Introduction to Reinforcement Learning
A Short Course:
Floating Offshore Wind Power Case Study

Scott Moura

July 2020

1 Introduction & Background

In this problem, we explore reinforcement learning for floating offshore wind turbines.

Locating wind farms at sea provides access to higher and more consistent wind speeds. Most offshore wind turbines are mounted on the sea floor to ensure structural stability. However, this limits offshore wind farms to shallow depths ($\leq$ 60 meters), whereas high speed wind typically occurs in deep water ($\geq$ 60 meters)[1][2]. Balancing a floating wind turbine to be upright, subject to transient wind gusts and ocean waves provides a fascinating and important technological challenge.

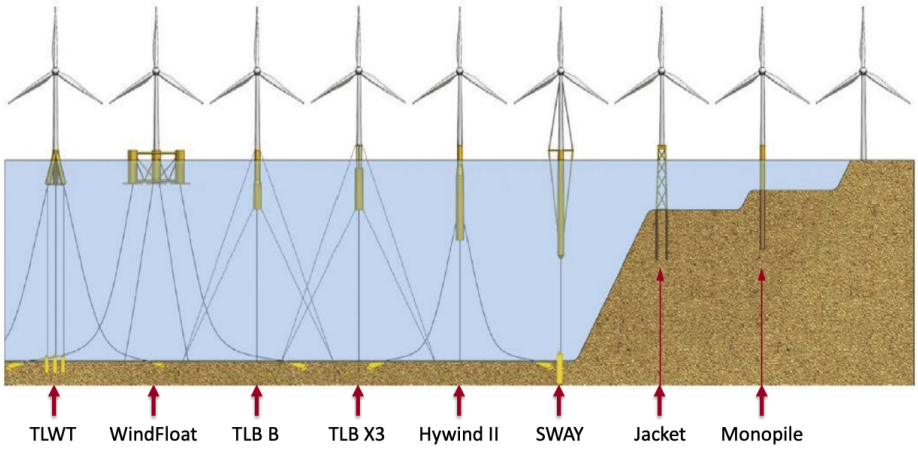

Figure 1: Various configurations for Offshore Floating and Bottom-Fixed Wind Turbines.

Consider the floating wind turbine schematic in Fig. 3. For this problem, consider rotational dynamics in the 2D plane only. The wind turbine is modeled as an inverted pendulum. That is, the nacelle and turbine blades (top part) have mass m. The mast has height L and is rigid and massless. The turbine's angle with vertical is denoted θ, and it is free to pivot on the float. Do not consider vertical and horizontal movements. The nacelle and blades experience transient force $F_w(t)$ from the wind. It also experiences a transient torque $\tau_s(t)$ from the sea. The float has a motor, and can compensate for angular displacements with controlled torque $\tau(t)$.

[1] A. Myhr, C. Bjerkseter, A. Ågotnes, T. Nygaard, Levelised cost of energy for offshore floating wind turbines in a life cycle perspective, Renewable Energy, Vol. 66, pp. 714-728, June 2014;

[2] J. Jonkman, S. Butterfield, W. Musial, and G. Scott, Definition of a 5-MW Reference Wind Turbine for Offshore System Development. Technical Report NREL/TP-500-38060, February 2009.

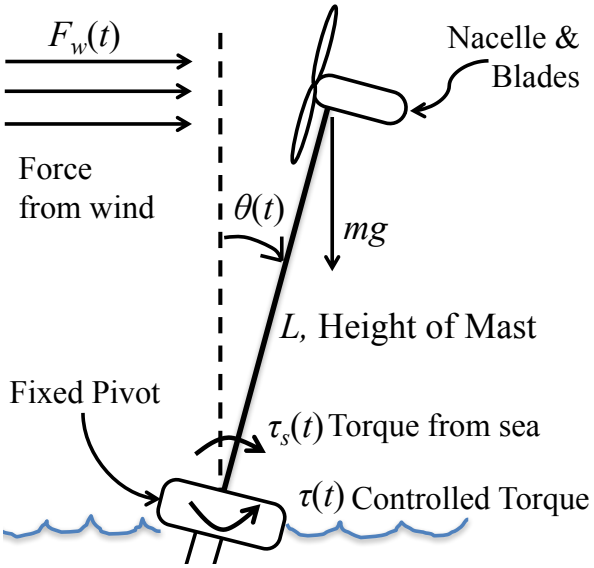

Figure 2: Schematic of floating wind turbine, including wind force, sea torque, and controlled torque.

1.1 Control Objective

Our control objective is to design a RL controller that stabilizes the turbine in the upright position, using controlled torque applied at the pivot. Online RL algorithms are appealing because

- Accurately modeling the wind force $F_w(t)$ and torque from the sea $\tau_s(t)$ can be very difficult. Instead, we seek to learn a controller online that is robust to these disturbances.
- There are unmodeled physics. We focus on one degree-of-freedom dynamics, but there are six degrees-of-freedom in reality.

Our control objective is to design a reinforcement learning controller that stabilizes the turbine in the upright position, using controlled torque applied at the pivot.

2 Mathematical Model

Application of Newton's second law in rotational coordinates yields the following equations of motion.

$$mL^2 \frac{d\omega}{dt}(t) = mgL \sin\theta(t) + F_w(t) L \cos\theta(t) + \tau_s(t) - \tau(t) \tag{1}$$

$$\frac{d\theta}{dt}(t) = \omega(t) \tag{2}$$

Note the inputs are divided into two categories:

- Controllable inputs: controlled torque $\tau(t)$
- Uncontrollable inputs: Torque from the sea $\tau_s(t)$; Force from wind $F_w(t)$

The uncontrollable inputs from the sea and wind represent exogeneous disturbances. Now, since our RL algorithms operate on discrete-time dynamical systems, we will approximate the time derivatives using the forward Euler method: $\frac{d}{dt}x(t) \approx [x(t+\Delta t) - x(t)]/\Delta t$. This yields:

$$\omega_{k+1} = \omega_k + \frac{g\Delta t}{L}\sin\theta_k + \frac{F_{w,k}\Delta t}{mL}\cos\theta_k + \frac{\Delta t}{mL^2}[\tau_{s,k} - \tau_k] \tag{3}$$

$$\theta_{k+1} = \theta_k + \Delta t \cdot \omega_k \tag{4}$$

where continuous time dimension t has been replaced by discrete time index k.

2.1 Linearized Model

The equations of motion are nonlinear in the state θ, due to the sin and cos terms. The model can be easily linearized using the small angle approximation, where for small θ:

- $\sin\theta = \theta$
- $\cos\theta = 1$

The small angle approximation applied to the equations of motion yields:

$$\omega_{k+1} = \omega_k + \frac{g\Delta t}{L}\theta_k + \frac{F_{w,k}\Delta t}{mL} + \frac{\Delta t}{mL^2}\left[\tau_{s,k} - \tau_k\right] \tag{5}$$

$$\theta_{k+1} = \theta_k + \Delta t \cdot \omega_k \tag{6}$$

This can be written in matrix-vector form as:

$$\underbrace{\begin{bmatrix}\omega_{k+1}\\ \theta_{k+1}\end{bmatrix}}_{=x_{k+1}} = \underbrace{\begin{bmatrix}1 & \frac{g\Delta t}{L}\\ \Delta t & 1\end{bmatrix}}_{=A}\underbrace{\begin{bmatrix}\omega_k\\ \theta_k\end{bmatrix}}_{=x_k} + \underbrace{\begin{bmatrix}\frac{-\Delta t}{mL^2}\\ 0\end{bmatrix}}_{=B}\underbrace{\tau_k}_{=u_k} + \underbrace{\begin{bmatrix}\frac{\Delta t}{mL} & \frac{\Delta t}{mL^2}\\ 0 & 0\end{bmatrix}}_{=B_d}\underbrace{\begin{bmatrix}F_{w,k}\\ \tau_{s,k}\end{bmatrix}}_{=d_k} \tag{7}$$

or more compactly:

$$x_{k+1} = Ax_k + Bu_k + B_d d_k \tag{8}$$

3 Optimal Control Formulation

We are now positioned to formulate the optimal control problem. Recall the objective is to stabilize the wind turbine in the upright position, when subjected to disturbances from wind and sea.

$$\text{minimize} \qquad J = \sum_{k=0}^{\infty}\left[x_k^T Q x_k + R u_k^2\right] \tag{9}$$

$$\text{subject to:} \qquad \omega_{k+1} = \omega_k + \frac{g\Delta t}{L}\sin\theta_k + \frac{F_{w,k}\Delta t}{mL}\cos\theta_k + \frac{\Delta t}{mL^2}\left[\tau_{s,k} - \tau_k\right] \tag{10}$$

$$\theta_{k+1} = \theta_k + \Delta t \cdot \omega_k \tag{11}$$

Alternatively, we may consider the linearized model:

$$\text{minimize} \qquad J = \sum_{k=0}^{\infty}\left[x_k^T Q x_k + R u_k^2\right] \tag{12}$$

$$\text{subject to:} \qquad x_{k+1} = Ax_k + Bu_k + B_d d_k \tag{13}$$

4 Actor-Critic RL

We consider an actor-critic RL controller. Specifically, we will apply value function approximation and policy approximation.

4.1 Value Function Approximation

Consider a quadratic value function:

$$V(x_k) = x_k^T P x_k = W^T \phi(x_k) \tag{14}$$

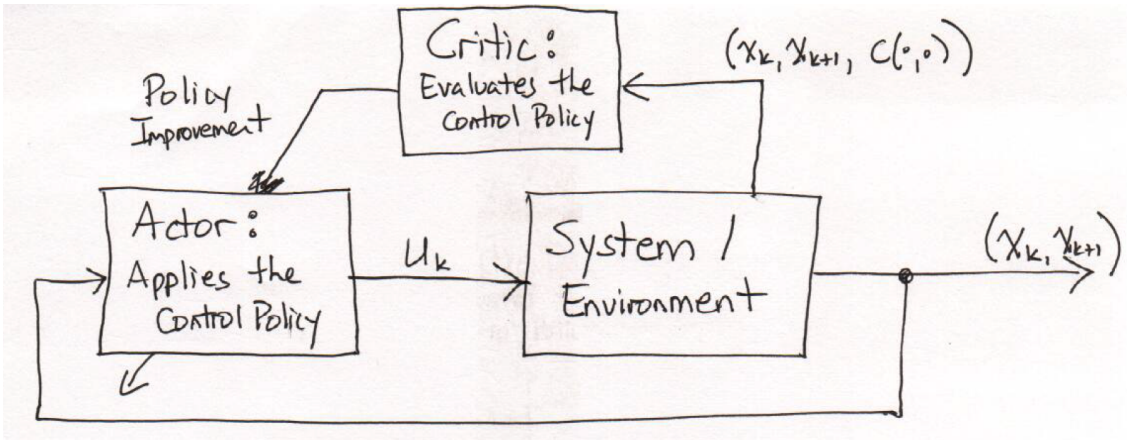

Figure 3: Schematic of actor-critic RL algorithm.

For the nonlinear model, there is no guarantee that the value function is quadratic. However, for small angles, the problem can be accurately approximated by an LQR formulation. Consequently, a quadratic value function is a very reasonable first choice. Consider:

$$x_k = \begin{bmatrix} \omega_k \\ \theta_k \end{bmatrix}, \qquad P = \begin{bmatrix} p_{11} & p_{12} \\ * & p_{22} \end{bmatrix} \tag{15}$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}, \qquad \phi(x_k) = \begin{bmatrix} \omega_k^2 \\ \omega_k \theta_k \\ \theta_k^2 \end{bmatrix} \tag{16}$$

4.2 Policy Approximation

We similarly approximate the control policy as linear in the state:

$$u_k = \pi(x_k) = K x_k = U^T \sigma(x_k) \tag{17}$$

where $U = K^T$ and $\sigma(x_k) = x_k$. We are ready to construct the actor-critic RL algorithm.

4.3 Actor-Critic Algorithm

0. **Initialization:** Select any admissible policy weights U_m. Set $m = 0$.

1. **Policy Evaluation:** Run control policy $\pi^m(x) = U_m^T \sigma(x)$ on the environment/system for one episode. Collect L measured data tuples $(x_k, x_{k+1}, c(x_k, \pi(x_k)))$. Find the least squares solution w.r.t. W_m for regression model (a.k.a. Bellman equation)

$$\underbrace{\begin{bmatrix} \vdots \\ x_k^T Q x_k + R\left(U_m^T \sigma(x_k)\right)^2 \\ \vdots \end{bmatrix}}_{=C, L \times 1} = \underbrace{\begin{bmatrix} \vdots \\ [\phi(x_k) - \phi(x_{k+1})]^T \\ \vdots \end{bmatrix}}_{=\Phi, L \times n_w} \underbrace{W}_{n_w \times 1} \tag{18}$$

   written compactly as $C = \Phi W$. For example, the ordinary least squares solution is

$$W_m \leftarrow W^\star = (\Phi^T \Phi)^{-1} \Phi^T C \tag{19}$$

2. **Policy Improvement:** Find an improved policy by solving the following optimization problem

$$\text{minimize}_U \quad T(U) = x_k^T Q x_k + R u_k^2 + W^T \phi(x_{k+1}) \tag{20}$$

$$= x_k^T Q x_k + R(U^T \sigma(x_k))^2 + W^T \phi(x_{k+1}) \tag{21}$$

where x_{k+1} is given by either the nonlinear model in (10)-(11) or the linearized model in (13). If we apply gradient descent to solve the minimization problem, then the algorithm in both the nonlinear and linearized cases is given by:

$$U_{j+1} = U_j - \beta \cdot \frac{\partial T}{\partial U}(U_j), \qquad \text{for } \beta > 0 \tag{22}$$

where the gradient is:

$$\frac{\partial T}{\partial U} = \left[2R(U^T\sigma(x_k)) + W^T\phi(x_{k+1}) \cdot B\right] \cdot \sigma(x_k) \tag{23}$$

Amazingly, the only details about the model dynamics that we require is the B matrix in (7). After gradient descent has converged, and we arrive at final iterate U_{j+1}, then...
Set $U_{m+1} = U_{j+1}$
Set $m \leftarrow m + 1$. Go to Step 1.