# Experiments on 2-DOF Helicopter Using Approximate Dynamic Programming

Anthony Birge, Andrew Fandel, Advisor: Dr. Suruz Miah

Department of Electrical and Computer Engineering · Bradley University · Peoria, IL

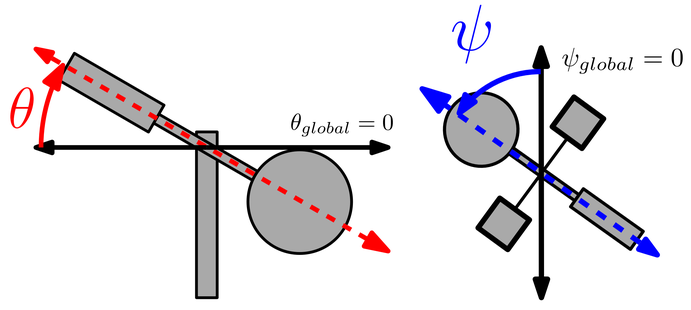
BRADLEY University

## Objectives

- Utilize the Quanser AERO as the platform system to be controlled by a machine learning controller (approximate dynamic programming)
- Simulate real-time control of the Quanser AERO using industry-standard software e.g. V-REP and MATLAB
- Model the controller architecture in Simulink which can access the Quanser AERO directly via licensed software
- Embed the controller on a Raspberry Pi 3

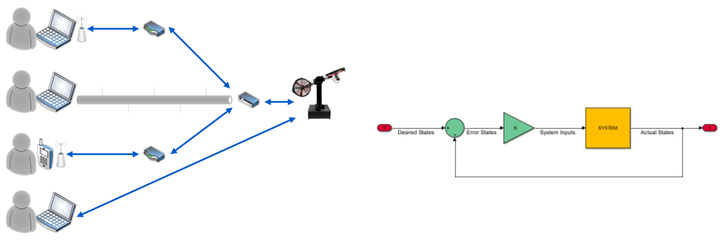
## Problem Setup

- Derive the state-space model of the physical system in order to implement model-based reinforcement learning
- Verify proper controller operation via MATLAB simulations prior to hardware implementation
- Construct the Quanser AERO in Virtual-Robot Experimentation Platform (V-REP) in order to obtain real-time simulations
- Implement the model-based learning controller in an industry-standard graphical programming language (e.g. Simulink) for embedding C-code on a Raspberry Pi 3




(a) (b)

Figure 1: Quanser AERO orientation. (a) pitch [$\theta$] and (b) yaw [$\psi$] are the directly measured system states.




(a) (b)

Figure 2: (a) High-level system block diagram. Control of the Quanser AERO can be achieved through various interfaces. (b) Generic state-feedback block diagram. System control is realized by determining the proper K value, but determining this value can be difficult or impossible.

## Model-Based Reinforcement Learning

### Approximate Dynamic Programming Overview

**(1)** $t = k\tau$ where $\tau$ is given

**(2)** Advance the system by applying determined inputs to the system and measure system states

**(3)** Calculate state error as $\mathbf{e}[k] = \mathbf{x}^{\text{ref}}[k] - \mathbf{x}[k]$, save state error

**(4)** If $t \neq T$, increment $k$ and start at **(1)**

**(5)** If $t = T$, update $K$ using collected error data

**(5a)** Use error states since last update of $K$ as inputs to the neural network

**(5b)** Recursively determine $\mathbf{w}_c$ values for each of the instances

**(5c)** Approximate new $\mathbf{w}_c$ value using regression analysis of the individually calculated $\mathbf{w}_c$ values

**(6)** Calculate new $K$ using $\mathbf{w}_c$ values

**(7)** Increment $k$ and start at **(1)**

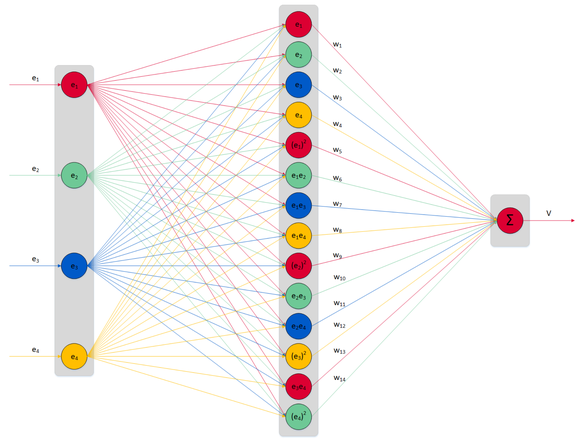
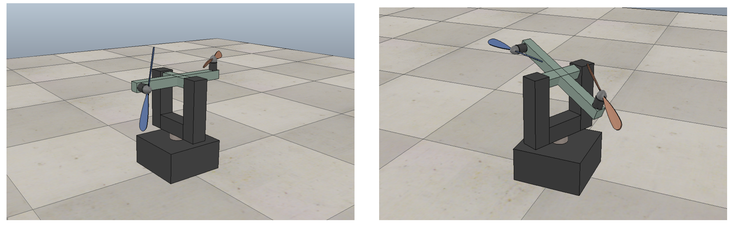



Figure 3: Neural network used in approximate dynamic programming. The neural network optimizes its weights to minimize $V$, the value function.

## Real-Time Simulation (V-REP)

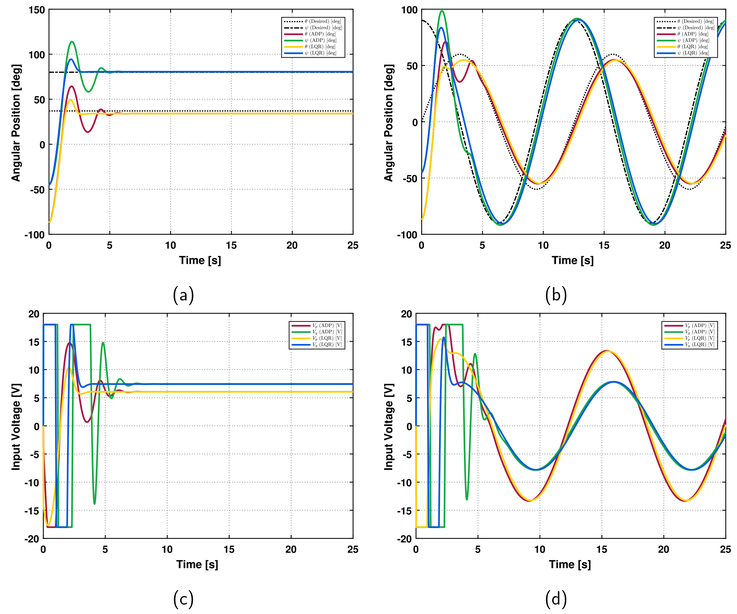
- Developed a dynamic model of the Quanser AERO using V-REP
- The dynamic model, shown in Figure 4(a) and Figure 4(b), can serve as a platform for testing other control techniques in future work



(a) (b)

Figure 4: Dynamic model of the Quanser AERO in V-REP.

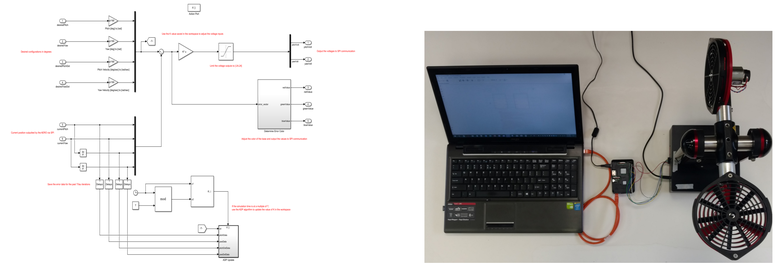
## MATLAB Simulation Results




(a) (b) (c) (d)

Figure 5: MATLAB simulation results for various trajectories. (a) and (c) represent steady state tracking. (b) and (d) represent tracking of a sinusoidal waveform in both $\theta$ and $\psi$.

## Hardware Implementation



(a) (b)

Figure 6: (a) Simulink model of the approximate dynamic programming algorithm. (b) The SPI (Serial Peripheral Interface) protocol is used for communication between embedded system and the Quanser AERO. The SPI protocol was also modeled in Simulink for C-code generation.

## Conclusion and Future Work

- Successfully simulated controller in MATLAB and V-REP and successfully implemented controller in Simulink and Raspberry Pi 3
- Began implementation of position control using Android cell phone communication with Raspberry Pi 3
- Future work will include tuning the algorithm, extending implementation applications, and updating the framework provided