Proceedings of Machine Learning Research 176:246–258, 2022 NeurIPS 2021 Competition and Demonstration Track

# Learning by Doing: Controlling a Dynamical System using Causality, Control, and Reinforcement Learning

**Sebastian Weichwald** SWEICHWALD@MATH.KU.DK
*Department of Mathematical Sciences, University of Copenhagen, Denmark*

**Søren Wengel Mogensen** SOREN.WENGEL_MOGENSEN@CONTROL.LTH.SE
*Department of Automatic Control, Lund University, Sweden*

**Tabitha Edith Lee** TABITHALEE@CMU.EDU
*The Robotics Institute, Carnegie Mellon University, USA*

**Dominik Baumann** DOMINIK.BAUMANN@IT.UU.SE
*Department of Information Technology, Uppsala University, Sweden*

**Oliver Kroemer** OKROEMER@CMU.EDU
*The Robotics Institute, Carnegie Mellon University, USA*

**Isabelle Guyon** GUYON@CHALEARN.ORG
*LISN/INRIA/CNRS, Université Paris-Saclay, France, and ChaLearn, USA*

**Sebastian Trimpe** TRIMPE@DSME.RWTH-AACHEN.DE
*Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Germany*

**Jonas Peters** JONAS.PETERS@MATH.KU.DK
**Niklas Pfister** NP@MATH.KU.DK
*Department of Mathematical Sciences, University of Copenhagen, Denmark*

**Editor:** Douwe Kiela, Marco Ciccone, Barbara Caputo

## Abstract

Questions in causality, control, and reinforcement learning go beyond the classical machine learning task of prediction under i.i.d. observations. Instead, these fields consider the problem of learning how to actively perturb a system to achieve a certain effect on a response variable. Arguably, they have complementary views on the problem: In control, one usually aims to first identify the system by excitation strategies to then apply model-based design techniques to control the system. In (non-model-based) reinforcement learning, one directly optimizes a reward. In causality, one focus is on identifiability of causal structure. We believe that combining the different views might create synergies and this competition is meant as a first step toward such synergies. The participants had access to observational and (offline) interventional data generated by dynamical systems. Track CHEM considers an open-loop problem in which a single impulse at the beginning of the dynamics can be set, while Track ROBO considers a closed-loop problem in which control variables can be set at each time step. The goal in both tracks is to infer controls that drive the system to a desired state. Code is open-sourced (github.com/LearningByDoingCompetition/learningbydoing-comp) to reproduce the winning solutions of the competition and to facilitate trying out new methods on the competition tasks.

**Keywords:** Causality; control; reinforcement learning; dynamical systems; robotics; system identification; chemical reactions

## 1. Introduction

Modeling actively performed changes in an observed system is an important goal that has appeared in various versions and settings in statistics, engineering, and computer science. Each community has developed their own terminology and methods to tackle the specific applications relevant to their disciplines, leading to the emergence of causality, control theory, and reinforcement learning (RL). Each of these fields brings a different perspective to modeling system changes and our goal of the *Learning by Doing* NeurIPS 2021 competition was to bring together researchers from each of these fields to work on the same set of tasks.

We decided to focus on dynamical systems as these appear in all three fields. To offer a sufficiently diverse set of problems, we provided two competition tracks: Track CHEM and Track ROBO. Track CHEM considers the open-loop problem of choosing a single impulse that can be set at the beginning of a chemical reaction with the goal of reaching a specific concentration of a target reactant. Track ROBO considers the closed-loop problem of continuously providing inputs to a robot that guides the tip of the robot to move along a target trajectory. In both cases, participants were given a recorded data set from the systems and needed to use this data to learn how to optimally interact with the system in new settings.

In Section 2, we briefly introduce the three fields, provide the relevant terminology, and discuss how each field models exogenous changes to a system. In Section 3, we introduce Track CHEM and in Section 4 Track ROBO. In both aforementioned sections, we also point out the challenges the tasks pose and how they relate to each field. We conclude in Section 5 with some of the lessons learnt throughout the competition.

The competition website learningbydoingcompetition.github.io provides tutorials, results, and presentations of some of the competing teams. We provide open-source code at github.com/LearningByDoingCompetition/learningbydoing-comp to reproduce the winning solutions and to allow the application of new methods to the competition tasks.

## 2. Causality, Control, and Reinforcement Learning

To foster cross-pollination, we begin by introducing causality, control, and RL and describe how each framework models changes to a system.

**Causality** In classical statistics, a multivariate stochastic system is thought of as describing a single observational distribution. In contrast, a causal system[1] describes a set of distributions that models system behavior not only under passive observation, but also under interventions (Pearl, 2009; Spirtes et al., 2000; Imbens and Rubin, 2015). To make this more precise, assume we observe a response variable $Y$ and a set of predictors $X = (X_1, \dots, X_p)$ and wish to model how $Y$ is affected by interventions on the predictors $X$. We now assume that there exists a subset $\text{PA} \subseteq \{1, \dots, p\}$ of the predictors, called the parents of $Y$, that determine the value of the response $Y$ via the following fixed functional form

$$Y = f(X_{\text{PA}}, \epsilon), \tag{1}$$

1. Here, the notion of a causal system differs from what is usually called a causal system in the systems and control literature where it refers to systems in which outputs only depend on past and current inputs.

where $\epsilon$ is a noise variable. This equation is understood to be functional in the sense that the expected value of $Y$ given that $X_{\mathrm{PA}}$ was set to a fixed value $x_{\mathrm{PA}}$ (denoted by $\mathbb{E}[Y \mid \mathrm{do}(X = x_{\mathrm{PA}})]$) is given by $\mathbb{E}[f(x_{\mathrm{PA}}, \epsilon)]$. Such structural equations are the building blocks of structural causal models (SCMs) (Pearl, 2009; Peters et al., 2017), which is an important class of causal models. Usually, causal models assume some version of stability of mechanisms under interventions (Haavelmo, 1944; Aldrich, 1989). In (1) this corresponds to assuming that $f$ remains fixed under any intervention on $X$. Much research in causality investigates under which conditions the function $f$ and the set of parent variables $X_{\mathrm{PA}}$ can be identified from data and how to do so data-efficiently. Causal models also exist for dynamical systems, for example, a multivariate process $Z(t) = (Z_1(t), \ldots, Z_p(t))$ can be modeled by differential equations of the form $dZ_j(t) = F(Z(t))_j$ for each component. Interventions then correspond to modifying parts of these equations, for example, by fixing one of the coordinate processes at a certain value over a given time interval (see Peters et al., 2022, for an overview). Causal models induce a *graph* over the coordinate processes, which is useful for visualizing a multivariate causal system. In such a graph, each node represents a coordinate process, $Z_i$. We include a *directed edge*, $Z_i \to Z_j$, $i \neq j$, if the right-hand side of $dZ_j(t) = F(Z(t))_j$ is not constant in $Z_i$ and in this case we say that $Z_i$ is a *(causal) parent* of $Z_j$ and that $Z_j$ is a *child* of $Z_i$.

**Control** In automatic control, one assumes that a target process $y : [0, T] \to \mathbb{R}^p$ is generated by a dynamical system. A common setup is a continuous-time state-space model, given by

$$\begin{aligned} \dot{x}(t) &= F(x(t), u(t)) \\ y(t) &= H(x(t), u(t)), \end{aligned} \tag{2}$$

where $x(t)$ is the time-dependent *state* of the systems (which could include $y(t)$ itself), $\dot{x}(t)$ the derivative of $x$ with respect to time, and $u(t)$ is the *control input*. *Control design* is the task of constructing a *controller*, that is, a map from measurements $y(t)$ to control inputs $u(t)$. The dynamical system (2), sometimes called a *plant*, is often nonlinear in the state $x(t)$, and linear in the control input $u(t)$. When both the inputs $x(t)$ and $u(t)$ and the target process are vector-valued, one also uses the term *multi-input multi-output* (MIMO) system.

A typical approach of control engineering for obtaining a controller may consist of the following steps: (1) Use problem insight to select a useful (often parametric) model class (such as linear/nonlinear, auto-regressive, continuous/discrete-time); (2) Fit the parameters of the model (that have not been set yet); (3) Use the model in a control design method to obtain a controller (for example, by minimizing a given loss function); (4) Test the controller on the system or in simulation; (5) Possibly repeat the cycle if results are not satisfactory.

Control design traditionally builds model classes from first principles, for example, laws in chemistry or physics, even though not all parameters may be known. The task of fitting the model from input-output data is known as *system identification* (see, for example, Ljung, 1998). Identifying a system and obtaining a controller is a well-understood problem for linear dynamical systems and we refer the reader to textbooks such as the one by Åström and Murray (2008) for a more detailed introduction into system identification and control design.

For nonlinear systems, however, many design methods exist for specific problem settings, but they lack the generality of approaches for linear systems. Furthermore, in some practical applications, including the setup of this competition, a derivation from first order principles (including all parameters) may be impossible and even the model class may be unknown. In slight deviation of items (1–3) above, one can attempt to control the system without an explicit model of the underlying dynamics, for example, using a PID controller (Åström and Murray, 2008). Alternatively, *model-predictive control* (Allgöwer and Zheng, 2012; Borrelli et al., 2017) and similar optimization-based controller schemes may not compute a controller explicitly but model the effect of control inputs directly and exploit this model in an online optimization procedure. While model-predictive control typically relies on a given system model with fixed parameters, the field of *adaptive control* (Åström and Wittenmark, 2013) considers settings where model or controller parameters need to be tuned online. More recently, researchers in control have been exploring ways to incorporate data-based and machine-learning approaches into control design, partly making the above pipeline more flexible. This emerging area between control and machine learning is known as *learning-based control* or *data-based control*.

**Reinforcement learning** In RL, one commonly starts by defining a *reward* that specifies how desirable it is to transition from one state to another under a given action. The system is modeled sequentially by explicitly accounting for interactions with the system in each time step. The goal is to learn how to interact with the system in a way that maximizes the expected value of the reward. Mathematically, this can be achieved using Markov Decision Processes (MDPs). An MDP consists of a tuple $(S, A, R, T, \gamma)$ (Sutton and Barto, 1998). $S$ is the set of states of the system and $A$ is a set of actions that the agent can execute. The reward $R(s, a, s')$ expresses the immediate reward for executing action $a \in A$ in state $s \in S$ and then transitioning to the next state $s' \in S$. $T(s'|s, a)$ is the transition distribution which gives the distribution over next states $s'$ given the current state $s$ and action $a$. $\gamma \in [0, 1]$ is a discount factor that expresses the agent's preference for immediate rewards over long-term rewards. Usually, $S$, $A$ and $\gamma$ are known (user-defined), $T$ and $R$ are unknown. To select an action, the agent applies a policy $\pi(a|s)$ that defines the distribution over the next action, $a$, to execute given the current state $s$. Policies can be stochastic or deterministic. The $t$-th sampled transition thus results in a tuple $(s_t, a_t, s'_t, r_t)$, where $s_t$ is the current state, $a_t$ is the sampled action, $s'_t$ is the next state after the transition, and $r_t \in \mathbb{R}$ is the resulting scalar reward. The goal of learning is to acquire an optimal policy, often denoted as $\pi^*$, that maximizes the expected return $\mathbb{E}_{s' \sim T, a \sim \pi}\left[\sum_t^T \gamma^t r_t\right]$ where T is the duration of the task (Bell et al., 1996).

## 3. Track CHEM: Optimally controlling a chemical reaction

Track CHEM tackles the problem of optimally choosing impulses or shocks in a dynamical system to control a specific part of the system. This task is motivated by applications related to chemical reactions in which one is interested in generating a desired concentration of a specific chemical compound by controlling the initial concentration of some other chemical compounds. When considering such systems there are constraints on how and at what cost experimentation can be performed. To reflect this, we only provided participants with

offline training data instead of allowing them to actively interact with the reactions. The task was to extract knowledge from observed experiments, and use it to control the system in previously unseen settings. Methods that tackle this problem may also apply to systems in which experimentation is infeasible and instead only exogenous shocks to the system can be observed and leveraged for learning.

**Background on chemical reaction networks** In a chemical reaction, one set of chemical compounds is transformed into another. We usually say that reactants are turned into products. Reactants and products are both called species. In Track CHEM, the goal is to find optimal controls (or policies or interventions) on the concentrations of reactants to ensure a desired concentration of one of the species. The dynamical behaviour of species concentrations in chemical reactions is modeled by mass-action kinetics (Waage and Guldberg, 1864), which results in an ordinary differential equation (ODE) over the species. During training, the participants were not able to interact with the system and instead only had access to past observations from the system. For these observations, the applied control inputs were known to participants. The goal in this track is to control one specific process in the observed system when provided only with initial observations. We give more details on chemical reaction networks in Appendix A.

**Data generating process** Data is generated by an artificial chemical reaction network. Specifically, a 15-dimensional process $Z(t)_{t\geq 0}$ is generated as:

$$\begin{aligned} Z(0) &= z \\ \dot{Z}(t) &= F(Z(t)) + BU(t), \end{aligned} \tag{3}$$

where $U(t) \in [-10, 10]^8$ is the control input at time $t$, $z \in (0, \infty)^{15}$ is an initial value, $B \in \mathbb{R}^{15\times 8}$ is a matrix specifying how the controls influence the dynamics and $F : \mathbb{R}^{15} \to \mathbb{R}^{15}$ is a function from the function class

$$\mathcal{F} = \left\{ F : \mathbb{R}^{15} \to \mathbb{R}^{15} \,|\, F_\ell(Z) = \sum_{j=1}^{15} \theta_j^\ell Z_j + \sum_{k,j=1}^{15} \theta_{j,k}^\ell Z_j Z_k, \quad \ell = 1, \ldots, 15 \right\}. \tag{4}$$

The parameter $\theta$ satisfies additional constraints since we only consider ODE systems that are generated by converting chemical reactions using the law of mass-action kinetics. Furthermore, the rates of the underlying chemical reactions are non-negative which also adds constraints on the coefficients $\theta$.

Among the 15 species, $Y = Z_{15}$ is the species for which the concentration should be controlled. There are eight controls, $U_1(t), \ldots, U_8(t)$, that affect the concentrations of a subset of the species. A (to participants unknown) graphical representation of the model that generated the data for the competition is given in Figure 1. The model has a simple structure consisting of four blocks of variables: $\{Z_1, Z_2, Z_9, Z_{13}\}$, $\{Z_3, Z_4, Z_{10}\}$, $\{Z_5, Z_6, Z_{11}\}$, and $\{Z_7, Z_8, Z_{12}, Z_{14}\}$. Each block corresponds to an interaction mechanism that can either increase or decrease the concentration of $Y$. There are two types of controls. (1) Control variables $U_1, U_2, U_3$, and $U_4$ affect the system strongly and they affect an increasing and a decreasing block simultaneously. (2) Variables $U_5, U_6, U_7$, and $U_8$ have a weaker effect on the system but they target only an increasing block ($U_7$ and $U_8$) or only a decreasing block

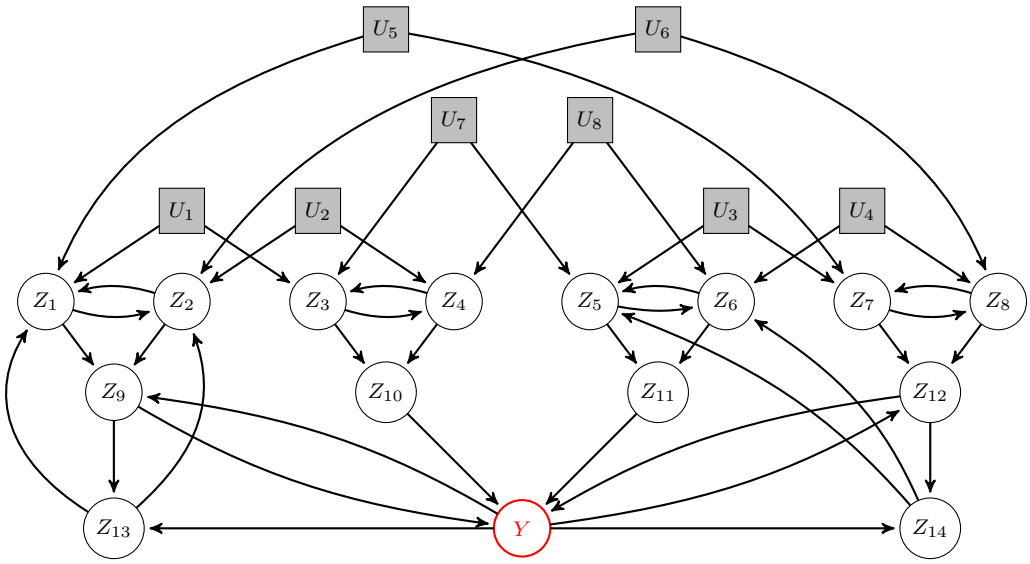


Figure 1: Graphical representation of the chemical reaction in Track CHEM.

($U_5$ and $U_6$). Hence, the controls in (2) offer an easy strategy to control $Y$ but are expensive, while using the controls in (1) is cheaper but might be more difficult. The participants only observed data with pre-specified control settings where we had incorporated confounding structure in the observed controls; the problem therefore also featured a causal challenge.

**Task** Participants knew the function class but did not know the parameters. They did not know the graphical structure of the system, either. Furthermore, participants did not observe the process $Z$ directly. Instead, they only observed $X$, a noisy version of the process, sampled on a time grid $(t_0, \ldots, t_L)$; that is, the observed data is sampled, at $t \in \{t_0, \ldots, t_L\}$, from the process

$$X(t) = (Z_1(t), \ldots, Z_{15}(t)) + N(t), \tag{5}$$

where $N$ is a mean-zero noise process such that $\{N(t) : t = t_0, \ldots, t_L\}$ are independent. The goal of the task is to choose a value $u \in \mathbb{R}^8$ such that the controls

$$U(t) = \begin{cases} u & \text{if } t \in [t_0, t_3) \\ 0 & \text{if } t \geq t_3, \end{cases} \tag{6}$$

lead to $Y$ being close to a (pre-specified) desired value $y_*$ at the end of the reaction. As the value $u$ for the control is set only once and after having observed $X(0)$, Track CHEM is an open-loop problem.

Participants had access to data from 12 different ODE systems which are specified by the functions $F^1, \ldots, F^{12} \in \mathcal{F}$ and they knew the index of the system from which data originated. The function class $\mathcal{F}$ was known but $F^1, \ldots, F^{12}$ were unknown to the participants. Participants knew that the 12 systems had the same structure, that is, the same parameters $\theta_j^\ell$ and $\theta_{j,k}^\ell$ were zero in all 12 systems, and that every $\theta_j^\ell$ and $\theta_{j,k}^\ell$ had the same sign in all systems. The parameters of the noise as well as the matrix $B$ were the same in all 12 systems and these facts were also known to participants.

The training data available to participants was generated by running the data generating process 20 times for each $F^i$ with different pairs of initial conditions $z$ and controls $u$. The distributions used to select $z$ and $u$ in the training data were unknown to participants and differed among systems.

**Evaluation** For each of the systems participants were provided with 50 additional sets of initial vectors, $X(0)$, as well as an indicator specifying the corresponding system ($i = 1, \ldots, 12$). For each of these combinations participants were asked to select a control input to minimize the loss function. The loss function measures the proximity of $Y^{i,k}$ to the desired value $y_*^{i,k}$ toward the end of the observation interval while also adding a penalty term depending on the size of the control input used. The exact loss function can be found in Appendix A.

**Three perspectives** *Causality* The system is causal in the sense that it specifies not only an observational distribution ($U \equiv 0$ in Equation (3)) but also a set of interventional distributions ($BU \not\equiv 0$ in Equation (3)). The intrinsic dynamics are the same regardless of which (if any) intervention is applied as they are described by the function $F$. That is, the mechanism described by $F$ is stable under interventions. The task can be thought of as a causal learning task where participants should predict the effect of interventions and choose an optimal intervention.

*Control theory* The task seeks a functional map from measurements $y(t)$ to control inputs $u(t)$ which is the classical task of *control design*. The control inputs are to be of the form (6), which can be understood as *impulse control*. Formalizing the control objective as an optimization problem is commonly known as *optimal control* (see, for example, Bertsekas (2000); Anderson and Moore (2007)). The objective function is a weighted sum that, as typical in control applications, balances *control performance* and *control effort*.

*Reinforcement learning* In this task, the vector $u$ is selected at the start of each trial and then executed for a number of steps. This problem formulation without state transitions is closely connected to bandit or contextual bandit problems (Sutton and Barto, 1998) where the agent receives a reward based solely on the selected action and context. Classical online RL approaches learn the policy by iteratively interacting with the environment and improving the policy. However, for the proposed task, the agent will need to use *offline* reinforcement learning as no interaction with the system is possible (see, for example, Lange et al., 2012; Levine et al., 2020).

## 4. Track ROBO: Controlling a robotic arm in a dynamical environment

Track ROBO is motivated by the long-term goal of learning skills that can be performed by a diverse set of robots to complete real-world tasks. For instance, one may want to teach robots new skills such as stirring a pot or cutting vegetables for cooking. The new skills can be learned more efficiently by leveraging prior experience in related tasks such as whisking eggs. Robots may also have different kinematic structures, requiring individualized control policies to accurately execute the end-effector trajectory required by a new skill. Even robots of the same type can differ because of minor variations in the production process. If we can leverage a robot's prior movement data to derive an individualized controller for a

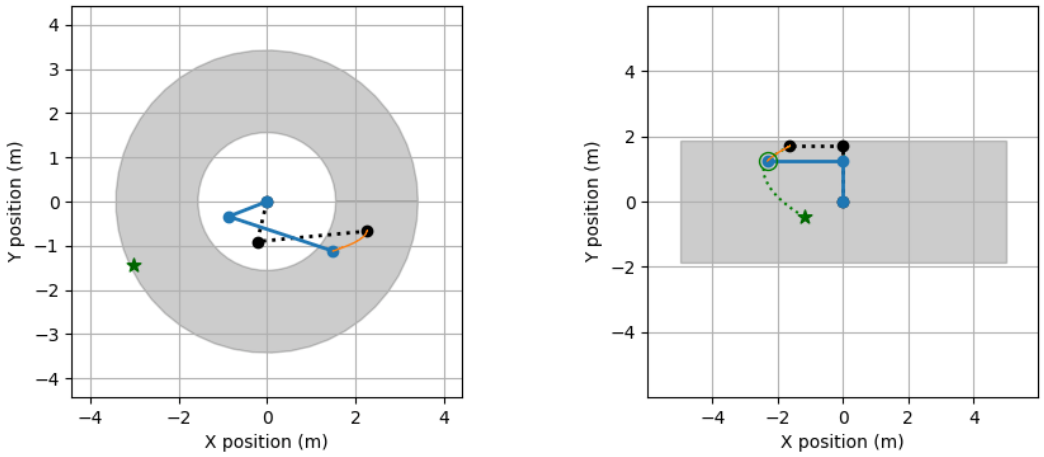


Figure 2: (Left) Rotational 2-link robot, the trail of previous positions of the robot tip (orange line), and the target position (green star). (Right) Prismatic 2-link robot and a target trajectory (green dotted line) to a target position (green star). The gray area corresponds to the reachable workspace of the robots. The black dotted lines indicate the initial position of the robots.

new skill, we may avoid the need for additional training and enable rapid roll-out of new skills.

We mimic this challenge in Track ROBO: participants are provided with movement data and asked to provide a controller that sequentially interacts with a robotic arm such that its end-effector reaches a target position provided for the next time step. The two difficulties are: (1) Participants can only set abstract control variables instead of, for example, setting the torques of individual joints directly. This restriction imitates a setting in which the robot dynamics are complicated to write down explicitly. (2) The training data is comprised of different types of trajectories than those in the test data, imitating a setting in which the robot must adjust to a new task given previous data of an old task.

We consider three robot arms: (1) A two-joint rotational robot arm, (2) a three-joint rotational robot arm, and (3) a two-joint prismatic robot arm. The rotational joints produce a rotary motion around the joint, and the prismatic joints produce a linear motion between links (see Figure 2). Each joint can be controlled by applying a voltage signal to a DC motor located in the joint. (In this challenge, the voltage is not set directly, see below.) In rotational joints, this creates a torque, while in prismatic joints this creates a linear acceleration. The resulting movement of the joints of the robot arm (and its tip in particular) are governed by the physical laws of motion. Given a specific robot one can derive exact differential equations that describe the robot's movement, known as the *dynamics model* (or *dynamics* for short) of the robot with parameters that depend on various specifications of the robot such as link mass, rotational moment of inertia, link length, location of center of mass, and friction coefficients, see Appendix B.

**Data generating processes** Participants were able to sequentially interact with 24 different robotic arm systems, with dynamics given by

$$\begin{aligned}(Z(0), W(0)) &= (z, w)\\ (\dot{Z}(t), \dot{W}(t)) &= F^s(Z(t), W(t), C(t)) = F^s(Z(t), W(t), A^s \cdot U(t)),\end{aligned} \tag{7}$$

for $s \in \{1, ..., 24\}$ where $Z(t) = ((X(t), Y(t)) \in \mathbb{R}^2$ is the position of the tip of the robot, the positions of other joints of the robot are $W(t) = (X_1(t), Y_1(t), ..., X_d(t), Y_d(t)) \in \mathbb{R}^{2d}$, $z \in \mathbb{R}^2$ and $w \in \mathbb{R}^{2d}$ are the initial values, and $C(t) \in \mathbb{R}^q$ are the underlying robot controls, that is, voltage signals applied to DC motors located in each joint ($d$ and $q$ depend on the underlying robot). The (to participants unknown) functions $F^s : \mathbb{R}^{2(d+1)+q} \to \mathbb{R}^{2(d+1)}$ are given by the second-order dynamic system of the underlying robots: $F^1$, ..., $F^8$ and $F^9$, ..., $F^{16}$ correspond to 2-link ($d = 1$) and 3-link ($d = 2$) open chain planar manipulators with revolute joints (cf. B.1), respectively, and $F^{17}$, ..., $F^{24}$ correspond to 2-link ($d = 1$) prismatic manipulators (cf. B.2); the dynamics differ further between robots due to different robot specifications denoted by $\theta^s$ (link masses and lengths, moments of inertia, friction coefficients, and locations of link center of masses). The (to participants unknown) *interface function* $G^s : \mathbb{R}^p \to \mathbb{R}^q, x \mapsto A^s x$ for $A^s \in \mathbb{R}^{q \times p}$ relates the participants' abstract control inputs $U(t) \in \mathbb{R}^p$ to the underlying robots' control inputs $C(t)$ via $C(t) = G^s(U(t))$; $A^s$ is either the identity, a square ($p = q$), or a rectangular ($p > q$) real matrix with imbalanced row-norms and full rank. Participants can control the systems only on a linearly spaced discrete time grid $(t_0, t_1, \ldots, t_{200})$ with $t_0 = 0$ and $t_{200} = 2$, that is, for each time step $\ell \in \{0, \ldots, 199\}$ it holds that $C(t) = G^s(U(t)) \equiv \text{const}$ for all $t \in [t_\ell, t_{\ell+1})$. Some of the 24 systems share robot dynamics ($F^s$), specifications ($\theta^s$), and/or the control interfaces ($A^s$) in a systematic way, which is reflected in the naming convention but was not explicitly revealed to the participants (cf. Table 1 for an overview).

**Task** The competition task is to control the robots' end-effector position $Z(t)$ to follow a given target process $t \mapsto z_*(t)$. More specifically, participants needed to implement a controller, that is, a function $\texttt{controller}^s : \mathbb{R}^{2(d+1)+2} \to \mathbb{R}^p$ for each robot ($s \in \{1, ..., 24\}$). At each time step $\ell \in \{0, \ldots, 199\}$, the controller is queried for the next control input $U(t_\ell) \in \mathbb{R}^p$ given the current positions $Z(t_\ell), W(t_\ell) \in \mathbb{R}^{d+1}$, their derivatives $\dot{Z}(t_\ell), \dot{W}(t_\ell) \in \mathbb{R}^{d+1}$, and a target end-effector position $z_*(t_{\ell+1}) \in \mathbb{R}^2$ for the next time step. The task does not involve planning as the controller only gets access to the next time step's end-effector target position, however, the implemented controller can gather information during the control process[2]. If the controller does not return within given compute time and resource constraints, we set $C(t_\ell) = G^s(U(t_\ell)) = \mathbf{0}$. This way, the different robots are propagated forward for different target trajectories $(z_*(t_0), ..., z_*(t_{200}))$ following their respective dynamics under the participant-provided controller; the participants' task is to align the resulting end-effector trajectory $(Z(t_0), ..., Z(t_{200}))$ with the target trajectory.

For deriving and implementing their controllers, participants were provided with (offline) training data for each system $(F^s, \theta^s, A^s)$ in the form of 50 realized end-effector trajectories and corresponding control input sequences. Training trajectories are obtained using an

2. Even though $\texttt{controller}^s$ is specified as a function of the current state and the next target state of the system, participants were able to log previous queries allowing them to use the entire past $Z(t_0), W(t_0), ..., Z(t_\ell), W(t_\ell)$, and $\dot{Z}(t_0), \dot{W}(t_0), ..., \dot{Z}(t_\ell), \dot{W}(t_\ell)$.

LQR-controller (Anderson and Moore, 2007) (based on inverse dynamics and the (pseudo-)inverse of $G^s$ to map robot controls to participant controls) to transition from some random starting positions to some random target positions in the robot's workspace. The one-step ahead end-effector target positions used to generate the training trajectories were not provided to participants. For each of these repetitions, participants were provided with the observed processes $W$ and $Z$, their derivatives $\dot{W}$ and $\dot{Z}$, a time indicator $t$, the applied controls $U$ and an indicator $i$ specifying the system.

**Evaluation** For each of the 24 systems $(F^1, \theta^1, A^1), \ldots, (F^{24}, \theta^{24}, A^{24})$ the participants' controller implementation is used to follow 10 different target processes. More specifically, for each system $i \in \{1, \ldots, 24\}$ and repetition $k \in \{1, \ldots, 10\}$, there is a target process $z_*^{i,k} : [0, 2] \to \mathbb{R}^2$ and the robot is propagated forward using the participant-provided controller. The loss function measures how far the realized end-effector trajectory is from the target process and penalizes the size of the participants' control inputs (cf. Appendix B for details). Mimicking a real-time robot control scenario, the participants' computational resources spent on evaluating the code that implemented their controller were restricted. If the time constraints were not met, the submission was invalid.

**Three perspectives** *Causality* We can formulate the task of controlling a robot as a causal task. First, we need to estimate a model that allows us to evaluate the effect of various interventions, where interventions now correspond to setting the inputs $U(t)$. Second, we optimize a sequential intervention scheme.

*Control* The task consists of both system identification and controller design and it relies on all the steps in a typical control engineering application as outlined in Section 2.

*Reinforcement learning* Track ROBO has a standard MDP formulation wherein the action is defined by the command sent to the abstract controller and the state space is given by the joint positions and velocities. The time-varying reward is given by the robot's accuracy in following the desired trajectory with a penalty on the size of the control inputs.

## 5. Results and lessons learned

The results of the competition can be found on our website[3] and in the appendices. The website also contains videos, in which some of the competing teams describe their solutions in more detail. Code is available, too[4].

For us, one of the key questions was whether, for the model classes considered in this challenge, it is better to aim to build a model and then infer the optimal control, or to directly estimate the effect of the applied control. The competition results suggest the former in that in both tracks the winning solution was indeed inferring a data-generating model first (see in A.1 and B.3 in the appendices); it outperformed approaches of the

3. See learningbydoingcompetition.github.io.
4. See github.com/LearningByDoingCompetition/learningbydoing-comp for open-source code to reproduce the winning solutions of the competition and to try out new methods on the competition tasks; see Bravo (2022) for code that implements the winning solutions to Track CHEM and Track ROBO; see github.com/Quarticai/learning_by_doing_solution for code that implements the second winning solutions to Track CHEM and Track ROBO; see Bussmann (2022) for code that implements the third winning solution to Track CHEM; see Patiño et al. (2022) for code that implements the third winning solution to Track ROBO.

latter type by a significant margin. We speculate that imposing a model structure (even if both the structure and the parameters still need to be inferred from data) acts as strong regularization helping to ensure successful control which is robust to environmental changes. Clearly, further research is needed to better understand in which settings this is expected to be the case. In the future, it would be interesting to consider situations where the model inference becomes even harder, for example, because more parts of the system are unobserved.

## Acknowledgments

We thank the NeurIPS 2021 competition track organizers, Barbara Caputo, Marco Ciccone, and Douwe Kiela. We also thank all the participants who took part in the competition. SW and JP were supported by the Carlsberg Foundation. SWM was supported by a DFF-International Postdoctoral Grant (0164-00023B) from Independent Research Fund Denmark. TL and OK were supported by the Office of Naval Research under Grant No. N00014-18-1-2775. IG was supported by ANR Chair of Artificial Intelligence HUMANIA ANR-19-CHIA-0022. JP was supported by a research grant (18968) from VILLUM FONDEN. NP was supported by a research grant (0069071) from Novo Nordisk Fonden. The competition has been supported by the Department of Mathematical Sciences at the University of Copenhagen.

## References

J. Aldrich. Autonomy. *Oxford Economic Papers*, 41:15–34, 1989.

F. Allgöwer and A. Zheng. *Nonlinear model predictive control*. Birkhäuser, Basel, Switzerland, 2012.

B. D. O. Anderson and J. B. Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, 2nd edition, 2008.

Karl J Åström and Björn Wittenmark. *Adaptive Control*. Courier Corporation, 2013.

D. Bell, J. Kay, and J. Malley. A non-parametric approach to non-linear causality testing. *Economics Letters*, 51(1):7–18, 1996.

D. P. Bertsekas. *Dynamic programming and optimal control: Volume 1*. Athena Scientific, Belmont, MA, 2000.

F. Borrelli, A. Bemporad, and M. Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, New York, NY, 2017.

J. Bravo. Learning By Doing NeurIPS 2021 Competition Solution Code, January 2022. URL https://doi.org/10.5281/zenodo.5895099.

B. Bussmann. A Neural Network Approach to Controlling Chemical Reactions, February 2022. URL https://doi.org/10.5281/zenodo.6006496.

T. Haavelmo. The probability approach in econometrics. *Econometrica*, 12:S1–S115 (supplement), 1944.

G. W. Imbens and D. B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, New York, NY, 2015.

X. Jian and L. Zushu. Dynamic model and motion control analysis of three-link gymnastic robot on horizontal bar. In *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003*, volume 1, pages 83–87. IEEE, 2003.

S. Lange, T. Gabel, and M. Riedmiller. Batch reinforcement learning. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 45–73, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv e-prints (2005.01643)*, 2020.

L. Ljung. *System Identification: Theory for the User*. Pearson Education, 1998.

A. J. Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 1909.

R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 2017.

C. M. Patiño, E. Lopez, and J. Rodriguez. factoredai/learn-by-doing-neurips-2021: v1.0.0, January 2022. URL https://doi.org/10.5281/zenodo.5888574.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, 2nd edition, 2009.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA, 2017.

J. Peters, S. Bauer, and N. Pfister. Causal models for dynamical systems. In *Probabilistic and Causal Inference: The Works of Judea Pearl (to appear); ArXiv e-prints (2001.06208)*. ACM, 2022.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, 2nd edition, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

P. Waage and C. M. Guldberg. Studier over affiniteten (in Danish). *Forhandlinger i Videnskabs-selskabet i Christiania*, pages 35–45, 1864.

D. J. Wilkinson. *Stochastic modelling for systems biology*. Chapman and Hall/CRC mathematical and computational biology series. Chapman & Hall/CRC, New York, NY, 2006.