1 Research Description

1.1 Introduction

Imagine an autonomous cyber-physical system (CPS) deployed into a remote area. This CPS has a robot arm used to collect rock samples. The robot arm is positioned via a sequence of unit step inputs. A proportional-integration-derivative (PID) controller is used to produce smooth robot arm movements. The PID controller gains K_p, K_I and K_d were tuned prior to deployment. The CPS communicates with a rear area base station over a radio link.

Now suppose a fault has occurred in the robotic arm. The exact nature of this fault is unknown, but the PID controller gains no longer produce acceptable arm movement. Fault recovery must be initiated to restore, as much as possible, the previous behavior. No redundant hardware is available so the only viable fault recovery method is to readjust the PID controller gains. Unfortunately, the CPS does not have the computational resources to compute new gain values. It does, however, have the capability of setting the gains to any value received from the base station over the radio link.

A duplicate CPS is located at the base station. Under ideal circumstances the fault could be duplicated at the base station and the new controller gains could be determined using say Zeigler-Nichols rules. Unfortunately the exact nature of the fault is unknown so this approach won't work.

The base station can send the CPS candidate gain values and then direct the CPS to apply a unit step input and report response times, steady-state errors, and so forth. In other words, the base station works with the CPS to get observation data needed for the Bayesian optimization.

Once Bayesian optimization has determined the updated PID controller gain values, the CPS can reconfigure the PID controller and recommence operation.

1.2 CPS Research Focus

Autonomous CPSs operating in remote environments must be fault-tolerant. Fault tolerance involves two tasks: fault detection and isolation (FDI) and fault recovery (FR). FDI is not addressed in this project.

Several fault recovery methods exist. Replacing a fault system with redundant hardware is one method but in many cases is impractical because there is no room for the extra hardware. In these cases reconfiguration of the faulty hardware may be the only viable option. That said, it is difficult to know exactly how to reconfigure a faulty system if the type of fault is not known. In this research effort Bayesian optimization is used to reconfigure a PID controller. To the best of our knowledge Bayesian approaches have been used for FDI (e.g. see [1]), but not FR, which is the focus of this research effort.

1.3 Background

1.3.1 Bayesian Optimization

This section gives a brief overview of Bayesian optimization. Further details can be found in the book by Rasmussen and Williams [2] or in the review article by Shahriari et. al [3].

The problem of interest is to find the global minimum (or maximum) of some objective function f

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$$

where $\mathcal{X}$ is the problem's design space. In general $\mathcal{X} \subset \mathbb{R}^d$. This is known as a black box optimization problem whenever the analytic form of the function f is unknown. It may be possible to estimate the gradient of f numerically or a grid search could be used to find the minimum. But such approaches are impractical whenever sampling f is costly. For example, suppose $\mathbf{x}$ is a set of hyperparameters for a deep neural network and $f(\mathbf{x})$ is a regression output. Each $\mathbf{x}$ must undergo testing, which would be costly if it takes two hours to complete.

Every sample $\mathbf{x}$ from $\mathcal{X}$ produces a real number evaluation $y = f(\mathbf{x})$. That is, $f : \mathbb{R}^d \to \mathbb{R}$. These evaluations (sometimes called observations) are often corrupted by noise so that the observed value is

$$y = f(\mathbf{x}) + \epsilon \quad ; \ \epsilon \sim \mathcal{N}(0, \sigma)$$

and $\mathbb{E}\left[y \mid f(\mathbf{x})\right] = f(\mathbf{x})$. A global minimizer should ideally need only a few samples (especially whenever drawing samples from f is costly.)

Bayesian optimization is a sequential process that can solve these kind of black box optimization problems. It creates a probabilistic model of f which is much cheaper to evaluate. Observations from previous experiments serves as training data for the model.

The probabilistic model used is a *Gaussian process* (GP). The GP is a probability distribution over functions where any finite collection of function values is a multi-variate Gaussian. A GP is completely described by its mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, also called a *kernel*.

$$f(\mathbf{x}) \sim \mathrm{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{1}$$

More specifically, at test points $\mathbf{x}_1, \ldots, \mathbf{x}_n$

$$\begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{x}_1) \\ \vdots \\ \mu(\mathbf{x}_n) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \right) \tag{2}$$

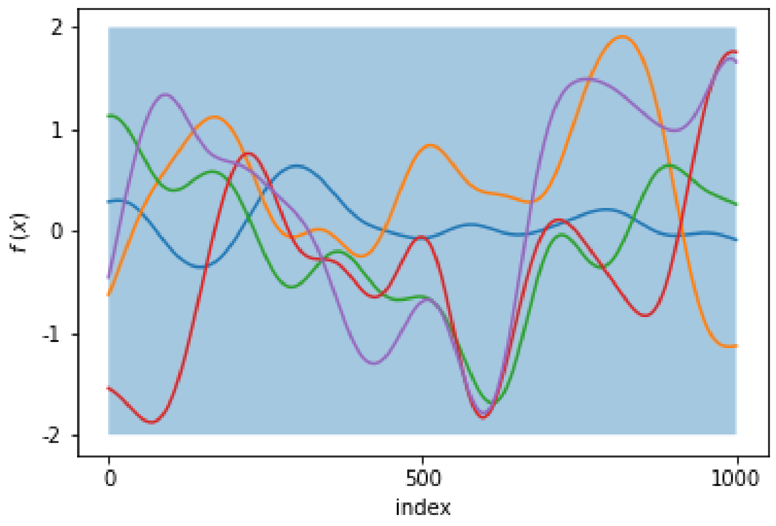
where $\mathcal{N}(\mu, C)$ denotes a multi-variate normal distribution with mean μ and covariance matrix C. The squared exponential kernel

$$k\left(x, x'\right) = \exp\left(-\frac{(x - x')^2}{2l^2}\right) \tag{3}$$

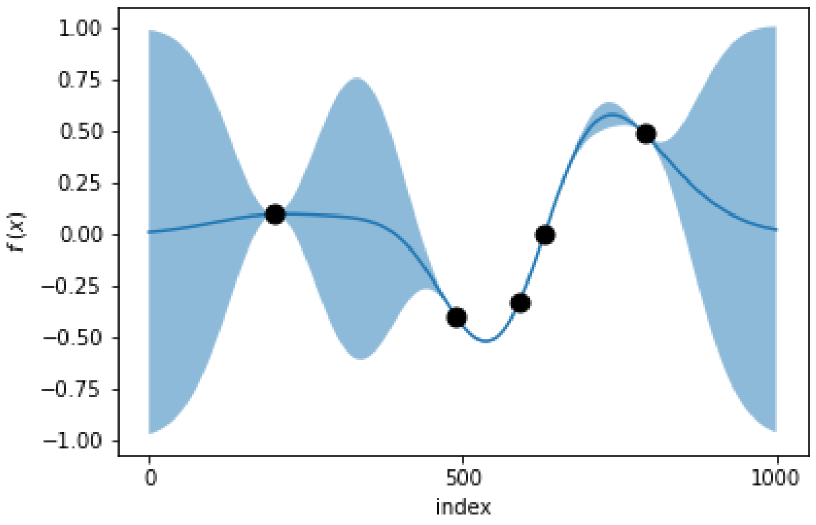
is popular. Figure 1(a) shows five sample functions with 1000 test points from a GP with a squared exponential kernel. Other kernel functions are available. The squared exponential kernel function tends to produce overly smooth functions which may be unsuitable for some problems. In such cases a more appropriate choice might be the Matérn 5/2 kernel.

It is not obvious any of the functions in Figure 1(a) could accurately model some unknown objective function. Fortunately, GPs have a nice property: they are closed under Bayesian conditions. Let $\mathcal{D} = [(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)]$ be a data set where $\mathbf{x}_i$ is the i-th training point and $y_i = f(\mathbf{x}_i)$ the observed value. Closed under Bayesian conditions means if $f \sim \mathrm{GP}$, then $f \mid \mathcal{D} \sim \mathrm{GP}$. However, this does require incorporating $\mathcal{D}$ into both the mean and covariance functions.

Let X and X_* denote the set of training points and testing points, respectively, with Y and Y_* the corresponding observation sets. The joint distribution of training observations and test point outputs is



(a) Samples from the prior distribution.



(b) Samples from the posterior distribution.

Figure 1: Shaded region depicts our belief of where the unknown objective function may lie. It reflects our level of uncertainty. Black dots in (b) represent observations. The solid line is the mean function.

$$\begin{bmatrix} Y \\ Y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}\ ,\ \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{4}$$

$K(X, X_*)$ represents a covariance matrix between the training sample points and the test sample points. The other covariance submatrices are defined similarly. Note that all functions are defined with $\mu(\cdot) = 0$ for convenience.

Equation (2) represents a prior distribution because it does not contain any information about $\mathcal{D}$. Each function in Figure 1(a) is $f \sim \mathrm{GP}(0, k(x, x'))$. The posterior distribution should filter out all functions that are not consistent with observed data. In other words, any function that does not associate all of the $(\mathbf{x}_i, y_i)$ pairs in $\mathcal{D}$ should be rejected. We can get a correct posterior distribution by redefining the mean and covariance functions. That is,

$$f(x) \mid \mathcal{D} \sim \mathrm{GP}\left(\bar{m}(\mathbf{x}), \bar{k}(\mathbf{x})\right) \tag{5}$$

with

$$\begin{aligned} \bar{m}(\mathbf{x}) &= K(X_*, X)K(X,X)^{-1}Y \\ \bar{k}(\mathbf{x}) &= k(x,x') - K(X_*, X)K(X,X)^{-1}K(X,X_*) \end{aligned} \tag{6}$$

The mean function $\bar{m}(\mathbf{x})$ is used to make predictions whereas the variance function $\bar{k}(\mathbf{x})$ measures the confidence of the prediction. Figure 1 shows both the prior and the posterior distribution of functions. The blue regions indicate our belief or uncertainty about where the unknown objective function might exist. In the prior distribution the blue area is large because there is no information available. Conversely, the posterior distribution has much smaller uncertainty because some data is available.

The more data acquired, the more accurate the probability proxy model. However, taking observations of f is costly so the number of observations should be limited. This restriction causes an inevitable conflict between exploration and exploitation in the search. This tradeoff is handled by an *acquisition function* $\alpha(\mathbf{x})$ which helps guide the search. Several different acquisition functions have been proposed such as probability of improvement [4], expected improvement [5] and upper confidence bound [6]. For example, $\alpha_{\mathrm{UCB}}(\mathbf{x}) = m(\mathbf{x}) + \kappa\sigma(\mathbf{x})$ is the upper confidence bound acquisition function where $\kappa > 0$ is user-selected to balance exploitation and exploration. But regardless of which one is selected, the basic idea is the same: at each iteration of the Bayesian optimization algorithm a global optimizer finds $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x})$. $\hat{\mathbf{x}}$ is then the next location of an observation. This process is shown in Figure 2.

A global optimizer is needed to find the maximum of $\alpha(\cdot)$. This gives rise to a question: if a global optimizer is available, why not just use it find

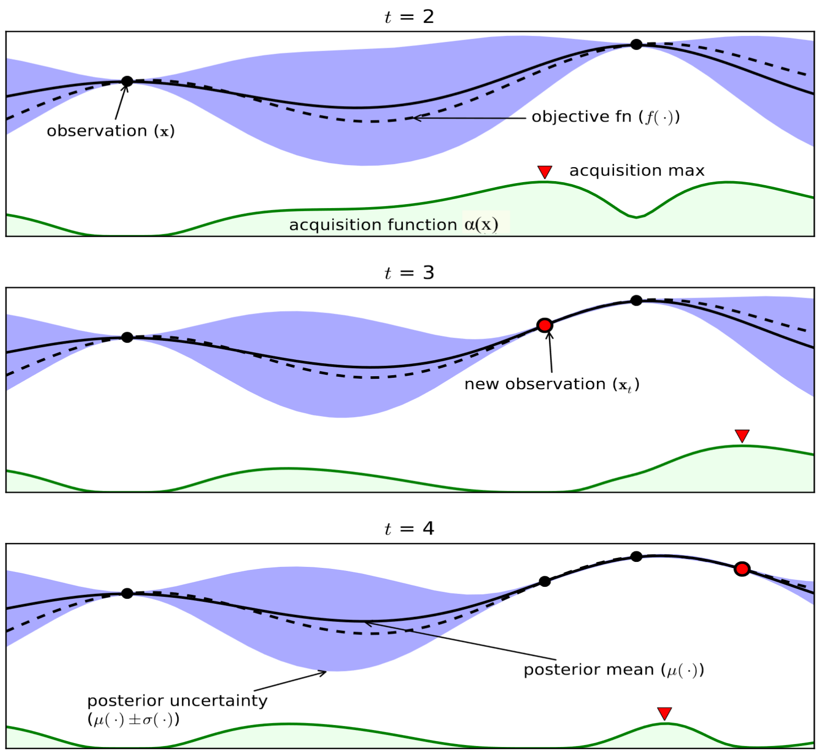


Figure 2: Using an acquisition function to choose observation points of an unknown objective function f. The dashed line is the objective function whereas the solid line is the mean function $\bar{m}(\mathbf{x})$. This figure appeared in [3].

the minimum of $f(\cdot)$? The answer is each observation of f is costly and it will take many samples to find the minimum. On the other hand, it is relatively cheap to find the maximum of $\alpha(\cdot)$.

In summary, Bayesian optimization creates a probabilistic model of an unknown objective function f. It does this by fitting a GP to observed data. The mean function $\bar{m}(\mathbf{x})$ from this GP is a proxy function that approximates f. More samples produces a more accurate model but sampling f is costly. It is therefore wise to choose sample points believed to be near a global optima of f. An acquisition function, which trades off exploration and exploitation, figures out where to take the next sample. Optimizing a proxy function is more efficient than optimizing an objective function.

Algorithm 1 shows the steps in a Bayesian optimization. $\mathbf{x_t}$ is a sample point at time t and $\mathbf{y_t}$ is the corresponding observed value of $f(\cdot)$. Each iteration the acquisition function $\alpha(\cdot)$ tells where to take the next sample. The algorithm assumes you are trying to find the minimum of an objective function $f(\mathbf{x})$. Only line 8 must be changed if the maximum of f is required.

Algorithm 1 Bayesian Optimization

```
1: INIT: Data $\mathcal{D}_0 = \{\mathbf{x}_0, y_0\}$
2: for $t = 1, 2, \ldots$ do
3:     Update GP using $\mathcal{D}_{t-1}$
4:     Find $\mathbf{x}_t = \arg\max_{\mathbf{x}} \alpha(\mathbf{x})$
5:     Evaluate $y_t = f(\mathbf{x}_t)$
6:     $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$
7: end for
8: return $\mathbf{x}^* = \arg\min_{\mathbf{x}} y(\mathbf{x})$
```

1.3.2 Evolutionary Algorithms

A global optimizer is needed to find the maximum of the acquisition function $\alpha(\mathbf{x})$. This function is multimodal so gradient techniques (e.g., hill-climbing) are likely to fail. Evolutionary algorithms (EAs) are stochastic search techniques. Although they are not guaranteed to find the global optima, they are quite effective at locating near optimal points even on high dimensional, multimodal functions [7].

There are different varieties of EAs (e.g., genetic algorithms, evolution strategies, etc.) but all of them generally follow the steps shown in Figure 3. Individuals (solutions) are collected into a population which evolves over time. The initial population is randomly generated. Thereafter, each gen-

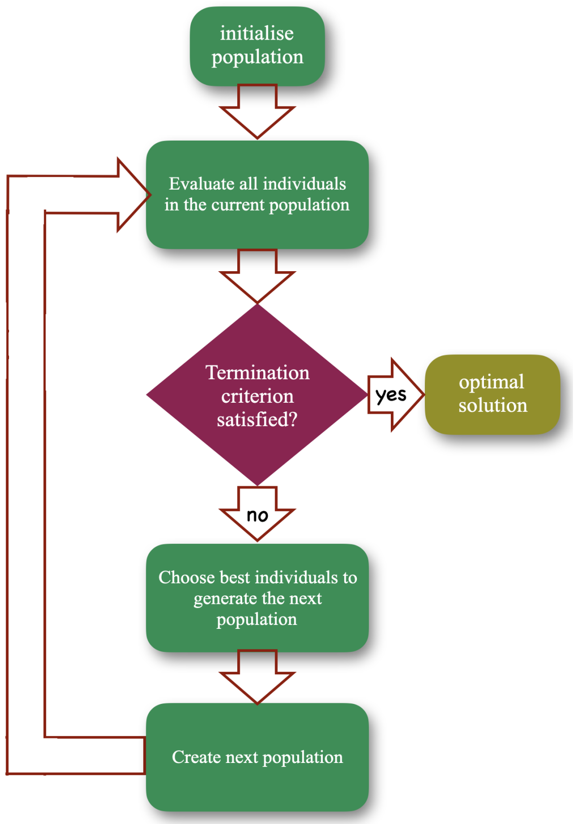

initialise population
Evaluate all individuals in the current population
Termination criterion satisfied?
yes
optimal solution
no
Choose best individuals to generate the next population
Create next population

Figure 3: A basic evolutionary algorithm.

eration (iteration) individuals in the current population are evaluated for fitness (solution quality). Highly fit parents are selected for reproduction (random variation) to generate offspring (new candidate solutions). In some EAs the offspring replace the parents whereas in other EAs parents and offspring compete for survival. Survivors become the parents in the next generation. This evolutionary process of reproduction, fitness evaluation and survival continues until a termination criteria is met, typically after a fixed number of generations have evolved. The best fit solution in the final population is the optimal solution.

We will use an evolution strategy to find the global maxima of $\alpha(\mathbf{x})$.

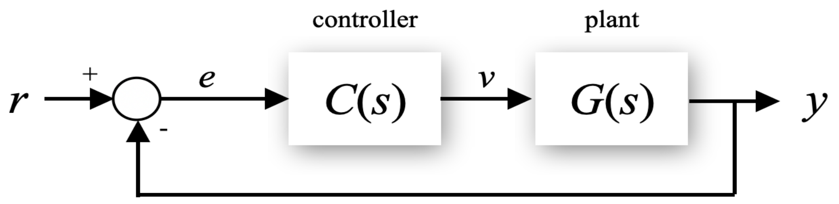


Figure 4: A closed loop system with a PID controller.

1.3.3 PID Control

The *proportional-integral-derivative* (PID) controller is simple, versatile and in wide-spread use. Referring to the block diagram in Figure 4, the controller output $v(t)$ is the input to the plant. In the time domain

$$v(t) = K_p e(t) + K_I \int e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

where $e(t) = r(t) - y(t)$ is an error signal. The controller transfer function is

$$C(s) = \frac{K_d s^2 + K_p s + K_I}{s}$$

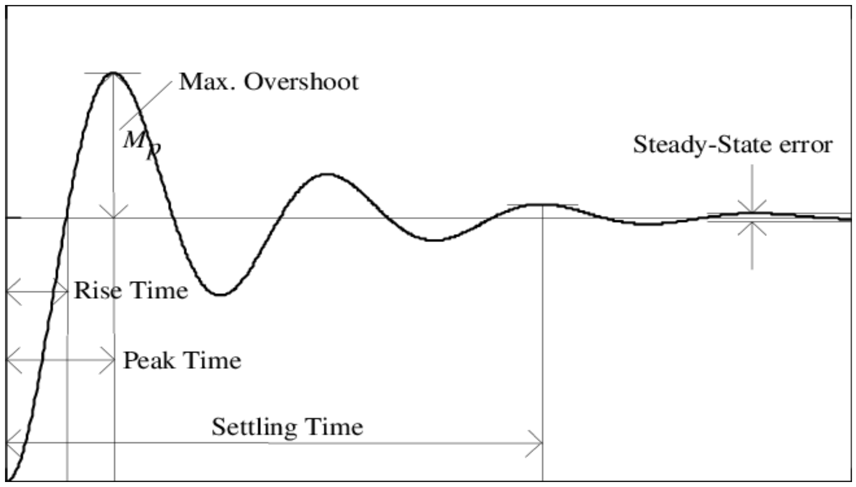


Figure 5: Unit step response

Figure 5 shows a typical linear system output response to a unit step input. The purpose of the PID controller is to make a plant's step response meet specific design criteria such as a maximum settling time or steady-state error. The plant behavior is set by adjusting the PID controller gains K_p, K_I, and K_d. The table below shows how these gains effect a plant's step response.

gain	rise time	overshoot	settling time	steady-state error
K_p	↓	↑	small change	↓
K_I	↓	↑	↑	↓
K_d	small change	↓	↓	no change

The derivative term reduces transient overshoots and is effective for plants with large dead-time. Nevertheless, it can amplify high frequency sensor noise. It is therefore not uncommon to just use a PI controller—i.e., $K_d = 0$.

It is not an easy task to determine the controller gains. Manual tuning can be done although such trial-and-error techniques can be tedious. More often tuning heuristics such as the Zeigler-Nichols rules or the Rivera et. al [8] methods are used. Some automatic tuning tools are also now available.

2 Evaluation/Experimentation Plan

The research will be split into two phases: a proof of concept phase and a validation phase.

2.1 Proof of Concept Phase

The plant of a CPS will be simulated by a 3rd-order, type 0 transfer function $G(s)$ with a PID controller. (This system is depicted in Figure 4.) Faults in this system will be simulated by adding additional (possibly complex-conjugate) poles and/or zeros to $G(s)$. It is assumed the number and location of these poles/zeros are unknown so the PID controller gains cannot be found using conventional tuning methods. Several different scenarios will be investigated.

There is always some underlying objective function that relates PID gain values to step response behavior. For example, the steady-state error θ to a step input is a function of the integration gain K_I—i.e., $\theta = f(K_I)$. This function is unknown because the exact nature of the fault is unknown. Bayesian optimization can create a probabilistic model of $f(K_i)$ and the proxy function minimized to find a new K_I gain that minimizes the steady-state error in the faulty system. The other PID controller gain values can be optimized in a similar manner.

The Bayesian optimizer software will propose new candidate K_p, K_I and K_d gain values which will be implemented in the physical system PID controller. A step input will produce a new step response. The overshoot, rise time, settling time or steady-state error can be extracted and constitute data observations of some underlying objective function. The Bayesian optimizer uses this data to create more accurate probabilistic models of the underlying objective function from which a new gain value can be obtained. This new gain value is sent to the CPS for evaluation. This iterative process repeats until acceptable behavior in the physical system is restored.

Full fault recovery is desired, but not guaranteed. Therefore, acceptable behaviors will be defined and the fault recovery method will be considered successful if acceptable behavior is achieved.

All simulation will be performed using MATLAB and SIMULINK. The GP and EA programming will be done in Python.

2.2 Validation Phase

A ball & beam system will be used to validate our proposed fault recovery method. A ball is placed on a beam as shown in Figure 6. The ball rolls to the left or right depending on the beam angle α. A servo motor rotates through an angle θ which moves a lever arm up or down thus changing α. The objective is to get the ball to stop at some desired position along the beam length L. Sensors determine the ball position on the beam.

Funding is requested to purchase a commercial quality, off-the-shelf ball & beam system. A suitable candidate system is manufactured by Acrome and is shown in Figure 7. The Acrome ball & beam system has a built-in PID controller to drive the servo motor. The PID controller gains are user defined and will be initially set for a slightly under-damped step response. This will be the fault-free system. Data will be obtained by applying a unit step input and then recording the response (settling time, steady-state error, etc.) A faulty system will be created by attaching small weights at the end of the beam where the lever arm is attached. Bayesian optimization determines the new PID controller gains and the new behavior is compared against the design specifications.

It is not expected that the new gains will completely restore the original behavior. Therefore the specifications will be slightly relaxed. Fault recovery is achieved if the system meets the relaxed specifications. For example, the original specification may require zero volt steady-state error (for a unit step input). Fault recovery may be defined as successful if the faulty system has a steady-state error of say at most ±20mV. Similar relaxed specifications will be formulated for rise time and settling time.

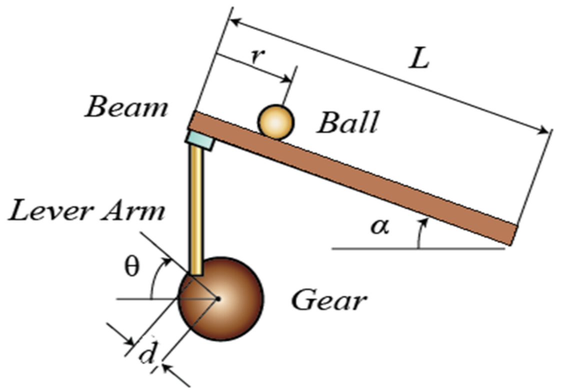


Figure 6: A ball and beam system

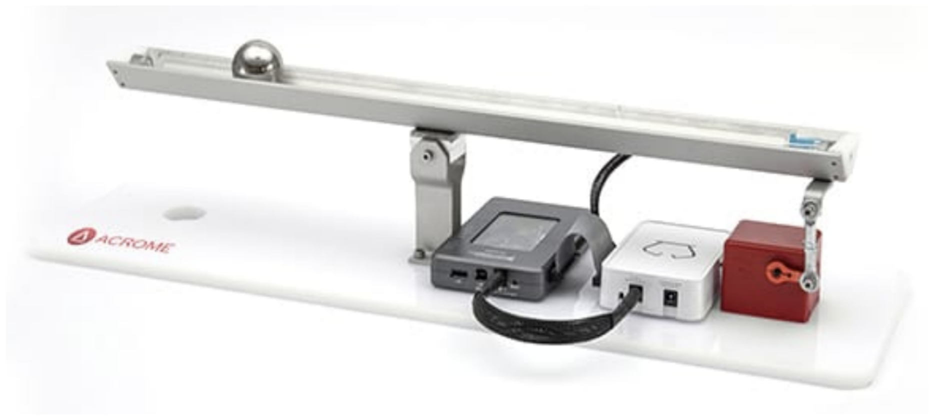


Figure 7: Acrome Ball & Beam [9]

3 Project Management and Collaboration Plan

The PI intends to employ one upper-division, undergraduate student in this research project. (No graduate research assistant is required.) The undergraduate research assistant has the following tasks:

- Construct models using MATLAB+SIMULINK. Both fault-free and faulty system models will be required.
- Determine PID controller gains for the fault-free system
- Collect data from the models as needed for the Bayesian Optimization
- Become familiar with the ball & beam system hardware and software.
- Conduct testing during the validation phase
- Assist in writing a peer-reviewed paper describing this research project

4 Broader Impacts

A full description of this research project plus all MATLAB and Python source code will be uploaded to GitHub, making it available to the general public.

The results of this research will be documented in a peer-reviewed conference paper. The IEEE Computational Intelligence Society annually runs the Symposium Series in Computational Intelligence. Either the control and automation symposium or the evolvable systems symposium would be suitable. Funding is requested for the PI and the student to attend the conference. Although not mandatory, the student will be encouraged (and coached) to orally present the paper at the conference.

The PI believes in the full participation of women, persons with disabilities and underrepresented minorities in STEM research. Selection priority for the undergraduate research assistant will go to an upper-division electrical or computer engineering student from a historically underrepresented group. This student must have appropriate programming language skills (MATLAB/simulink) and a strong control systems background. All engineering students at Portland State University learn MATLAB and use it in several courses. The control systems background requirement is met by completing ECE 317 (*Signals & Systems III*) or equivalent with a grade of **B** or better.

The PI annually teaches a graduate level course ECE 584 *Foundations of Cyber-physical Systems.* One topic in that class is the design of fault-tolerant embedded systems. The results of this research project will be incorporated into those lectures.