# Design, Analysis, and Learning Control of a Fully Actuated Micro Wind Turbine

J. Zico Kolter, Zachary Jackowski, Russ Tedrake*

***Abstract*— Wind power represents one of the most promising sources of renewable energy, and improvements to wind turbine design and control can have a significant impact on energy sustainability. In this paper we make two primary contributions: first, we develop and present a actuated micro wind turbine intended for research purposes. While most academic work on wind turbine control has largely focused on simulated evaluations, most turbine simulators are quite limited in their ability to model unsteady aerodynamic effects induced by the turbine; thus, there is a huge value to validating wind turbine control methods on a physical system, and the platform we present here makes this possible at a very low cost. The second contribution of this paper a novel policy search method, applied to optimizing power production in Region II wind speeds. Our method is similar in spirit to Reinforcement Learning approaches such as the REINFORCE algorithm, but explicitly models second order terms of the cost function and makes efficient use of past execution data. We evaluate this method on the physical turbine and show it it is able to quickly and repeatably achieve near-optimal power production within about a minute of execution time without an a priori dynamics model.**

## I. INTRODUCTION

Energy issues pose one of the greatest challenges facing society. More than 86% of the world's energy currently comes from (unsustainable) fossil fuels, and worldwide energy demand continues to grow rapidly [1]. Wind power represents one of the most promising sources of renewable energy: currently wind is more economically feasible than solar or biomass for electricity generation, with some projections predicting wind, given good environmental conditions and proper government subsidies, to be of similar cost to fossil fuels for electricity generation [2]. Despite this promise, significant improvements in the deployment and control of wind turbines are needed if wind is to contribute a significant portion of electricity worldwide [3]. One particularly important challenge, which we consider in this paper, is extracting maximum power in "Region II" wind speeds, where the turbine is generating power but has not yet reached it's rated power output [4]; since efficient operations in such speeds can potentially allow for wind turbines in many more locations than currently possible, addressing this problem is crucial for continued expansion of wind power [5].

In this paper, we make two primary contributions. First, we develop and present a fully actuated variable pitch, variable speed micro wind turbine (shown in Figure 1) with power sensing, load control, and individual blade pitch control, along with a software architecture for controlling the system. While simulators naturally play a large role in wind research due to the impracticality and cost of many control experiments on full-scale turbines, there are also considerable limitations to wind turbine simulators: a turbine under non-uniform incoming flow or employing periodic pitch control can induce unsteady aerodynamic flows [6], whereas most common simulators are based on quasi-steady aerodynamic assumptions. Thus, there is a huge value to wind experiments conducted on physical systems, since these can validate control methods beyond what is possible with simulation alone; indeed, as we illustrate in this paper, even steady-state power predictions from a standard simulation tool are quite far from what is actually produced by our turbine. While micro turbines are of course much smaller scale that utility-sized, empirical results on such platforms are a natural complement to the simulation-based results that currently dominate most academic turbine control work.

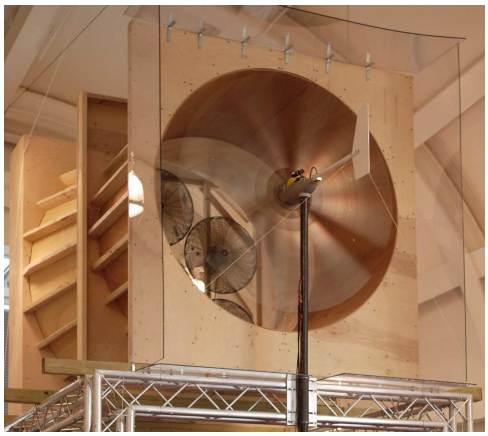

Fig. 1. Picture of our micro wind turbine in operation.

The second contribution of this paper is a trust-region method for online policy optimization, applied on the turbine to maximize power output in Region II winds speeds. Our algorithmic method builds upon reinforcement learning algorithms for stochastic policy gradient methods [7] (also called likelihood ratio stochastic gradient methods [8]), but explicitly includes second-order terms that approximate the Hessian of the cost function in order to take larger steps the policy parameter space. By using past data via importance sampling methods, the algorithm is able to efficiently build local second order estimates of the cost function, and then find the parameters that optimize this cost function. We compare the method to existing reinforcement learning techniques, and show that the algorithm is able to quickly

*All authors are with the Computer Science and Artificial Intelligence Laboratory, Massaschusetts Institute of Technology, Cambridge MA, 02139 {kolter,zackbass,russt}@csail.mit.edu.

and reliably steer the turbine operating parameters to those that maximize its power production.

## II. BACKGROUND AND RELATED WORK

Wind energy has received a great deal of attention in recent years, and there is a vast amount of work on the design, modeling, and control of wind turbines. A full review of relevant literature is beyond the scope of this paper, and we focus instead only on those elements of turbine design and control that are most relevant to the presentation here. However, there are several introductory texts on wind energy in general, and we refer the reader to these for more information [4], [9]. A number of papers survey more recent control work and challenges [10], [11].

The piece of wind turbine control work most closely related to this paper is work on power optimization using collective pitch control. Most initial work in this area was largely model-based, and focused on methods for accurately tracking the optimal control setpoints under changing conditions [12], [13], [14]. Alternatively, newer work has focused on model-free methods for optimizing power coefficients, using pitch control, via extremum seeking control [15] or finite difference methods [16]. These approaches are conceptually similar to the proposed approach here, but are strictly first-order methods, do not efficiently reuse past trajectories, and were tested only in simulation.

Since this paper is about a physical platform, we also briefly discuss turbine controls work on real hardware. The vast majority of research in advanced wind control techniques (including all papers cited above ) is performed entirely in simulation: the ubiquity of standard simulation packages such as FAST [17] and WT_Perf [18], as well as the cost involved with work on a a full-scale turbine, have made full system real-world experiments a rarity in wind research. Notable exceptions to this include work on the Controls Advanced Research Turbine (CART) [19], including work on load reduction [20] and individual pitch control [21], and work on adaptive control for Region II power optimization (but without pitch control). However, with the exception of this last paper, work on the CART has focused largely on model-based approaches, largely due to the fact that significant effort has been invested in ensuring accurate models for this particular turbine; furthermore, the CART is a relatively large-scale system, and not replicable by most research labs for testing purposes. There is also some work on smaller scale turbines [22], but we know of no such systems with active pitch control.

From a controls perspective, our online learning method builds most directly upon the REINFORCE algorithm [7] and likelihood ratio gradient estimation [8]. As stochastic methods that seek to improve control performance via interaction with the system, these methods are closely related to adaptive control techniques in general, and also iterative learning control (ILC) [23] and extremum seeking control [24]. Informally speaking, a characteristic of the policy search methods we consider is their batch formulation: the methods typically consider several executions of a policy on the real system, and then use this experience to update the parameters of the policy by approximately computing gradient or other update information. Of the above methods, this is most similar to iterative learning control, but the focus of such research has been on more on improving probabilistic estimates of the gradient (e.g. [25]) and devising different optimization-based approaches to using these gradient (e.g. [26]) rather than the stability analysis that is more common in ILC. Fundamentally speaking, though, the policy search methods have similar goals, and the algorithms we discuss could certainly be viewed as ILC methods.

## III. HARDWARE AND SOFTWARE DESIGN

We begin by discussing the physical turbine platform we have developed for this work. Our platform is a variable pitch, variable speed wind turbine, with a blade radius of 1.52 meters and a maximum power output of 300W DC. The system is based upon the Extractor turbine designed by Alternate Power Technologies, Inc.[1], but with significant modifications. We replaced the mechanical blade pitch mechanism of the Extractor turbine with servo motors attached to each blade root; this allows us to individually control the turbine blades at very high frequencies. We transmit data and power signals to the servos using a slip ring in the turbine nacelle and track the rotor angle using an encoder off the main rotor shaft. Offboard, the power generated by the system is fed into a board that both monitors the power output by the system, and can programmably vary its resistance (this in turn produces more or less torque on the generator). Finally, a simple wind tunnel built using commodity fans powers the turbine itself. All the elements are controlled by an offboard computer via the Lightweight Communication and Marshaling (LCM) software [27]. A block diagram of these components is shown in Figure 2.

One element we want to highlight is the total cost of the system, which is approximately $3,250 for the turbine and associated hardware ($1,000 for the Extractor turbine, $1,550 for the Dynamixel servos, $350 for encoder and slip ring, $250 for the power monitoring hardware and $100 for various cables and wiring) plus $2,600 for the wind tunnel ($2,100 for fans, and $500 for lumber). We are not aware of any low-cost micro turbines that have the capabilities of our system (individual pitch control, variable speed operation); thus, we feel the platform itself has the potential to significantly impact small and academic-scale research in wind power. We are happy to share design specifications with anyone wishing to build a similar system.

### A. Hub and Nacelle Design

The turbine hub and nacelle house the pitch actuators, rotor encoder, and generator, as shown in Figure 3. To control blade pitch, we use three Dynamixel EX-106+ servo motors attached to the blade roots. The servos are daisy chained together and controlled digitally via an RS-485 link, which allows for setpoint control at 250 Hz with less than 1ms

[1] http://www.vpturbines.com

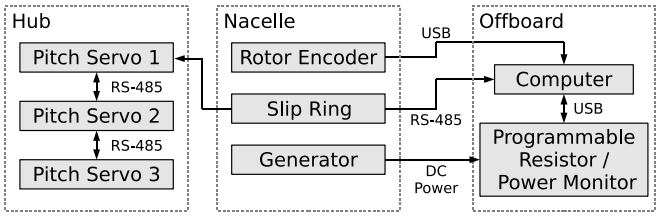


Fig. 2. Block diagram of the hardware components in the micro turbine.

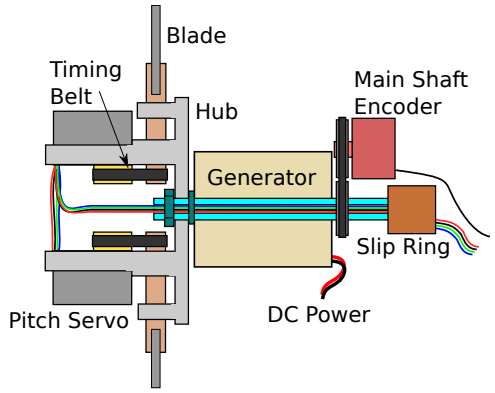


Fig. 3. Mechanical diagram of the turbine hub and nacelle.

latency (in addition to the onboard PD controller in the servos). The hub and servo are shown in Figure 4.

Since the hub itself rotates, to provide data and power to the servos we use a Mercotac 430 slip ring in the nacelle, and run the wires through the rotor shaft. The nacelle also houses a US Digital HB5M encoder off the main shift used to track the orientation of the rotor (necessary both to estimate speed and to synchronize blade orientation with rotor angle) as well as the generator itself, a Delco 12SI alternator modified for use with the Extractor turbine. The components of the nacelle are shown in Figure 5.

### B. Power Control and Monitoring

To monitor and regulate the power output by the turbine we attach a resistor to the DC power output lines, and monitor current and voltage using a Phidgets USB A/D converter and associated voltage and current measuring devices. In order to regulate the power output it is important to be able to vary the effective resistance: lower resistance will impose a larger torque on the generator, and some intermediate (but

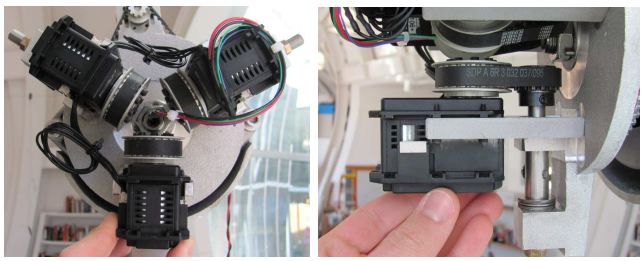

Fig. 4. Front view of the turbine hub and a closeup of the servo motor used for pitch control.

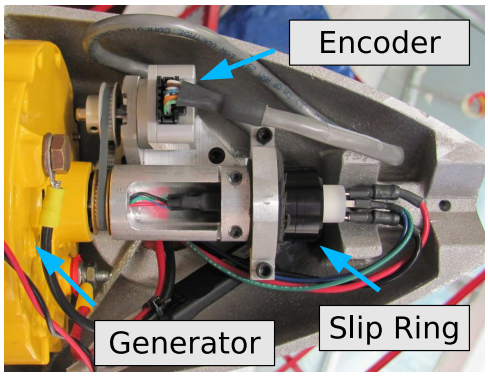


Fig. 5. Internals of the turbine nacelle.

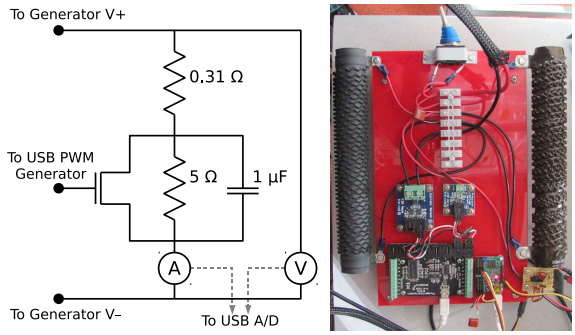


Fig. 6. Circuit diagram (left) and photograph (right) of the programmable resistor and power monitoring board.

unknown) resistance is needed to maximize power output. Thus, we place two high-power resistors (0.31 and 5 Ohm respectively) in series, and short the later with a MOSFET controlled by a PWM signal at 25kHz. By varying the duty cycle of the PWM from 0 to 1, we can smoothly interpolate between a resistance of 5.31 and 0.31 Ohms. The circuit diagram for this device, as well as a photograph of the board, is shown in Figure 6.

### C. Software Architecture

The turbine is controlled via an offboard computer, running a software system built upon the LCM message passing framework. LCM offers an attractive architecture for building such a system, as the different drivers and controllers can be developed in a modular fashion, and the system has built-in logging and playback capabilities. The basic software architecture is shown in Figure 7. In addition to the three driver modules that manage the servos, the encoder, and the power board, we use two basic controllers: 1) a collective pitch that controls all the blades equally, using a simple integral control loop (although the servos internally have a PD controller, the friction in the drive train and steady aerodynamic forces often cause these to have a steady state error, which we correct via an integral controller), and 2) an independent pitch controller that synchronizes blade pitch of the different blades with rotor angle using feed-forward compensation; this element will be discussed further in Section V-B. Also illustrated in Figure 7 is a policy search procedure that we will describe in Section V-C, which uses

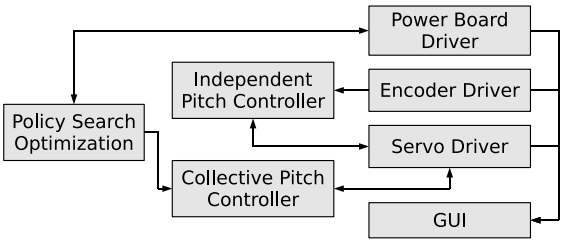


Fig. 7. Block diagram of the turbine software architecture.

Fig. 8. Rear view of the wind tunnel used for powering the turbine.

collective pitch control and feedback from the power board to optimize energy output, and a MATLAB graphical user interface that allows for easy visualization and control of the different modules.

### D. Wind Tunnel

To power the turbine, we constructed a small circular wind tunnel using wood and commodity commercial fans. The tunnel is eight feet long (a four foot contraction section and a four foot straight section), with an output diameter of 1.72 meters (20 cm larger than the diameter of the turbine), and generates an average wind speed of 6.5 m/sec over this area (approximately 13.5 miles per hour). This is not fast enough to reach the rated power output of 300W, but is still able to generate a significant amount of power at high rotational speeds. A photograph of the fans powering the tunnel is shown in Figure 8.

## IV. TRUST REGION POLICY SEARCH

The primary control task we consider in this paper is maximizing power production in wind speeds below the turbine's rated power; concretely, by adjusting the resistance of the load (which in turn affects the turbine rotational speed), and by changing the blade pitch angle, we seek to generate the largest amount of power possible from the turbine. The task is made challenging by the fact that we do not assume a known model of the plant dynamics and because we can only obtain *noisy* estimates of the power production for a given set of operating conditions. To accomplish this task in an efficient manner we develop a new method for policy search, similar to the REINFORCE algorithm [7] mentioned above.

To formalize the method, we let $w \in \mathbb{R}^n$ denote a set of *policy parameters* (e.g., the blade pitch and load resistance for the turbine), and let $p(J|w) : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}_+$ denote a probability distribution over costs for a given set of parameters. The goal is to choose parameters that minimize the expected cost

$$\mathbf{E}[J|w] = \int p(J|w) J dJ. \tag{1}$$

The difficulty of trying to optimize parameters $w$ is that we cannot directly compute the gradient of the expected cost with respect to $w$, and indeed this derivative may not exist depending on the form of $p(J|w)$.

Instead likelihood ratio methods typically consider *stochastic* policies that actually maintain a *distribution* over possible parameters and optimize with respect to this distribution. In particular, we let $\theta$ denote a set of *distribution parameters*, and let $p(w;\theta)$ denote the distribution over policy parameters $w$ given $\theta$. Then the expected cost for parameters $\theta$ is

$$\mathbf{E}[J;\theta] = \int p(w;\theta)\mathbf{E}[J|w]dw \tag{2}$$

and we can represent the gradient of this expected cost as

$$\begin{aligned}
\nabla_\theta \mathbf{E}[J;\theta] &= \int \nabla_\theta p(w;\theta)\mathbf{E}[J|w]dw \\
&= \int p(w;\theta)\frac{\nabla_\theta p(w;\theta)}{p(w;\theta)}\mathbf{E}[J|w]dw \\
&= \int p(w;\theta)\nabla_\theta \log p(w;\theta)\mathbf{E}[J|w]dw \\
&= \mathbf{E}[J\nabla_\theta \log p(w;\theta)].
\end{aligned} \tag{3}$$

The key here is that this last expression involves only easily computable quantities (since $p(w;\theta)$ is known, the $\nabla_\theta \log p(w;\theta)$ term can be computed exactly). Thus the gradient can be approximated by sampling: if we sample parameters $w^{(1)}, \ldots, w^{(m)} \sim p(w;\theta)$, execute these policies and receive costs $J_1, \ldots, J_n$, then we can approximate the gradient as

$$\tilde{\nabla}_\theta \mathbf{E}[J;\theta] = \frac{1}{m}\sum_{i=1}^{m} J_i \nabla_\theta \log p(w^{(i)};\theta) \tag{4}$$

For the Gaussian noise above, $\nabla_\theta \log p(w;\theta) = \epsilon(w-\theta)$ so that the update is equivalent to the so-called weight perturbation method [28]. Finally, since the gradient is not affected by adding an arbitrary constant to the cost, it is common to change the observed costs to be $\tilde{J}_i = J_i - b$ for $b$ chosen to minimize the variance of this estimate [25].

Despite its simplicity, the REINFORCE algorithm has significant drawbacks: it often takes several executions of the policy in order to obtain a gradient estimate, and even then one is typically limited to first-order optimization techniques such as stochastic gradient descent. Instead, we propose an alternative approach that 1) explicitly models second order function information to take larger steps in parameter space and 2) makes efficient use of past data so as to quickly obtain accurate estimates. Formally, we seek a second order

approximation of $\mathbf{E}[J|w]$ that is as accurate as possible for $w \sim p(w;\theta)$, and which captures the change in $J$ with respect to changes in the gradient of the log probability $\nabla_\theta \log p(w;\theta)$,

$$\begin{aligned}\hat{\mathbf{E}}[J|w] \approx J_0 + g^T(\nabla_\theta \log p(w;\theta)) \\ + \frac{1}{2}(\nabla_\theta \log p(w;\theta))^T H(\nabla_\theta \log p(w;\theta)).\end{aligned} \tag{5}$$

For $p(w;\theta) = \mathcal{N}(\theta, \epsilon I)$ (the case we will focus on for the remainder of the paper) this simplifies to a weighted version of the standard second-order approximation

$$\hat{\mathbf{E}}[J|w] \approx J_0 + \epsilon g^T(w-\theta) + \frac{\epsilon^2}{2}(w-\theta)^T H(w-\theta). \tag{6}$$

We can find the parameters that maximize the likelihood of this approximation by simple least-squares regression. Letting $\mathrm{svec} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n(n+1)/2}$ denote a function that maps symmetric $n \times n$ matrices into a vector containing all their upper triangular entries, we want to find $J_0$ $g$, and $H$ so as to minimize

$$\mathbf{E}\left[\left(\hat{\mathbf{E}}[J|w] - \begin{bmatrix}\mathrm{svec}(H)\\ g \\ J_0\end{bmatrix}^T \phi(w)\right)^2\right] \tag{7}$$

where the outer expectation is taken with respect to $w \sim p(w;\theta)$ and where

$$\phi(w) \equiv \begin{bmatrix}(\epsilon^2/2)\,\mathrm{svec}((w-\theta)(w-\theta)^T) \\ \epsilon(w-\theta) \\ 1\end{bmatrix}. \tag{8}$$

This can be approximated by simply sampling several $w^{(i)} \sim p(w;\theta)$ and minimizing the empirical least squares error: we denote this approximate solution as $\hat{J}_0, \hat{g}$ and $\hat{H}$ and for simplicity of notation we will fold the scaling constants $\epsilon$ and $\epsilon^2$ into $\hat{g}$ and $\hat{H}$.

While the above approach (in the Gaussian case) just is a standard method for approximating $\mathbf{E}[J|w]$ locally as a quadratic function, there are two pitfalls when applying this method to policy search. First, when optimizing parameters, the possible non-convexity of our estimated $\hat{H}$ is problematic (either due to the non-convexity of $\mathbf{E}[J|w]$ itself, or because we may not have enough samples to accurately estimate $H$). To address this issue, we take an approach common in optimization and apply a trust region solver to update parameters $\theta$ as

$$\theta_{t+1} \leftarrow \theta_t + \arg\min_{\Delta\theta^T Q \Delta\theta \le 1} \frac{1}{2}\Delta\theta^T \hat{H}\Delta\theta + \hat{g}^T\Delta\theta. \tag{9}$$

where $Q \in \mathbb{R}^{n\times n}$ defines is a positive semidefinite trust region. Although this problem is non-convex for general $\hat{H}$, it is a well-known trust region sub-problem, and can for example be solved exactly by semidefinite programming [29, Appendix B]. Further, we argue that trust region methods are particularly well-suited to policy search. Whereas trust region methods for optimization typically solve the trust region sub-problem only approximately, because policy execution is often the most time-consuming portion of the policy search process, we can solve these sub-problems exactly, using the semidefinite method mentioned above. Furthermore, while choosing the trust region $Q$ is a somewhat arbitrary decision for most trust region methods, for policy search there is a natural choice; since $Q$ represents the region where we are "confident" in our quadratic approximation, choosing $Q = \epsilon^{-1} I$ (the inverse covariance of our distribution over $w$) will update the parameters to lie within a region that is well-approximated by our least-squares approximation of $H$ and $g$.

The second difficulty with using second-order approximations for policy search is that the number of samples required to estimate second-order information can be much higher than that needed to estimate the gradient alone. To overcome this limitation, we employ importance sampling, as recently described in the context of policy search [30], in order to efficiently re-use past data to estimate $\hat{H}$ and $\hat{g}$. In particular, let $\theta_t$ be the nominal parameters executed at iteration $t$ of the algorithm, and suppose that at a previous iteration $\tau < t$, when the nominal parameters were $\theta_\tau$, we and we sampled parameters $w^{(i)} \sim p(w;\theta_\tau)$. Then even though $w^{(i)}$ was not sampled from $p(w;\theta_t)$ we can still use it to estimate the parameters at iteration $t$ using importance-sampling, by re-weighting it by the correction factor $p(w^{(i)};\theta_t)/p(w^{(i)};\theta_\tau)$. A summary of the entire algorithm, dubbed Trust Region Policy Search (TRPS), is shown in Algorithm IV.

**Algorithm 1** Trust Region Policy Search (TRPS)

**Input:** initial parameters $\theta_0 \in \mathbb{R}^n$, variance $\epsilon \in \mathbb{R}_+$

**Repeat until convergence,** $t = 0, 1, 2, \ldots$

1) Sample parameters $w^{(t)} \sim \mathcal{N}(\theta_t, \epsilon I)$. Execute policy with parameters $w^{(t)}$ and receive cost $\hat{J}_t$.
2) For $i = 0, \ldots, t$, compute importance weights

$$s_i = \frac{p(w^{(i)};\theta_t)}{p(w^{(i)};\theta_i)} = \frac{\exp\left(\frac{\epsilon^2}{2}(w^{(i)}-\theta_t)^T(w^{(i)}-\theta_t)\right)}{\exp\left(\frac{\epsilon^2}{2}(w^{(i)}-\theta_i)^T(w^{(i)}-\theta_i)\right)}$$

3) Compute weighted least-squares solution

$$\begin{bmatrix}\mathrm{svec}(\hat{H})\\ \hat{g} \\ \hat{J}_0\end{bmatrix} \leftarrow (\Phi^T S \Phi)^{-1}\Phi^T S \hat{J}$$

where

$$\Phi = \begin{bmatrix}\phi(w^{(0)})^T \\ \vdots \\ \phi(w^{(t)})^T\end{bmatrix}, \quad S = \mathrm{diag}(s_0, \ldots s_t).$$

4) Update distribution parameters

$$\theta_{t+1} \leftarrow \theta_t + \arg\min_{\|\Delta\theta\|^2 \le \epsilon} \frac{1}{2}\Delta\theta^T \hat{H}\Delta\theta + \hat{g}^T\Delta\theta.$$

## V. ANALYSIS AND RESULTS

Here we present a number experimental results both demonstrating our turbine system and evaluating the TRPS algorithm. The first set of experiments use exhaustive search

to sweep out the power curve for a number of different operating conditions of the turbine; this establishes the best possibly performance might might expect from an online policy search method, though of course is a much more time-consuming process. The second set of experiments departs somewhat from the rest of the paper, but is crucial for demonstrating the capabilities of the turbine platform: here we demonstrate individual pitch control at high rotor speeds, using feed-forward compensation based upon a learned dynamics model of the servo actuators. Finally, we evaluate the TRPS algorithm on the turbine and show that the method is able to quickly and reliably obtain near-optimal energy extraction after about one minute of execution time.

### A. Modeling and Power Output Analysis

Our first set of experiments characterize the power output of the turbine relative to a simulated model of the system, under a variety of different operating conditions. Specifically, we independently varied the (collective) blade pitch angle of the turbine from 4 to 17 degrees in 24 equal increments, and the resistance from 1.8 to 5 Ohms in 14 equal increments, for a total of 336 different operating conditions. For each setting, we let the turbine run for five seconds (to settle to its steady state), and recorded the resulting power.

In addition, we developed a simulated model of the system using the WT_Perf aerodynamics simulator [18], an industry standard tool developed by the National Wind Technology Center for predicting the the performance of wind turbines using blade element momentum (BEM) theory [4]. The simulator takes as input the physical properties of the system such as the number of blades and their shape, the lift and drag curves of the blade airfoil, and aerodynamic constants such as the density and viscosity of the air. It then computes the aerodynamic properties of the blades, along with momentum-based properties of the turbine power output as a whole, to compute steady-state power output for different operating conditions of the turbine.

The resulting power curves, for both the real and simulated systems, are shown in Figure 9. The figures show power output in Watts, indicated by color, as a function of pitch angle and tip speed ratio, the ratio of the blade's tip speed to the incoming wind speed. Figure 9 also emphasizes both the the benefits and drawbacks of simulation models. Qualitatively, the two power curves look quite similar: they both have a characteristic "triangle" shape, and the optimal operating conditions are at similar points. But the figures also differ in crucial respects: the optimal tip speed ratio on the real system is significantly higher than for the simulated system, and the range of near-optimal points is significantly smaller. Indeed, if one were to simply choose the operating conditions of the turbine based upon the simulated model (as is often done in practice), then we would be operating at a substantially suboptimal point. This highlights the usefulness of experimentation on real systems and the development of learning methods that can optimize performance without relying on an a priori model of the system.

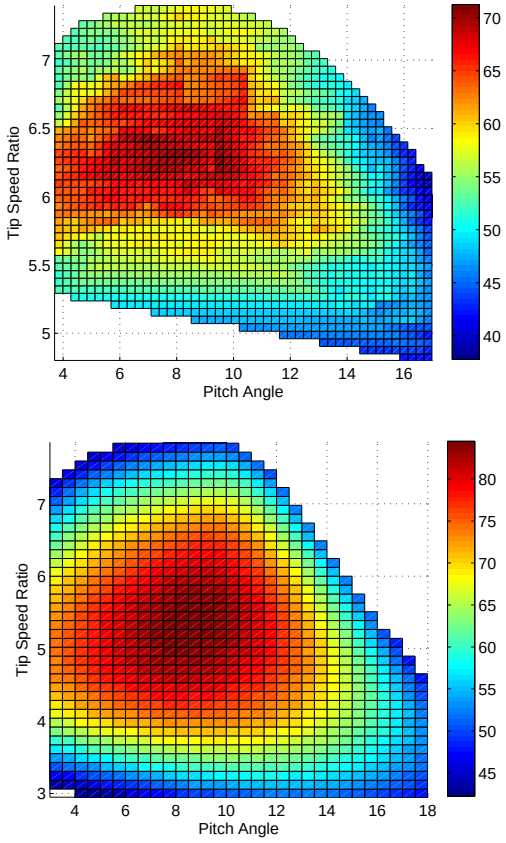


Fig. 9. Real (top) and simulated (bottom) power curves for the turbine under different operating conditions. Power levels indicated by color are in Watts. Note the different vertical scales of the two plots.

### B. Independent Pitch Control

Our second set of experiments demonstrate the feasibility of independent pitch control (IPC) on the turbine, a crucial design requirement for the system. IPC on a micro turbine is a challenging task: in order to maintain reasonable tip speed ratios, micro turbines must rotate much faster than large turbines, and IPC requires at a minimum that we be able to vary the blade pitch on the order of one rotational period, typically synchronized with the absolute rotor angle in order to account for spatial effects such as wind shear. Thus, we need both high frequency control and accurate compensation for the servo motor dynamics in order to accurately track the desired pitch angles.

To achieve accurate independent pitch control, we employ feedback compensation using a learned dynamics model. The dynamics of the motors are well-modeled by a second order system

$$\ddot{q} = k_1(q - u) + k_2\dot{q} + k_3\text{sign}(\dot{q}) + k_4 \qquad (10)$$

where $q$ denotes the motor angle, $u$ denotes the control input (corresponding to the desired angle setpoint), $k_1, \ldots, k_4$ are parameters of the model. The model is non-linear due to the sign term, which captures the effects of Coulomb friction. We fit this model to the data using system identification procedures: we generate data by commanding a sequence of

"chirp" commands (sinusoidal inputs and varying frequencies and magnitudes), then minimize the *simulation error* of the model (the deviation between the observed and predicted sequences, simulated over the entire sequence of inputs) using non-linear optimization. We collect this data when the turbine is stationary, so the models do not attempt to capture any of the aerodynamic forces on the blades; however, since the inertia in the servos is typically much larger than the aerodynamic torques associated with the the blades, these models still perform well when the turbine is in operation.

Supposing we want to the pitch angles to track some know function of rotor angle $q^d = f(\theta)$, we analytically differentiate this function to achieve desired velocity and accelerations

$$\dot{q}^d = f'(\theta)\dot{\theta}, \quad \ddot{q}^d = f''(\theta)\dot{\theta} + f'(\theta)\ddot{\theta} \tag{11}$$

and assume for simplicity that the rotor speed is constant $\ddot{\theta} = 0$. Instead of commanding the desired angle $u = q^d$, we invert the dynamics model and command the input

$$u = q^d + (\ddot{q}^d - k_2\dot{q}^d - k_3\text{sign}(\dot{q}^d) - k_4)/k_1 \tag{12}$$

which is simply a feed-forward compensator to the desired position, based upon the dynamical system.

Figure 10 shows the pitch tracking performance using the model-based feed-forward compensation versus simple PD control. We commanded a desired angle of $q^d(\theta) = 12 - 2.9\sin(\theta)$, with the rotor spinning at 525 RPM. The top plot of Figure 10 shows tracking performance of the feed-forward controller: in this case we are able to track the reference trajectory quite accurately even at high speeds, with an RMSE of 0.26 degrees. In contrast, the bottom plot shows the performance of a simple PD controller attempting to track the trajectory; here the settling time of the motor dynamics cause the actual trajectory to lag significantly behind the desired trajectory, a well-known effect in PD control. While it would also be possible to adaptively tune the phase and amplitude of the desired trajectory to make the PD controller match the desired behavior, the forward controller does this automatically through its model of the dynamics.

### C. Policy Search for Online Parameter Optimization

Finally, our last experiments evaluate our proposed trust region policy search method on the task of optimizing power output (using collective pitch) at below the turbine's rated speed. The policy parameters $w \in \mathbb{R}^2$ here are simply the collective blade pitch angle and resistance of the load (since the resistance is more directly controllable, we use it as our policy parameters, but could equivalently parametrize the policy in terms of tip speed ratio). To evaluate the method, we randomly initialized policies with uniformly random pitch and resistance values within 10 to 20 degrees, and resistance between 5 and 0.33 ohms. We then ran our trust region policy search approach for 40 iterations, using the power output averaged over two seconds (after three seconds of settling time) as the reward signal (the negative cost function).

Figure 11 shows the evolution of the reward versus iteration number, averaged over 10 trials (with different

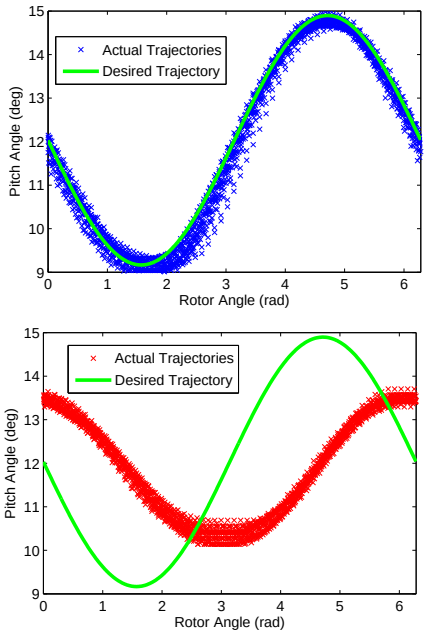


Fig. 10. Individual Pitch Control tracking performance using learned feed-forward compensation (top), and PD control alone (bottom).

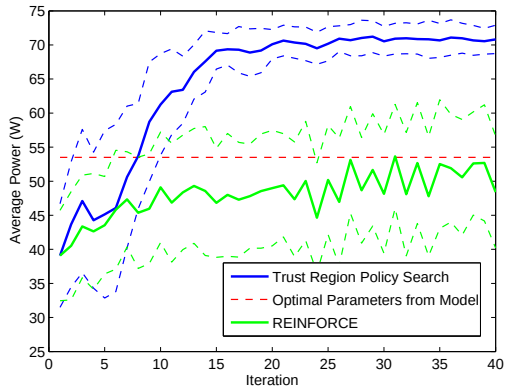


Fig. 11. Average reward versus iteration number, with 95% confidence intervals for our trust region policy search method optimizing power output.

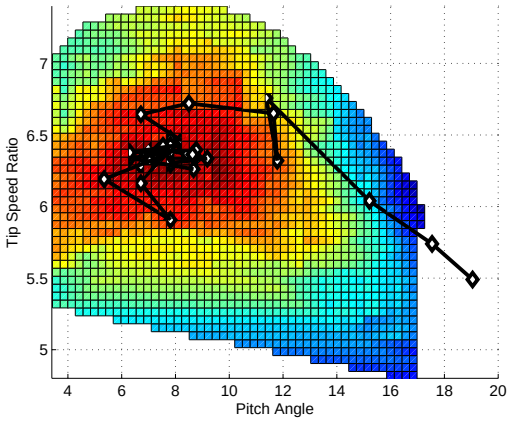


Fig. 12. Evolution of policy parameters for a typical run of policy search.

starting parameters), along with 95% confidence intervals. Also shown is the average reward obtained by trusting the WT_Perf simulation model and using its prescribed optimal operating point. Despite receiving very little feedback from the system, just the average power for a small number of samples, without any model of the system, our algorithm is able to quickly obtain near-optimal parameter settings, typically within about 15 iterations (75 seconds of real-time operation). Figure 12 shows the evolution of the policy parameters for a typical run of the policy search; as expected the method quickly gets to (and remains in) a near-optimal region of the parameter space. In contrast the REINFORCE algorithm (using both the importance sampling technique discussed in [30] and the optimal baseline technique from [25]), performs significantly worse in this domain.

## VI. CONCLUSIONS AND FUTURE WORK

This paper has presented an actuated wind turbine system. We presented the physical system design, illustrated the power output of the turbine and demonstrated accurate individual pitch control via feed-forward compensation. We proposed and evaluated the Trust Region Policy Search method, an online learning algorithm similar to Reinforcement Learning policy search methods, but which explicitly models second order information about the cost function and uses past data efficiently. We applied the algorithm to power optimization in Region II wind speeds, and show that it is able to quickly and reliably achieve near-optimal power output on our turbine. Next steps for the research involve integrating the policy search and individual pitch control mechanisms to cope with static obstructions in the incoming airflow. Building upon this, the eventual goal is to develop a system that can autonomously adjust to dynamic disturbances, such as those caused by an upwind turbine, using a combination of both model-based and model-free optimization techniques.

## ACKNOWLEDGMENTS

We thank Morgan Quigley for helpful discussions, Paul Stearns of Alternate Power Technologies, Inc for the Extractor turbine and his assistance in modifying hub parts for our turbine, and Ron Wiken for his help designing and building the wind tunnel. J. Zico Kolter is supported by an NSF Computing Innovations Fellowship.

## REFERENCES

[1] Various, "Annual energy review 2009," U.S. Energy Information Administration, Tech. Rep., 2009.

[2] ——, "Annual energy outlook 2011," U.S. Energy Information Administration, Tech. Rep., 2010.

[3] U. D. of Energy, "20% wind energy by 2030: Increasing wind energy's contribution to u.s. electricty supply," U.S. Department of Energy, Tech. Rep., 2008.

[4] J. Manwell, J. McGowan, and A. Roberts, *Wind Energy Explained: Theory, Design, and Application*. Wiley, 2009.

[5] K. E. Johnson, L. Fingersh, M. J. Balas, and L. Y. Pao, "Methods for increasing region 2 power capture on a variable-speed wind turbine," *Transactions of the American Society of Mechanical Engineers*, vol. 126, 2004.

[6] T. Sebastian and M. Lackner, "Characterization of the unsteady aerodynamics of offshore floating wind turbines," *Wind Energy*, vol. 14, 2011.

[7] R. J. Williams, "Simple statistical graident-follwoing algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 23, 1992.

[8] P. Glynn, "Likelihood ratio gradient estimation: an overview," in *Proceedigns of the 1987 Winter Simulation Conference*, 1987.

[9] T. Burton, D. Sharpe, N. Jenkins, and E. Bossanyi, *Wind Energy Handbook*. Wiley, 2001.

[10] T.K.Barlas and G. van Kuik, "Review of state of the art in smart rotor control research for wind turbines," *Progress in Aerospace Sciences*, vol. 46, no. 1, pp. 1–27, 2010.

[11] L. Y. Pao and K. E. Johnson, "Control of wind turbines: Approaches, challenges, and recent developments," *IEEE Control Systems Magazine*, vol. 31, pp. 44–62, 2011.

[12] K. Pierce, "Control methods for improved energy capture below rated power," in *Proceedings of the ASME/JSME Joint Fluids Engineering Conference*, 1999.

[13] J. Youqin, Y. Zhongging, and C. Binggang, "A new maximum power point tracking control scheme for wind generation," in *Proceedings of the IEEE International Conference on Power Systems Technology*, 2002.

[14] E. Koutroulis and K. Kalaitzakis, "Deisng of a maximum power tracking system for wind-energy conversion applications," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 2, pp. 386–394, 2006.

[15] T. Hawkins, W. White, G. Hu, and F. D. Sahneh, "Wind turbine power capture control with robust estimation," in *Proceedings of the Dynamic Systems and Control Conference*, 2010.

[16] J. Creaby, Y. Li, and J. Seem, "Maximizing wind turbine energy capture ssing multi-variable extremum seeking control," *Wind Engineering*, vol. 33, no. 4, pp. 361–387, 2009.

[17] "NWTC Design Codes (FAST by Jason Jonkman, Ph.D.)," http://wind.nrel.gov/designcodes/simulators/fast/. Last modified 08-March-2012; accessed 15-March-2012.

[18] "NWTC Design Codes (WT_Perf by Marshall Buhl)," http://wind.nrel.gov/designcodes/simulators/wtperf/. Last modified 17-February-2011; accessed 15-March-2012.

[19] L. Fingersh and K. Johnson, "Controls advanced research turbine (cart) commisioning and baseline data collection," National Renewable Energy Laboratory, Tech. Rep., 2002.

[20] A. Wright, L. Fingersh, and M. Balas, "Testing state-space controls for the controls advanced research turbine," in *Proceedings of the AIAA Aerospace Sciences Meeting and Exhibit*, 2006.

[21] K. A. Stol, W. Zhao, and A. D. Wright, "Individual blade pitch control for the controls advanced research turbine (cart)," *Transactions of the ASME*, vol. 128, pp. 498–505, 2006.

[22] D. Adams, "Model identification and operating load characterization for a small horizontal axes wind turbine rotor using integrated blade sensors," in *Proceedings of the Dynamic Systems and Control Conference*, 2010.

[23] K. L. Moore, "Iterative learning control: an expository overview," *Applied and Computational Controls, Signal Processing, and Circuits*, vol. 1, no. 1, pp. 151–214, 1999.

[24] K. Ariyur and M. Krstic, *Real-Time Optimization by Extremum-Seeking Control*. Wiley, 2003.

[25] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *Journal of Machine Learning Research*, vol. 5, pp. 1471–1530, 2004.

[26] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proceedings of the IEEE Conference on Intelligent Robotics Systems*, 2006.

[27] A. S. Huang, E. Olson, and D. Moore, "Lcm: Lightweight communications and marshalling," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2010.

[28] M. Jabri and B. Flower, "Weight perturbation: an optimal architecture and learning technique for analog VSLI feedforward and recurrent multilayer networks," *IEEE Transactions on Neural Networks*, vol. 3, pp. 154–147, 1992.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[30] J. Tang and P. Abbeel, "On a connection between importance sampling and the likelihood ratio policy gradient," in *Neural Information Processing Systems*, 2011.