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
1501 W. Bradley Avenue
Peoria, IL 61625

Saturday, April 28, 2018

Outline

- Most physical systems are nonlinear - one example being a helicopter with the stochastic nature of weather
- Control techniques frequently only apply to linear systems
- Machine learning has seen significant growth in the areas of diagnostics, forecasting, and optimization
- With the growth of machine learning, is there a way to learn the most efficient control strategy for a nonlinear system
- One such machine learning algorithm is a reinforcement learning model-based approach known as approximate dynamic programming

- Investigate the use of approximate dynamic programming in nonlinear systems
- Demonstrate approximate dynamic programming on a 2-DOF helicopter, the Quanser AERO, using theoretical simulations
- Design virtual simulation platforms for the Quanser AERO
- Implement approximate dynamic programming on the Quanser AERO
- Extend approximate dynamic programming via an embedded system and smart phone
- Research the Quanser AERO for future teaching experiments at Bradley University

- **2-DOF** - 2 Degrees-Of-Freedom
- **ADP** - Approximate Dynamic Programming
- **FAA** - Federal Aviation Administration
- **LQR** - Linear Quadratic Regulator
- **SPI** - Serial Peripheral Interface
- **UDP** - User Datagram Protocol
- **V-REP** - Virtual Robot Experimentation Platform

- $J_p(J_y)$ - Total moment of inertia about pitch (yaw) axis
- $D_p(D_y)$ - Damping constant about pitch (yaw) axis
- $K_{\mathbf{sp}}$ - Stiffness about pitch axis
- $K_{\mathbf{pp}}(K_{\mathbf{py}})$ - Torque thrust gain acting on pitch from pitch (yaw) rotor
- $K_{\mathbf{yy}}(K_{\mathbf{yp}})$ - Torque thrust gain acting on yaw from yaw (pitch) rotor
- $V_0(V_1)$ - Applied voltage to pitch (yaw) motor
- $\theta(t)[\psi(t)]$ - Pitch [yaw] angle at time $t \geq 0$

- Most systems can be controlled by state-feedback
- Determining the most efficient gains of the system can be difficult
- ADP uses an approximate model of the system and measured data to estimate possible future system states
- Dynamically, ADP adjusts the gains as the system progresses through time

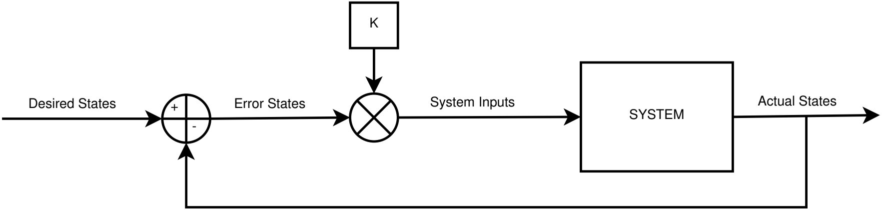


Figure: Generic state-feedback block diagram.

- Linearized system model:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{1}$$

- Error of system model:

$$\mathbf{e}[k+1] = \mathbf{f}(\mathbf{e}[k]) + \mathbf{G}\mathbf{u}[k] \tag{2}$$

$$\mathbf{f}(\mathbf{e}[k]) = e[k] + T\mathbf{A}e[k] - T\mathbf{A}\mathbf{x}^{[d]} \tag{3a}$$

$$\mathbf{G} = -T\mathbf{B} \tag{3b}$$

- The cost function to be minimized by ADP is

$$J(\mathbf{u}) = \sum_{k=0}^{\infty} \left(\mathbf{e}^T(k)\mathbf{Q}\mathbf{e}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) \right) \tag{4}$$

- Cost–to–go function (value function):

$$V(\mathbf{e}(k)) = \sum_{\kappa=k}^{\infty} \left(\mathbf{e}^T(\kappa)\mathbf{Q}\mathbf{e}(\kappa) + \mathbf{u}^T(\kappa)\mathbf{R}\mathbf{u}(\kappa) \right) \tag{5}$$

- The cost–to–go function can be rewritten as

$$V(\mathbf{e}(k)) = \mathbf{e}^T(k)\mathbf{Q}\mathbf{e}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) + V(\mathbf{e}(k+1)) \tag{6}$$

- Optimal control inputs found by

$$\mathbf{u}^*(k) = \text{argmin}_{\mathbf{u}(k)} \left(\mathbf{e}^T(k)\mathbf{Q}\mathbf{e}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) + V^*(\mathbf{e}(k+1)) \right) \quad (7)$$

- The minimization problem is solved by using the discrete–time Hamilton–Jacobi–Bellman equation:

$$V^*(\mathbf{e}(k)) = \min_{\mathbf{u}(k)} \left(\mathbf{e}^T(k)\mathbf{Q}\mathbf{e}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) + V^*(\mathbf{e}(k+1)) \right) \quad (8)$$

- Calculating the gradient of the right side of (8) yields

$$\frac{\partial}{\partial \mathbf{u}(k)} \left(\mathbf{e}(k)^T \mathbf{Q}\mathbf{e}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) \right) + \left(\frac{\partial \mathbf{e}(k+1)}{\partial \mathbf{u}(k)} \right)^T \nabla V^*(\mathbf{e}(k+1)) = \mathbf{0} \quad (9)$$

- The optimal inputs at time instant k are given by

$$\mathbf{u}^*(k) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{G}^T \nabla V^*(\mathbf{e}(k+1)) \quad (10)$$

- The cost–to–go function can be expressed as a quadratic function:

$$V(\mathbf{e}(k)) = \mathbf{e}^T(k)\mathbf{P}\mathbf{e}(k) \quad (11)$$

- The cost–to–go function can then be approximated by

$$V(\mathbf{e}(k)) = (\mathrm{vec}(\mathbf{P}))^T(\mathbf{e}(k) \otimes \mathbf{e}(k)) \equiv \mathbf{w}_c^T \phi(\mathbf{e}(k)) \quad (12)$$

where $\otimes$ is the Kronecker product operator and the weight $\mathbf{w}_c = \mathrm{vec}(\mathbf{P})$, where the operator $\mathrm{vec}(\mathbf{P})$ forms the vector by stacking columns of the matrix $\mathbf{P}$

- The weight vector, $\mathbf{w}_c$, is approximated using a critic neural network using collected error data as inputs

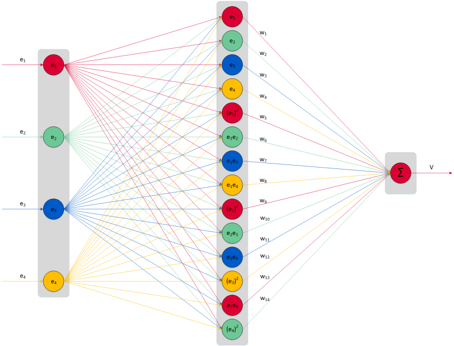


Figure: Critic neural network.

- The estimated cost–to–go function is defined as

$$\hat{V}(\mathbf{e}(k)) \cong \mathbf{w}_c^T \phi(\mathbf{e}(k)) \tag{13}$$

- The target value function is determined by using

$$V(\mathbf{e}(k)) = \mathbf{e}^T(k)\mathbf{Q}\mathbf{e}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k) + \mathbf{w}_c^T \phi(\mathbf{e}(k+1)) \tag{14}$$

- The least square error is then defined as

$$\delta_c = \frac{1}{2}\sum_{\kappa=0}^{\bar{n}-1}[V(\mathbf{e}(k)) - \hat{V}(\mathbf{e}(k))]^2 \tag{15}$$

- The weight vector, $\mathbf{w}_c$, that minimizes the sum–of–square error δ_c is given by

$$\mathbf{w}_c = \left(\mathbf{\Lambda}^T\mathbf{\Lambda}\right)^{-1}\mathbf{\Lambda}^T\mathbf{V} \tag{16}$$

where the matrix $\mathbf{\Lambda} = [\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{\bar{n}-1}]^T$ with $\mathbf{a}_\kappa = \phi^T(\mathbf{e}(k+\kappa))$ and $\mathbf{V} = [v_0, v_1, \ldots, v_{\bar{n}-1}]^T$ with $v_\kappa = V(\mathbf{e}(k+\kappa))$ for $\kappa = 0, 1, \ldots, \bar{n}-1$

- We then extract our state-feedback gain using

$$\mathbf{u}^*(k) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{G}^T\nabla V^*(\mathbf{e}(k+1)) \tag{17}$$

ADP
Flowchart

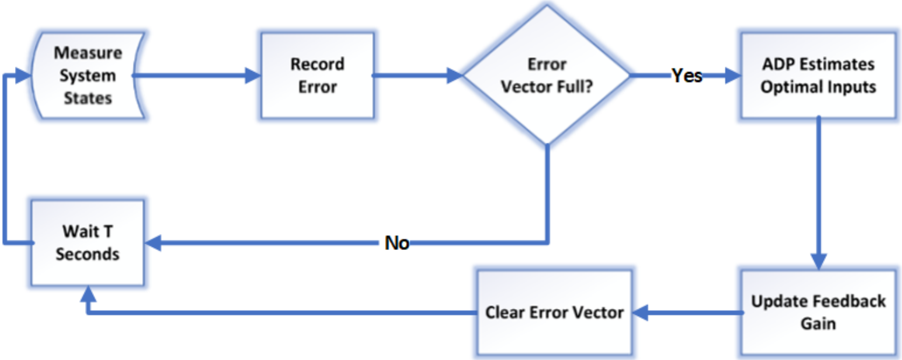


Figure: ADP flowchart.

- Advanced control research platform used in education and industry
- Configurable as a half-quadcopter or 2-DOF helicopter
- System inputs - motor voltages
- System outputs - pitch, yaw, angular pitch velocity, and angular yaw velocity
- Difficulty arises due to the coupling effect

Figure: Quanser AERO configured as 2-DOF helicopter. Image courtesy of Quanser [8].

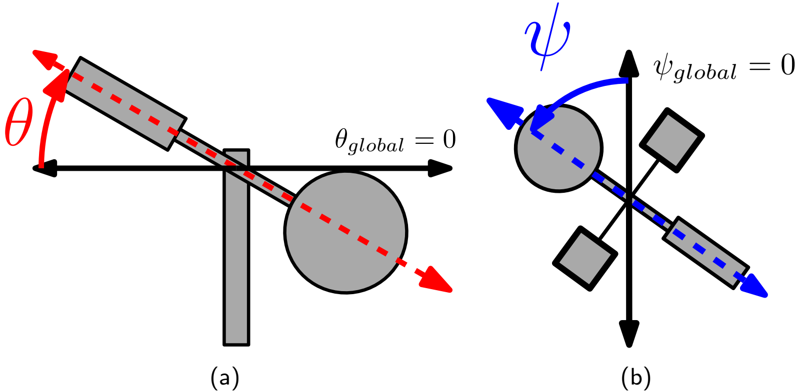


Figure: (a) pitch [θ] and (b) yaw [ψ] are the measured system states.

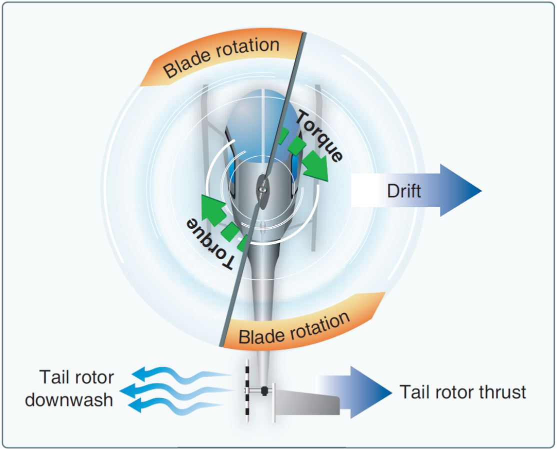


Figure: Controlling a helicopter can be difficult due to a coupling effect. Image courtesy of the FAA Helicopter Pilot's Handbook [1].

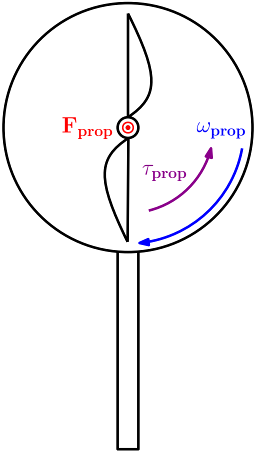


Figure: Quanser AERO rotor.

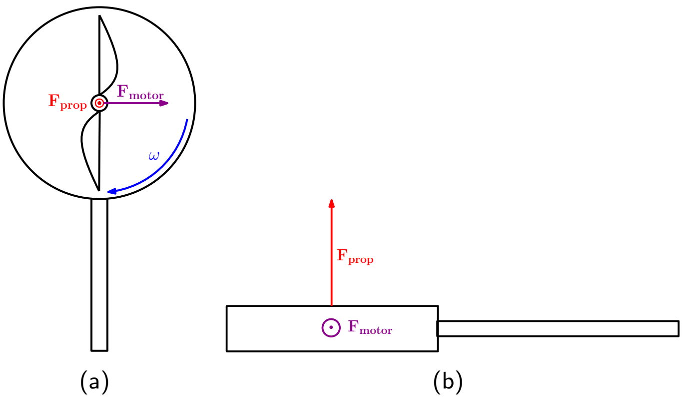


Figure: (a) Force diagram (top view) and (b) force diagram (side view).

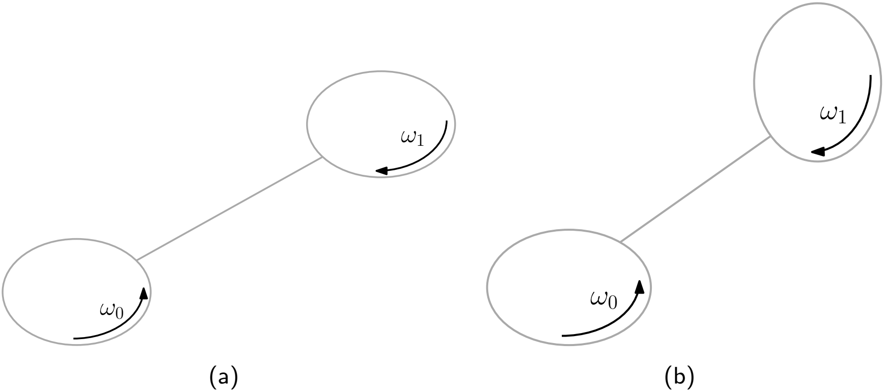


Figure: (a) Half-quadcopter blade rotations ($V_0 = V_1$) and (b) 2-DOF helicopter blade rotations ($V_0 = V_1$).

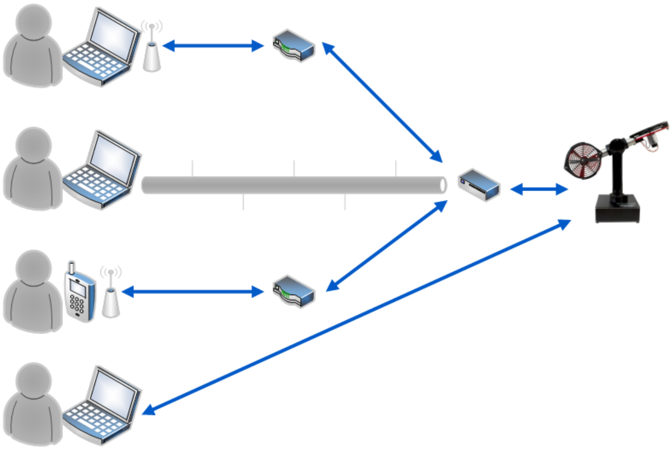


Figure: High-level system block diagram.

- ADP utilizes a linearized state-space model of the system
- Half-Quadcopter

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-K_{sp}}{J_p} & 0 & \frac{-D_p}{J_p} & 0 \\ 0 & 0 & 0 & \frac{-D_y}{J_y} \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{pp}}{J_p} & \frac{-K_{pp}}{J_p} \\ \frac{K_{yy}}{J_y} & \frac{-K_{yy}}{J_y} \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} \tag{18}$$

- 2-DOF Helicopter

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{K_{sp}}{J_p} & 0 & -\frac{D_p}{J_p} & 0 \\ 0 & 0 & 0 & -\frac{D_y}{J_y} \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{pp}}{J_p} & \frac{K_{py}}{J_p} \\ \frac{K_{yp}}{J_y} & \frac{K_{yy}}{J_y} \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} \tag{19}$$

MATLAB Simulation

Results

- Specified desired trajectories and ADP constants
- Exactly simulating coupling is difficult which hindered more realistic results
- Simulation results limited due to approximate linear system model

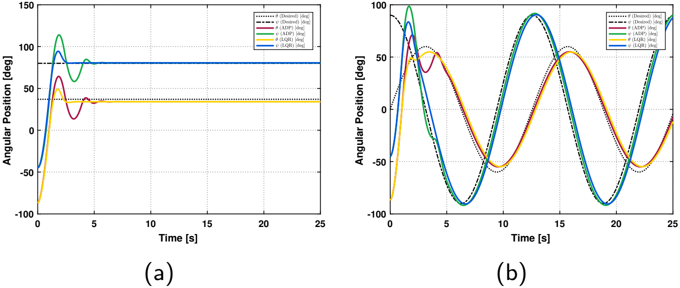
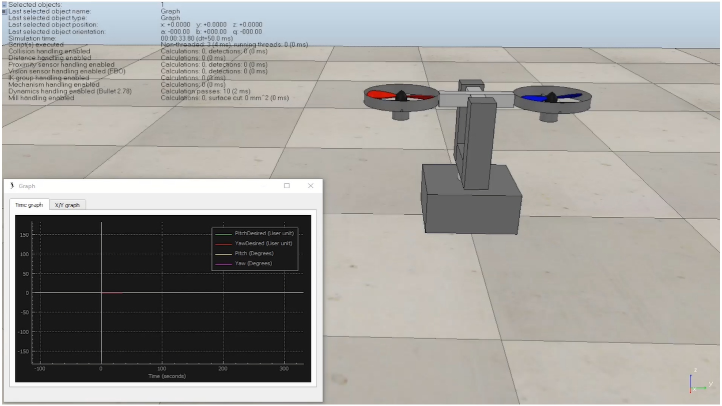


Figure: MATLAB simulation results for 2-DOF helicopter with (a) constant desired trajectories and (b) sinusoidal desired trajectories.

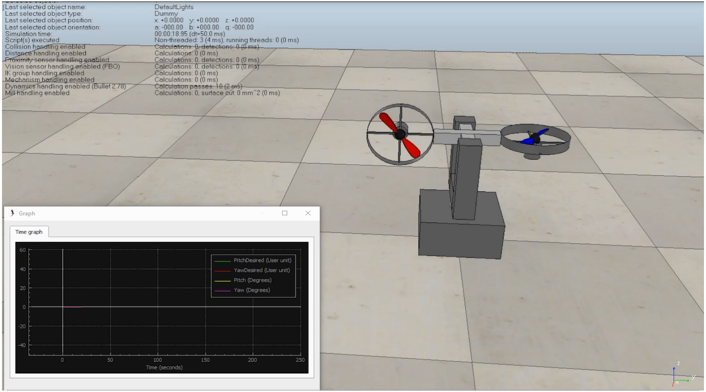
- Integrated development environment offering tools to experiment with robotic prototypes through a wide array of control techniques
- Provides a graphical result realizing the Quanser AERO through software
- Utilized in industry for simulating factory automation systems
- Very limited resources available
- Real-time algorithms frequently limited by processing power of the host system

V-REP

Results - Half-Quadcopter



BRADLEY University

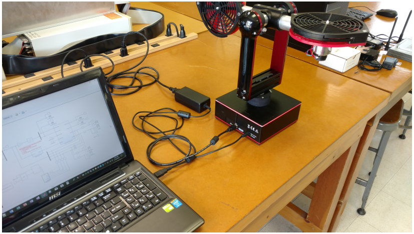


Last selected object name:
DefaultLights
Last selected object type:
Dummy
Last selected object position:
x: +0.0000 y: +0.0000 z: +0.0000
Last selected object orientation:
a: -000.00 b: +000.00 g: -000.00
Simulation time:
00:00:18.95 (dt=50.0 ms)
Script(s) executed
Non-threaded: 3 (4 ms), running threads: 0 (0 ms)
Collision handling enabled
Calculations: 0, detections: 0 (0 ms)
Distance handling enabled
Calculations: 0 (0 ms)
Proximity sensor handling enabled
Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)
Calculations: 0, detections: 0 (0 ms)
IK group handling enabled
Calculations: 0 (0 ms)
Mechanism handling enabled
Calculations: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78)
Calculation passes: 10 (2 ms)
Mill handling enabled
Calculations: 0, surface cut: 0 mm^2 (0 ms)
Graph
Time graph
PitchDesired (User unit)
YawDesired (User unit)
Pitch (Degrees)
Yaw (Degrees)
60
40
20
0
-20
-40
0
50
100
150
200
250
Time (seconds)

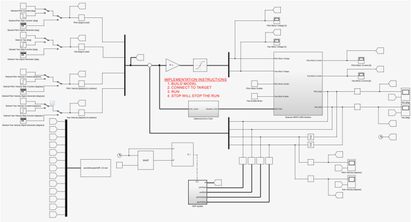
- Graphical programming language that uses a model-based approach
- Algorithm must be converted to a model in order to use
- Simulink and its various support packages can generate C-code for other application uses
- Widely supported in the automotive industry for C-code generation

Simulink
Modeling

- ADP can be structured in the form of a Simulink model by adapting the previous MATLAB simulation code
- Quanser AERO can be controlled directly via Simulink and licensed Quanser software

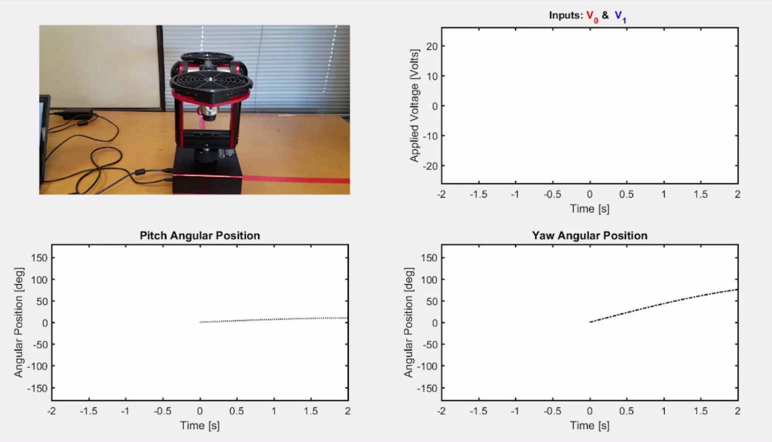


(a)



(b)

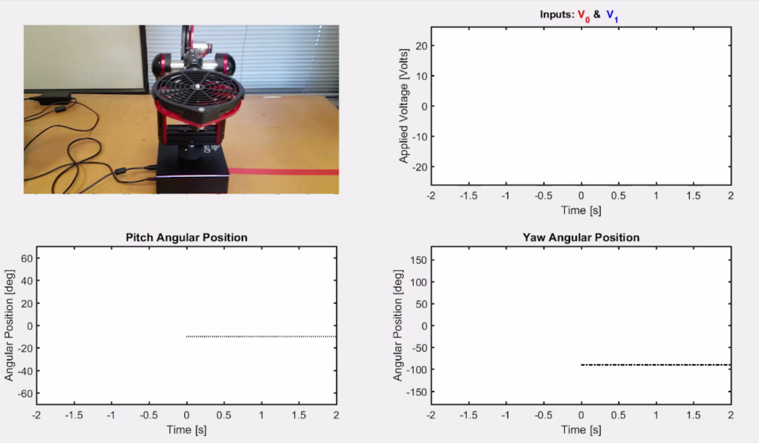
Figure: (a) Laptop connected directly to Qunaser AERO via USB. (b) Simulink model of ADP.



Inputs: V0 & V1
Applied Voltage [Volts]
20
10
0
-10
-20
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
Time [s]
Pitch Angular Position
Angular Position [deg]
150
100
50
0
-50
-100
-150
Time [s]
Yaw Angular Position
Angular Position [deg]
Time [s]

Simulink

Results - 2-DOF Helicopter



BRADLEY University

Raspberry Pi 3

Overview

- Quanser AERO can be controlled via an embedded system through a different interfacing panel
- QFLEX 2 Embedded panel utilizes SPI communication between the embedded system and the Quanser AERO
- To realize ADP feasibility, ADP was implemented for use on a Raspberry Pi 3
- Simulink support packages allow C-code generation compatible with a Raspberry Pi and its GPIO pins

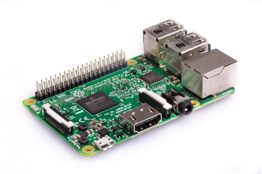
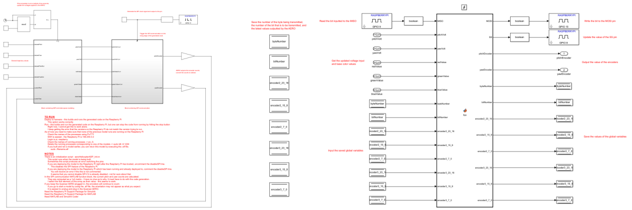


Figure: Raspberry Pi 3. Image courtesy of the Raspberry Pi Foundation [2].

- Utilize the Simulink model of ADP
- Model the SPI communication by correlating information between the Quanser AERO and ADP outputs
- To our knowledge, this is the first time a Raspberry Pi controlled a Quanser AERO using Simulink



(a) (b)

Figure: (a) High-level Simulink model of ADP and SPI communication. (b) SPI communication subsystem.

- Raspberry Pi is accessible through MATLAB and Simulink support packages
- Simulink C-code generation can result in unexpected results through unknown compiling solutions
- Raspberry Pi GPIO pins can only produce a 10 kHz clock signal
- Generated code can be executed on Raspberry Pi via PuTTY
- Once C-code is implemented on the Raspberry Pi, there is no easy way to modify the desired pitch and yaw

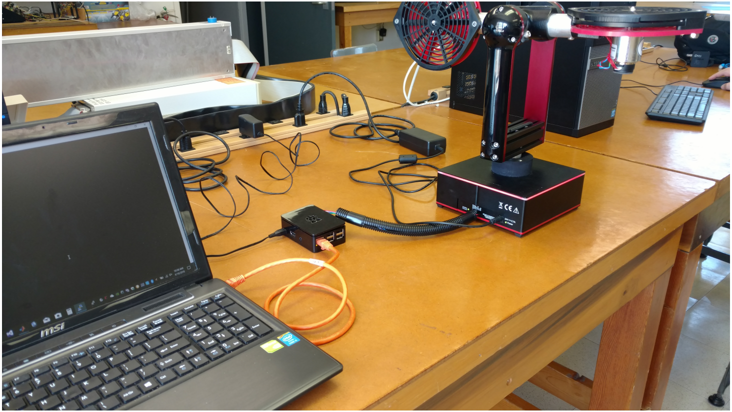


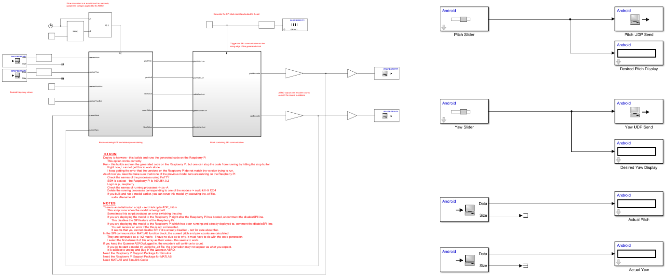
Figure: System connections for the Quanser AERO to be controlled via a Raspberry Pi.

Android Application

Overview

- No easy way to modify C-code generated for the Raspberry Pi
- Some sort of communication is needed between the Raspberry Pi and another external device
- Simulink offers a support package capable of compiling Android applications
- Application makes use of local network to stream commands from phone to Raspberry Pi
- Allows remote articulation of pitch and yaw through ADP

- Modify Raspberry Pi Simulink model to accept/receive UDP packets
- Design an Android smart phone application to accept/receive UDP packets

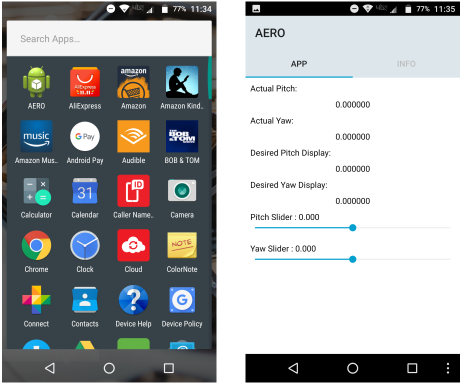


(a) (b)

Figure: (a) Modified Raspberry Pi Simulink model. (b) Simple Android model for setting desired pitch and yaw.

Android Application

Results



(a) (b)

Figure: (a) Android application created on smart phone. (b) Running application to change desired pitch and yaw.

Android Application

Results

- Simulated ADP for the Quanser AERO in MATLAB
- Designed V-REP testing platforms for the Quanser AERO
- Implemented ADP in Simulink for direct-use on the Quanser AERO
- Integrated ADP to the Raspberry Pi through SPI communication
- Designed a simple Android smart phone application which can communicate with the Raspberry Pi
- Provided a framework for Bradley University in using the the Quanser AERO for further research

- Quanser AERO provides a quasi-linear system for research
- Questionable advantages of using ADP for this system due to higher complexity
- Latency issues can arise between MATLAB and V-REP with complex control strategies
- C-code generation using Simulink can be difficult if the model is not ideal for the compiler
- Raspberry Pi GPIO pins are limited to approximately 10 kHz when using generated C-code

Future Directions

- Research a more advanced reinforcement learning strategy that does not utilize a system model
- Implement ADP on a more chaotic or nonlinear system
- Optimize ADP calculation times such that MATLAB and V-REP can execute in real-time
- Research increasing GPIO pin frequency of the Raspberry Pi
- Discover methods to make a more compiler-friendly ADP Simulink model

References I

Federal Aviation Administration. "Helicopter Flying Handbook". English. In: *Journal of Electrical Engineering Technology* (2012). Helicopter Dynamics, pp. 2–14. URL: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/helicopter_flying_handbook/media/hfh_ch02.pdf.

Raspberry Pi Foundation. *Raspberry Pi Model 3 B*. Website. 2018. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

W. Gao and Z. P. Jiang. "Data-driven adaptive optimal output-feedback control of a 2-DOF helicopter". In: *2016 American Control Conference (ACC)*. July 2016, pp. 2512–2517. DOI: 10.1109/ACC.2016.7525294.

References II

Eswarmurthi Gopalakrishnan. "Quadcopter Flight Model and Control Algorithms". In: *Czech Technical University - Department of Control Engineering*. May 2016. DOI: 10.1109/CDC.2011.6160521.

M. Hernandez-Gonzalez, A.Y. Alanis, and E.A. Hernandez-Vargas. "Decentralized discrete-time neural control for a Quanser 2-DOF helicopter". In: *Applied Soft Computing*. Feb. 2012, pp. 2462–2469. DOI: 10.1016/j.asoc.2012.02.016.

Quanser Inc. *QFLEX 2 Embedded for Quanser AERO*. Datasheet. 2016.

Quanser Inc. *Quanser AERO*. Datasheet. 2016.

Quanser Inc. *Quanser AERO*. Website. 2018. URL: https://www.quanser.com/products/quanser-aero/.

Quanser Inc. *Quanser AERO - 2-DOF Lab Guide*. Tutorial. 2016.

References III

- Quanser Inc. *User Manual - Quanser AERO Experiment*. Datasheet. 2016.
- E. Kayacan and M.A. Khanesar. “Recurrent interval type-2 fuzzy control of 2-DOF helicopter with finite time training algorithm”. In: *IFAC-PapersOnLine*. July 2016, pp. 293–299. DOI: 10.1016/j.ifacol.2016.07.977.
- Teppo Luukkonen. “Modelling and control of quadcopter”. In: *2014 American Control Conference*. Vol. Aalto School of Science. June 2011.
- S. Miah. “Value iteration based approximate dynamic programming for mobile robot trajectory with persistent inputs”. In: ().

- R.G. Subramanian and V.K. Elumalai. "Robust MRAC augmented baseline LQR for tracking control of 2-DOF helicopter". In: *Robotics and Autonomous Systems*. Aug. 2016, pp. 70–77. DOI: 10.1016/j.robot.2016.08.004.
- J.H. Moon W. Chang and H.J. Lee. "Fuzzy Model-Based Output-Tracking Control for 2 Degree-of-Freedom Helicopter". English. In: *Journal of Electrical Engineering Technology* 12.00.1 (2017). Quanser product(s): 2 DOF Helicopter, pp. 1921–1928. ISSN: 1975-0102. URL: http://www.jeet.or.kr/LTKPSWeb/uploadfiles/be/201705/290520170957344003750.pdf.

Discussion

Introduction	Motivation Project Objectives Abbreviations Mathematical Symbols
ADP	Overview Mathematics Flowchart
Quanser AERO	Overview Coordinates Coupling Effect Methods of Control
Modeling	State-Space Models
MATLAB Simulation	Results
V-REP	Overview Results - Half-Quadcopter Results - 2-DOF Helicopter

Simulink	Overview Modeling Results - Half-quadcopter Results - 2-DOF Helicopter
Raspberry Pi 3	Overview Modeling Research Findings Results
Android Application	Overview Modeling Results
Conclusion	Objectives Research
Future Directions	
References	

BRADLEY University