grad-Shafranov equation
  - Circuit equations for time evolution of currents
- Simple (feedforward, small) actor network to run in real time
- Complex (recurrent, large) critic network to learn system’s dynamics

![Diagram showing the learning loop and simulated environment](image-url)
Various plasma shapes controlled in TCV with reinforcement Learning

J. Degrave, F. Felici et al. Nature 2022
Opening new frontiers for TCV: droplet plasmas

TCV#69545

Time since breakdown (s)

$Z_a (m)$

$I_p (kA)$

J. Degrave, F. Felici et al. Nature 2022
### Features of traditional / RL controllers

<table>
<thead>
<tr>
<th>Traditional controllers (MIMO PID)</th>
<th>Our Reinforcement Learning implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need to compute error for each control loop in real time</td>
<td>Single reward function, no explicit error signals or estimation</td>
</tr>
<tr>
<td>Need separate tuning of various control loops, using linear control techniques assuming (local) linearity</td>
<td>Joint solution to entire stabilization/control problem including any nonlinearities</td>
</tr>
<tr>
<td>Need control engineering + tokamak knowledge to break down control problems, design separate controllers</td>
<td>Domain knowledge is in simulator. Just define reward functions</td>
</tr>
<tr>
<td>Tuning of several control parameters</td>
<td>Reward function engineering</td>
</tr>
<tr>
<td>(Usually) Clear relation between parameters and aspects of control performance</td>
<td>Black-box agent</td>
</tr>
<tr>
<td>Integral control nominally gives zero steady-state error on desired quantities</td>
<td>No certainty of zero steady-state errors in case of external disturbances</td>
</tr>
</tbody>
</table>
Final Note on Robustness and Stability of Control

- Lots of people working on that

- Many recent progress by ML researchers (e.g. colleague at Princeton Prof. Anirudha Majumdar)


- Within this decade, my guess is that, we will have widely used metrics similar to the linear theory on the stability/robustness
Conclusions: Data-Based Control will gain more visibility going forward

- Lots of diagnostics information
- ML tools are maturing
- Initial applications of ML Control are begin tested
- They show promise
- We have many undergraduate/graduate students and researchers working together in ML Control Project.
- Need to train more students!
- Interested in Data Science / ML for Fusion contact me ekolemen@Princeton.edu